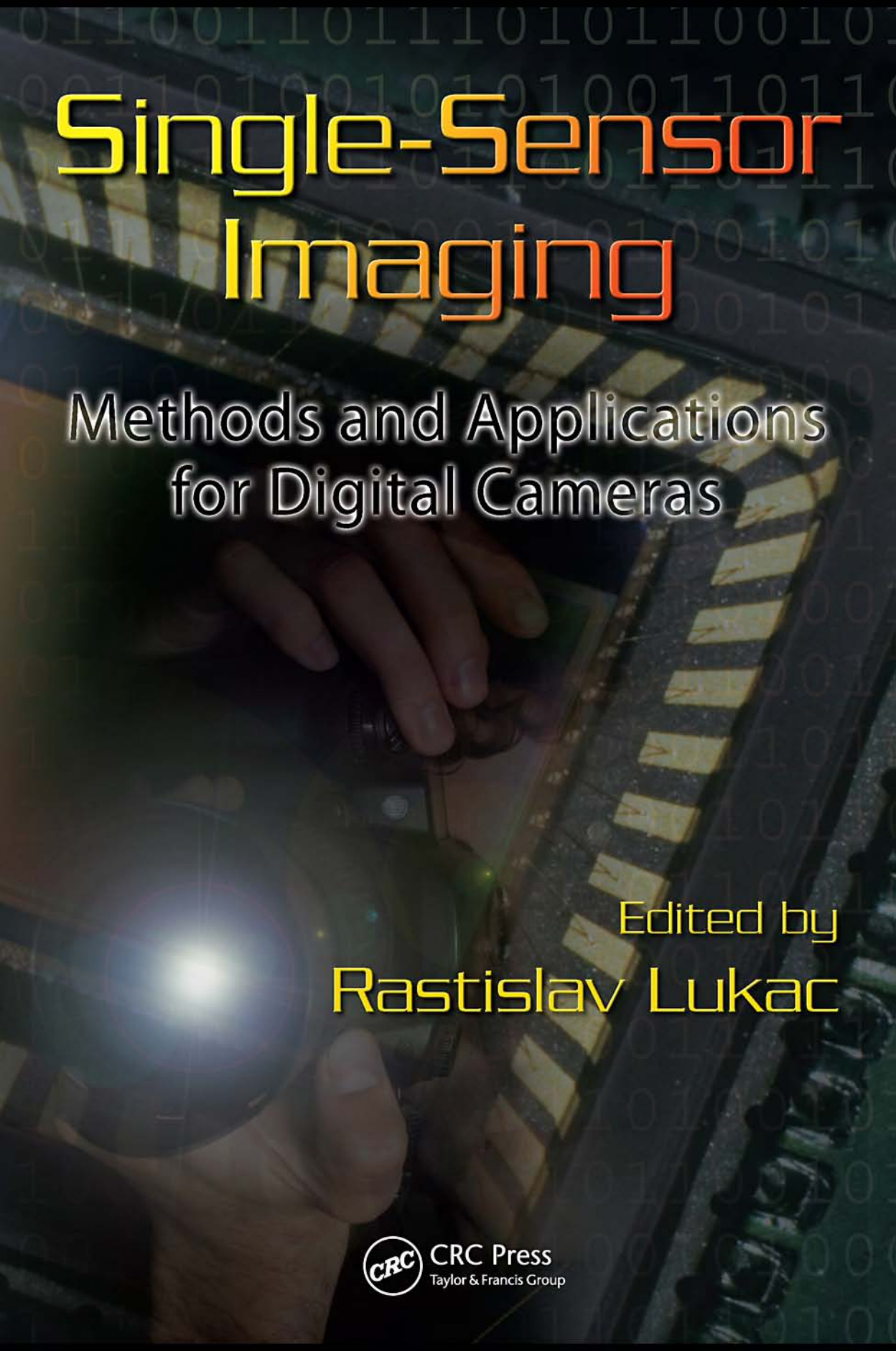

Single-Sensor Imaging
Methods and Applications for Digital Cameras
Edited by
Rastislav Lukac
CRC Press
Taylor & Francis Group

Single-Sensor Imaging

Methods and Applications for Digital Cameras

IMAGE PROCESSING SERIES

Series Editor: Phillip A. Laplante, Pennsylvania State University

Published Titles

Adaptive Image Processing: A Computational Intelligence Perspective
Stuart William Perry, Hau-San Wong, and Ling Guan

Color Image Processing: Methods and Applications
Rastislav Lukac and Konstantinos N. Plataniotis

Image Acquisition and Processing with LabVIEW™
Christopher G. Relf

Image and Video Compression for Multimedia Engineering
Second Edition
Yun Q. Shi and Huiyang Sun

Multimedia Image and Video Processing
Ling Guan, S.Y. Kung, and Jan Larsen

Shape Analysis and Classification: Theory and Practice
Luciano da Fontoura Costa and Roberto Marcondes Cesar Jr.

Single-Sensor Imaging: Methods and Applications for Digital Cameras
Rastislav Lukac

Software Engineering for Image Processing Systems
Phillip A. Laplante

Single-Sensor Imaging

Methods and Applications for Digital Cameras

Edited by

Rastislav Lukac

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-1-4200-5452-1 (Hardcover)

Library of Congress Cataloging-in-Publication Data

Single-sensor imaging : methods and applications for digital cameras / editor, Ratislav Lukac.
p. cm. -- (Image processing series ; 9)
Includes bibliographical references and index.
ISBN 978-1-4200-5452-1 (hardback : alk. paper)
1. Digital cameras. 2. Image processing--Digital techniques. 3. Image converters. I. Lukac, Rastislav.

TR256.S556 2009
771.3'3--dc22 2008026505

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Artists can color the sky red because they know it's blue. Those of us who aren't artists must color things the way they really are or people might think we're stupid.

—Jules Feiffer, cartoonist and satirist

Dedication

To my wife Ivana

Preface

Over the past two decades, advances in hardware and software technology have allowed for massive sales of consumer electronics based on the concept of converting analog information into its digital form. As in many other application areas where digital devices have replaced their analog predecessors, manufacturers and consumers have been losing interest in conventional film cameras and have been turning instead to digital cameras. This is mainly due to the fact that capturing and developing photos using chemical and mechanical processes cannot provide users with the conveniences of digital cameras which record, store and manipulate photographs electronically using image sensors and built-in computers. Features such as displaying an image immediately after it is recorded, the capacity to store thousands of images on a small memory device and the ability to delete images from this device in order to allow its further re-use, and the ability to edit images and even record them with sound make digital cameras very attractive consumer electronic products.

To create an image of a scene, digital cameras use a series of lenses that focus light onto a sensor which samples the light and records electronic information which is subsequently converted into digital data. The sensor is an array of light-sensitive spots, called photosites, which record the total intensity of the light that strikes their surfaces. Unfortunately, common image sensors are monochrome devices which cannot record color information. Among existing technologies developed to overcome the problem, single-sensor imaging offers trade-offs among performance, complexity and cost. Thus, most of today's digital cameras are single-sensor devices which capture visual scenes in color using a monochrome image sensor in conjunction with an array of color filters.

It should not be surprising that single-sensor digital camera imaging is considered one of the most rapidly developing research and application fields and numerous commercial products capitalizing on its principles have already appeared in diverse market applications. The extreme and still increasing popularity of consumer single-sensor digital cameras boosts research activities in the fields of digital color image acquisition, processing, and storage. Single-sensor camera image processing methods are becoming increasingly important due to the development and proliferation of emerging digital camera imaging applications and commercial devices, such as consumer digital still and video cameras, image-enabled mobile phones and personal digital assistants, sensor networks, surveillance and automotive apparatus. The surge of emerging applications, such as digital photography, visual communications, machine vision, multimedia, digital cinema, art, visual surveillance, medical imaging and astronomy, suggests that the demand for single-sensor imaging and digital camera image processing solutions will grow considerably in the next decade.

The purpose of this book is to fill the existing gaps in the literature and comprehensively cover the system design, implementation, and application aspects of single-sensor imaging and digital camera image processing. Due to rapid developments in specialized areas of

single-sensor imaging and digital camera image processing, the book is a contributed volume in which well-known experts deal with specific research and application problems. It presents both the state-of-the-art and the most recent trends in digital camera imaging and applications. It serves the needs of different readers at different levels. It can be used as a textbook in support of a graduate course in digital imaging and visual data processing or as a stand-alone reference for graduate students, researchers and practitioners. For example, a researcher can use it as an up-to-date reference since it offers a broad survey of the relevant literature. A development engineer and technical manager may find it useful in the design and implementation of various digital camera image and video processing tasks.

This book details recent advances in single-sensor imaging and digital camera image processing methods and explores their applications. The book begins by focusing on single-sensor imaging fundamentals, a reusable embedded software platform for versatile digital cameras, and digital camera image processing chain design. The next part of the book presents optical antialiasing filter design, spatio-spectral sampling and color filter array design, and mosaicking / demosaicking for multispectral digital cameras. Moving along the camera image processing pipeline, this book targets frequency-domain analysis of color filter array sampling for the design of demosaicking algorithms, linear minimum mean square error demosaicking, and color filter array image analysis for joint demosaicking and denoising. This is followed by automatic white balancing, enhancement of digital photographs using color transfer techniques, and exposure correction. The next part of the book focuses on image storage issues, targeting three areas: digital camera image storage formats, modelling of image processing pipelines from a data compression point of view, and lossless compression of color mosaic images and videos. Then, the reader's attention is turned to optional, but frequently used steps in the camera image processing pipeline such as automatic red-eye removal for digital photography and single-sensor image resizing. Finally, the remaining chapters explore video processing approaches across a broad spectrum of single-sensor imaging applications ranging from video-demosaicking, through simultaneous demosaicking and resolution enhancement, to image and video stabilization.

Chapters 1 through 3 discuss concepts and technologies which allow for effective design and high performance of single-sensor imaging devices. *Single-sensor digital color imaging fundamentals* are essential for understanding image formation using a color filter array and a monochrome image sensor. As demonstrated by numerous examples, despite the fact that finished digital photographs are achieved from captured sensor data through extensive image processing, they often suffer from various visual impairments due to the shortcomings of image acquisition systems, various constraints imposed on imaging devices, and a lack of information during image processing. To improve visual quality, processing solutions should be able to fully use various image characteristics. A *reusable embedded software platform for versatile single-sensor digital cameras* has an important role in designing an imaging device, as it usually supports both attractive features in user operation mode and calibration / test functions in engineering mode. Using such a platform, embedded software designers can easily capture the whole view of the camera hardware architecture without being sidetracked by the study of detailed hardware specifications. The embedded software architecture allows for fast stepping into practical camera design. In addition, the embedded self-calibration flow and sensor/shutter calibration algorithms give a valuable reference to efficiently build a commercial camera for mass production. In practice, the problem of

digital camera image processing chain design is usually seen as taking relatively simple, well-known image processing operations and staging them in a manner that produces the best synergistic effects. In an image processing chain that transforms digital camera raw sensor image data into a full-color fully processed image, the possible orderings of individual operations and associated implementation details have a great impact on both image quality and computational efficiency. Image processing operations that are highly effective may not be viable candidates for image processing chains in constrained computing environments. Therefore, the image processing task is to balance the opposing requirements of desirable image quality and modest computing resource use.

Chapters 4 through 6 are intended to cover the basics of and review recent advances in visual information sampling. Sampled imaging systems such as digital cameras often produce aliasing artifacts. Once an image is sampled, the aliased low-frequency content is difficult to correct automatically because it has similar characteristics as actual low-frequency content. To prevent such artifacts, most cameras use *optical antialiasing filters* to band-limit the optical image spatial frequencies. Limiting the image spatial frequencies is equivalent to blurring the image, so these filters are sometimes called blur filters. Analysis of antialiasing filter performance must include all capture system parameters, particularly the pixel aperture size, lens performance, and interpolation technique. Ideally the antialiasing filter and the interpolation technique should be co-optimized to maximize system modulation transfer function below the Nyquist frequency and minimize system modulation transfer function above the Nyquist frequency. In single-sensor digital cameras, visual information is sampled by a color filter array and an image sensor. *Spatio-spectral sampling and color filter array design* can be seen as a problem of simultaneously maximizing the spectral support of luminance and chrominance channels subject to their mutual exclusivity in the Fourier domain. Key to this design paradigm is the notion that the measurement process, an inner product between the color filter array and the image data, induces a modulation in the frequency domain. Modulating the chrominance spectra away from the baseband luminance channel constitutes a basis for a design of a physically realizable color filter array by specifying these modulation frequencies directly. This method generates panchromatic color filter arrays that mitigate aliasing and admit favorable trade-offs between demosaicking quality and computational efficiency. It is probably no surprise that some ideas from single-sensor imaging can be adopted in multispectral imaging which expands color cameras' capability to capture spectral information at multiple wavelengths other than that of visible light. *Mosaicking and demosaicking in the design of multispectral digital cameras* focuses on the design of multispectral filter arrays and the development of the corresponding demosaicking algorithms. The binary tree-driven multispectral filter array generation process guarantees that the pixel distributions of different spectral bands are uniform and highly correlated. These spatial features facilitate the design of the generic demosaicking method based on the same tree, which considers three interrelated issues: band selection, pixel selection, and interpolation. The development of a generic demosaicking algorithm enables cost-effective multispectral imaging.

Chapters 7 through 9 address important issues in the area of demosaicking which is a crucial step in the single-sensor imaging pipeline to restore the color image from the raw mosaic sensor data. *Color filter array sampling of color images* involves spatial domain multiplexing of three or more color components of a color image, each on a subset of the

lattice consisting of all sensor elements. In the frequency domain, this same operation can be viewed as the multiplexing of a luma component at baseband and two or more chrominance components centered at certain spatial modulation frequencies. This view leads to efficient demosaicking algorithms that would not normally be evident from the spatial domain representation. *Linear minimum mean square error demosaicking* constitutes another computationally-efficient method for reconstructing the color information of a captured image. This method is applicable to single-sensor data obtained using different color filter arrays. It takes advantage of a model of spatio-chromatic sampling applicable to both the human visual system and digital cameras and allows the construction of both linear and data-adaptive demosaicking solutions. Fourier and wavelet-packet filterbank-based *color filter array image analysis for joint demosaicking and denoising* reveals that the observed data consists of a mixture of baseband luminance signals, spectrally shifted difference images, and noise. It is quite well known that noise can significantly affect the perceptual quality of the output from a digital camera. Since preserving the sharpness of edges and textures is a key factor in demosaicking, noise is often amplified by demosaicking as noise patterns may form false edge structures. The problem of estimating the noise-free image signal given a set of incomplete observations of pixel components that are corrupted by noise can be approached statistically from a point of view of Bayesian statistics. Such an approach allows for different design regimes that can be thought of as simultaneous demosaicking and image denoising and can allow for solutions with better performance than when these two processing steps are handled separately.

Chapters 10 through 12 are focused on color and exposure corrections. *White balancing* is used to adjust color in the captured image in order to compensate for shifts from perceived color in the scene due to the ambient illumination. In manual mode, users can choose from white balance settings predetermined by a camera manufacturer for typical lighting scenarios or define a unique white balance reference. In automatic mode, digital cameras can use special sensors to dynamically detect the color temperature of the ambient light and compensate for its effects. Cost-effective cameras achieve color-balancing effects solely using an image processing algorithm which sets, in a fully-automated manner, the white balancing parameters based on the image content and statistics. *Enhancement of digital photographs using color transfer techniques* constitutes an advanced approach to altering the color of captured images. This approach transforms the captured image so that its final colors match the palette of the target image regardless of the content of the pictures. One way to treat this recoloring problem is to find a one-to-one color mapping that is applied to every pixel in the captured image, producing an image which is identical in every aspect to the original captured image, except that it now exhibits the same color statistics, or palette, as the target image. *Exposure correction in imaging devices* is essential to compensate an image for improper exposure of the sensor to light. Digital consumer devices make use of ad-hoc strategies and heuristics to derive exposure setting parameters. Typically such techniques are completely blind with respect to the specific content of the scene. Unfortunately, it is not rare for images to be acquired with a nonoptimal or incorrect exposure due to complex visual scene lighting conditions or poor optics, resulting in too dark or bright images. Correcting the exposure thus often means reproducing the most important regions, according to contextual or perceptive criteria, with intensities more or less in the middle of the possible range.

Chapters 13 through 15 address the important issues of image storage and data compression. Adopting the same standard image file format enables readers, including computers and photo printers, to make use of the image data. Current *digital camera image storage formats* evolved over time, adopting some features first introduced in their predecessors. According to the standard, captured files are named and organized into folders, and contain image data along with metadata created by the camera. One of two current standards stores the sensor image data prior to demosaicking, thus providing higher image quality while requiring that the reader performs camera image processing. The other standard format stores the developed photograph, after the complete camera image processing has been performed on sensor data, in an image format compatible with existing imaging hardware and software. The position of the image compression step with respect to the demosaicking step can be used as the basis for *modelling of image processing pipelines in single-sensor digital cameras*. The current development of digital camera image processing is typically guided by empirical performance evaluations. However, results provided by empirical evaluations are usually limited to the training set and are often inconclusive. Therefore, taking advantage of mathematical models linking the performance of the camera image processing to image content and algorithm settings can allow for better understanding of design issues. Focusing on image quality and computational efficiency issues, *lossless compression of single-sensor mosaic images and videos* is of paramount interest in a number of applications, ranging from digital photography and cinema to medical imaging. Technically, lossless compression of color filter array mosaic images poses a unique challenge of spectral decorrelation of spatially interleaved samples of three or more sampling colors. Among a number of reversible lossless transforms that can remove statistical redundancies in both spectral and spatial domains, Mallat wavelet packet transform constitutes an ideal solution which is extendable for lossless compression of a time sequence of color filter array mosaic frames.

Chapters 16 and 17 deal with two popular optional steps in the camera imaging pipeline. *Automatic red-eye detection and removal* is needed to eliminate red-eye effects in digital or digitized film photographs. These effects are caused by the reflection of the blood vessels in the retina when a strong and sudden light, such as the camera flash, strikes the eye. A common technique to reduce red-eye effects is to adopt multiple flashes to contract the pupils before the final shot. However, the flash consumes a significant amount of power and the effects cannot be completely removed. Therefore, red-eye removal techniques, which aim at locating and correcting red-eyes in captured photos digitally using image processing and pattern recognition solutions, are implemented directly in cameras, or externally in printer drivers and software running on a personal computer. *Image resizing* is another optional step in the camera imaging pipeline. The spatial resolution of digital camera images is often modified by the user or as a result of application constraints. For instance, images are downsampled to reduce bandwidth for their transmission, to fit more pictures on the storage media, or to obtain the effective resolution for printing. On the other hand, images may be upsampled to overcome the limitations in optical capabilities of inexpensive cameras with fixed-zoom lenses or to allow close visual inspection of fine details and areas of interest. The specific spatial resolution of the visual input may also be required in some processing steps such as scene analysis and object recognition to achieve desired performances.

Finally, Chapters 18 through 20 discuss various issues in single-sensor video processing. Recently, *video-demosaicking* has gained the interest of the digital camera imaging community. With advances in hardware and software, more and more cameras allow for the recording of digital video. Since digital video represents a three-dimensional image signal or a time sequence of two-dimensional images, video-demosaicking goes one step beyond traditional spatial demosaicking in order to utilize spectral, spatial, temporal, and motion characteristics during processing to produce high-quality color video. Obviously, a multiframe approach is also essential for *simultaneous demosaicking and resolution enhancement*. The goal of multiframe processing here is to estimate the high resolution image from a collection of low resolution images which typically suffer from noise, and warping, blurring, downsampling, and color-sampling effects. To mitigate the shortcomings of imaging systems, the presented multiframe approach solves both demosaicking and resolution enhancement problems in a joint fashion offering improved performance. *Image and video stabilization* has become more and more important due to the enhancement of camera portability which usually results in less stable image and video capturing. Unwanted position fluctuations of the camera affect the visual quality of captured image sequences and reduce the performance of automated machine vision systems. In order to accurately predict both camera fluctuations and motion of objects in the scene, and to compensate for unwanted shakes of the camera, digital cameras use various solutions such as optical, electronic and digital image stabilization. Among these three popular approaches, digital image stabilization is the most cost-effective as it is implemented purely in software and requires no motion sensors or adjustable prisms.

The bibliographic links included in all chapters of the book provide a good basis for further exploration of the presented topics. The volume includes numerous examples and illustrations of single-sensor image processing results, as well as tables summarizing the results of quantitative analysis studies. Complementary material is available online at *http://www.colorimageprocessing.org*.

I would like to thank the contributors for their effort, valuable time and motivation to enhance the profession by providing material for a wide audience while still offering their individual research insights and opinions. I am very grateful for their enthusiastic support, timely response and willingness to incorporate suggestions from me and from other contributing authors who served as reviewers and helped to improve the quality of contributions. I also appreciate my colleagues at Epson Edge, particularly Ian Clarke, Sohaib Sajid and Graham Sellers for their understanding and support. Finally, a word of appreciation for CRC Press / Taylor & Francis for giving me the opportunity to edit a book on single-sensor imaging. In particular, I would like to thank Dr. Phillip A. Laplante for his encouragement, Nora Konopka for initiating and supporting this project, and Shashi Kumar for his LaTeX assistance.

Rastislav Lukac
Epson Canada Ltd., Toronto, ON, Canada
E-mail: lukacr@colorimageprocessing.com
Web: www.colorimageprocessing.com

The Editor

Rastislav Lukac (www.colorimageprocessing.com) received M.S. (Ing.) and Ph.D. degrees in Telecommunications from the Technical University of Kosice, Slovak Republic, in 1998 and 2001, respectively. From February 2001 to August 2002, he was an assistant professor with the Department of Electronics and Multimedia Communications at the Technical University of Kosice. From August 2002 to July 2003, he was a researcher with the Slovak Image Processing Center in Dobsina, Slovak Republic. From January 2003 to March 2003, he was a postdoctoral fellow with the Artificial Intelligence and Information Analysis Laboratory, Aristotle University of Thessaloniki, Greece. From May 2003 to August 2006, he was a postdoctoral fellow with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada. Since September 2006, he has been a senior image processing scientist at Epson Edge, Epson Canada Ltd., Toronto, Canada. He is a contributor to seven books and he has published over 200 papers in the areas of digital camera image processing, color image and video processing, multimedia security, and microarray image processing.

Dr. Lukac is a member of the Institute of Electrical and Electronics Engineers (IEEE) and IEEE Circuits and Systems, IEEE Consumer Electronics, and IEEE Signal Processing societies. He is an editor of the book *Color Image Processing: Methods and Applications* (Boca Raton, FL, CRC Press / Taylor & Francis, 2006). He is a guest editor of *Real-Time Imaging*, Special Issue on Multi-Dimensional Image Processing, *Computer Vision and Image Understanding*, Special Issue on Color Image Processing, *International Journal of Imaging Systems and Technology*, Special Issue on Applied Color Image Processing, and *International Journal of Pattern Recognition and Artificial Intelligence*, Special Issue on Facial Image Processing and Analysis. He is an associate editor for the *Journal of Real-Time Image Processing*. He serves as a technical reviewer for various scientific journals, and he participates as a member of numerous international conference committees. He is the recipient of the 2003 North Atlantic Treaty Organization / National Sciences and Engineering Research Council of Canada (NATO/NSERC) Science Award, and the Most Cited Paper Award for the *Journal of Visual Communication and Image Representation* for the years 2005–2007.

Contributors

Rastislav Lukac Epson Canada Ltd., Toronto, ON, Canada

Wen-Chung Kao National Taiwan Normal University, Taipei, Taiwan

Hung-Hsin Wu National Taiwan Normal University, Taipei, Taiwan

Sheng-Yuan Lin Cheng-Wei Precision Industry Co., Ltd., Taipei, Taiwan

James E. Adams, Jr. Eastman Kodak Company, Rochester, NY, USA

John F. Hamilton, Jr. Eastman Kodak Company, Rochester, NY, USA

Russ Palum Eastman Kodak Company, Rochester, NY, USA

Keigo Hirakawa Harvard University, Cambridge, MA, USA

Patrick J. Wolfe Harvard University, Cambridge, MA, USA

Lidan Miao University of Tennessee, Knoxville, TN, USA

Hairong Qi University of Tennessee, Knoxville, TN, USA

Wesley E. Snyder North Carolina State University, Raleigh, NC, USA

Eric Dubois University of Ottawa, Ottawa, ON, Canada

David Alleysson Université Pierre-Mendès France, Grenoble, France

Brice Chaix de Lavarène Université Joseph Fourier, Grenoble, France

Sabine Süsstrunk Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

Jeanny Hérault Université Joseph Fourier, Grenoble, France

Edmund Y. Lam The University of Hong Kong, Hong Kong

George S. K. Fung The University of Hong Kong, Hong Kong

François Pitié Trinity College, Dublin, Ireland

Anil Kokaram Trinity College, Dublin, Ireland

Rozenn Dahyot Trinity College, Dublin, Ireland

Sebastiano Battiato University of Catania, Catania, Italy

Giuseppe Messina ST Microelectronics, Catania, Italy

Alfio Castorina ST Microelectronics, Catania, Italy

Kenneth A. Parulski Eastman Kodak Company, Rochester, NY, USA

Robert Reisch Eastman Kodak Company, Rochester, NY, USA

Nai-Xiang Lian Nanyang Technological University, Singapore

Vitali Zagorodnov Nanyang Technological University, Singapore

Yap-Peng Tan Nanyang Technological University, Singapore

Ning Zhang IMAX Corporation, Mississauga, ON, Canada

Xiaolin Wu McMaster University, Hamilton, ON, Canada

Lei Zhang The Hong Kong Polytechnic University, Kowloon, Hong Kong

Francesca Gasparini University of Milano-Bicocca, Milano, Italy

Raimondo Schettini University of Milano-Bicocca, Milano, Italy

Wei Lian The Hong Kong Polytechnic University, Hong Kong

Sina Farsiu Duke University Eye Center, Durham, NC, USA

Dirk Robinson Ricoh Innovations Inc., Menlo Park, CA, USA

Michael Elad The Technion – Israel Institute of Technology, Haifa, Israel

Peyman Milanfar University of California, Santa Cruz, CA, USA

Contents

1

Single-Sensor Digital Color Imaging Fundamentals

Rastislav Lukac

1.1 Introduction

Capturing visual scenes in color and producing digital photographs which are faithful representations of the original scene is quite challenging due to a number of constraints under which digital cameras operate. Differences in characteristics between image acquisition systems and the human visual system constitute an underlying problem in digital color imaging. In today's digital cameras, most challenges result from sampling visual information spectrally, spatially, tonally and temporally.

Using a color filter array, *spectral sampling* reduces available color information to light of certain wavelengths which can be acquired by a monochrome image sensor. *Spatial sampling* reduces the angle of view that the camera sees to a rectangular array of pixels in the captured image. This is realized by an image sensor representing a two-dimensional

array of light-sensitive spots which record the total intensity of the light that strikes their surfaces. *Tonal sampling* characterizes the quantization process used to represent the original continuously-varying visual information by discrete values. Finally, *time sampling* characterizes an exposure of the sensor to the light for a certain amount of time.

Obviously, the first step in achieving high-quality images is to design sampling procedures in a manner which allows for a precise digital representation of various visual scenes. However, as with other real-life imaging systems, this is not quite possible and extensive processing of acquired sampled data is needed to compensate for the shortcomings of an imaging system and to produce digital photographs with a natural appearance matching the original scene.

To facilitate the following discussions on technical challenges in digital cameras, this chapter presents fundamentals of single-sensor color imaging and digital camera image processing. More specifically, Section 1.2 describes popular color image acquisition technologies. Of particular interest are consumer digital cameras which use color filter arrays to capture visual scenes in color using only one monochrome image sensor. Since color filter arrays distinguish this type of digital camera from other image acquisition solutions, this section also discusses color filter array design issues and digital image representations.

Section 1.3 presents image processing pipelines for single-sensor color imaging devices. The data acquired by such devices constitutes a grayscale image with a mosaic structure following the underlying pattern of the color filter array. A pipeline consists of a number of processing steps necessary to produce a finished photograph from the acquired sensor data. Depending on the image storage format and user requirements, such pipelines can be implemented in camera or software running on personal computers. Presented examples of real digital camera images suggest that both the choice and order of processing steps employed in the pipeline greatly influence the quality of produced digital photographs.

Section 1.4 focuses on typical image quality issues in single-sensor digital cameras. These issues mainly relate to the presence of noise introduced into the image during its acquisition and various color shifts and artifacts caused by insufficient image processing.

Section 1.5 discusses important characteristics of color images and videos. Exploring different types of pixel correlations, these characteristics relate to spatial, spectral, structural and temporal properties of the captured visual data and the omission of these characteristics during processing usually results in various visual artifacts in finished images. Thus, using as much of the information available in the image as possible is crucial for achieving high visual quality of captured images.

Finally, this chapter concludes with Section 1.6 by summarizing main single-sensor color imaging and digital camera image processing ideas.

1.2 Color Image Acquisition in Digital Cameras

Digital cameras acquire a scene by first focusing and then transmitting light through the optical system. Once the light reaches the sensor surface, it is sampled by the sensor in order to obtain the corresponding digital representation of the sensor values through

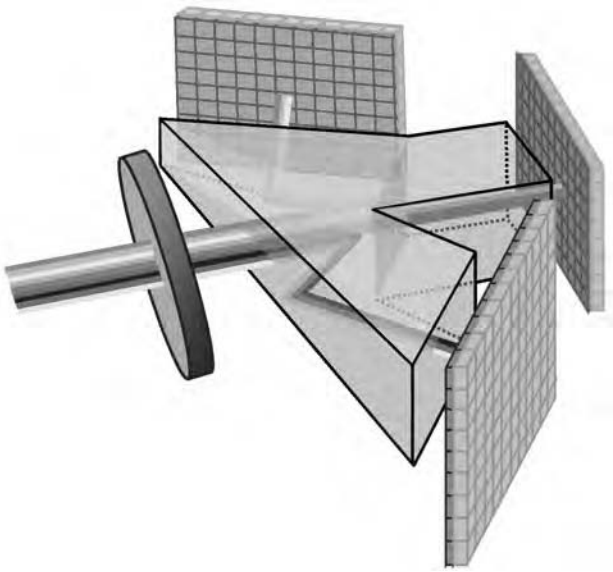


FIGURE 1.1 (See color insert.)

Three-sensor digital camera architecture.

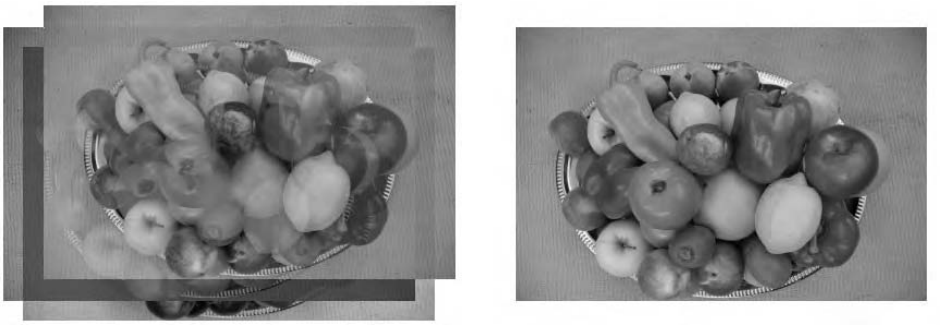
subsequent analog-to-digital conversion. Acquired digital data undergoes a series of image processing operations [1], [2] which are realized by an application-specific integrated circuit and a microprocessor. To reduce various artifacts due to the sampling of visual information, camera manufacturers often place blur filters in the optical system to reduce high-frequency content of the image. Chapter 4 discusses optical issues in detail.

Human vision is based on three types of color photo-receptor cone cells, implying that three numerical components are necessary and sufficient to describe a color [3]. However, common image sensors, such as *charge-coupled devices* (CCD) [4], [5], *charge-injection devices* (CID) [6], [7], or *complementary metal oxide semiconductor* (CMOS) sensors [8], [9], are monochromatic devices. Therefore, to capture color information using such sensors, digital camera manufacturers place a color filter on top of each sensor cell.[1] The two most popular camera technologies for color image acquisition — three-sensor and one-sensor solutions — are described below.

1.2.1 Three-Sensor Digital Cameras

To overcome the sensor's monochromatic nature and capture the visual scene in color, predecessors of today's consumer digital cameras used a *beam splitter* to separate incoming light onto *three optical paths*, each having its own red (R), green (G) or blue (B) color filter with different spectral transmittances [12] and sensor for sampling the filtered light

[1]There also exists a layered sensor which directly captures the complete color information at each spatial location in an image during a single exposure. This is possible by stacking and ordering color filters vertically according to the energy of the photons absorbed by silicon [10], [11]. The layered sensor is an alternative to earlier technology which rotates a series of red, blue and green filters in front of a single sensor in order to record three separate images in rapid succession.



(a) (b)

FIGURE 1.2 (See color insert.)

Three-sensor imaging. (a) Registration of three grayscale images on the left, top and bottom of the figure which were acquired using sensors with red, green and blue color filters, respectively. (b) Final full-color image achieved by registering three sensor images.

(Figure 1.1). Each optical path deals with the same visual information; however, because of the filters, each sensor only responds to one of the primary colors. Thus, each of three sensors acquires a monochromatic image corresponding to one channel of a color image output by a camera. The resulting color image is produced by registering three grayscale (monochromatic) images, requiring precise mechanical and optical alignment. Figure 1.2 shows the registration procedure.

The captured photo can be considered a $K_1 \times K_2$ RGB digital color image $\mathbf{x} : Z^2 \rightarrow Z^3$ representing a two-dimensional matrix of three-component pixels $\mathbf{x}_{(r,s)} = [x_{(r,s)1}, x_{(r,s)2}, x_{(r,s)3}]^T$. As noted, each individual channel of $\mathbf{x}$ is a $K_1 \times K_2$ monochromatic image $x_k : Z^2 \rightarrow Z$, for $k = 1,2,3$. The pixel $\mathbf{x}_{(r,s)}$ represents the color vector [13] indexed by its spatial location (r,s), with $r = 1,2,...,K_1$ denoting the image row and $s = 1,2,...,K_2$ denoting the image column. The value of the R ($k = 1$), G ($k = 2$), and B ($k = 3$) component $x_{(r,s)k}$ defined in the integer domain Z is equal to an arbitrary integer value ranged from 0 to 255 in a standard 8-bits per component representation and denotes the contribution of the k-th primary in $\mathbf{x}_{(r,s)}$. The process of displaying an image creates a graphical representation of the image matrix where the pixel values represent particular colors in the visible spectrum.

1.2.2 Single-Sensor Digital Cameras

It is probably no surprise that the sensor is the most expensive component of the digital camera, usually taking from 10% to 25% of the total cost [14]. To reduce expenses and allow for high sales volumes, current consumer digital cameras use only *one image sensor* covered with a mosaic of color filters to capture all the necessary colors at the same time. Figure 1.3 shows the sensor with a *color filter array* (CFA) proposed by B.E. Bayer in 1976 [15] which has been the most widely used CFA since its introduction. Examples of various CFAs used by camera manufacturers can be found in References [16] and [17], and Chapters 5 and 8. Reference [17] also presents detailed discussions on CFA design issues together with performance evaluations of a number of CFAs.

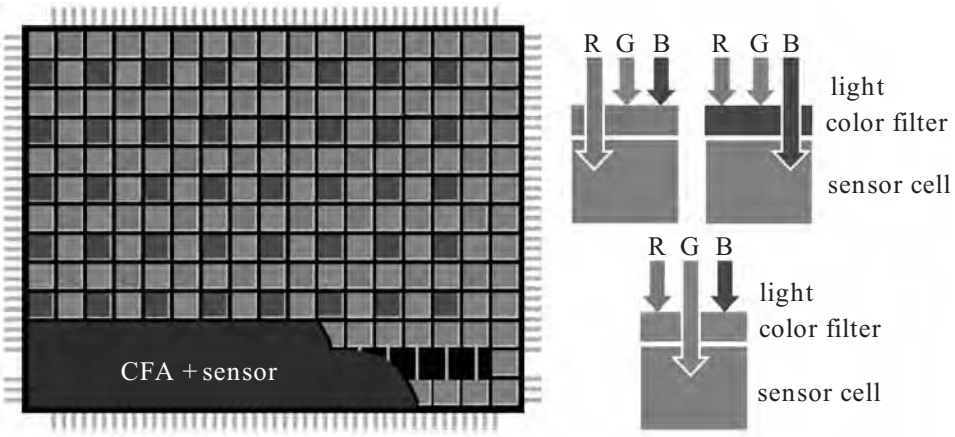


FIGURE 1.3 (See color insert.)

(left) Image sensor covered by a Bayer color filter array. (right) The concept of acquiring the visual information using color filters.

One of the most important features for the design of CFAs is the choice of a color system [17]. Popular CFA configurations are usually constructed using *tristimulus* (RGB) color filters. There also exist configurations based on mixed *primary* and *complementary* colors (e.g., mixtures of magenta, green, cyan, and yellow). Recently, CFAs with four and more colors were introduced; these typically contain white or colors with shifted spectral sensitivity which may be combined with primary or secondary colors. Four or more color systems may produce a more accurate hue gamut compared to tristimulus systems; unfortunately, they often limit the useful range of darker colors [16].

Regardless of which color system has been employed, the acquired CFA sensor readings constitute a $K_1 \times K_2$ monochromatic image $z : Z^2 \rightarrow Z$ with pixel values $z_{(r,s)}$. Figure 1.4a shows a simulated example. The image has a mosaic-like structure dictated by the CFA and is either passed through the camera image processing pipeline to develop a final digital photograph or stored as the so-called *raw camera file*. In either case, the arrangement of color filters in the actual CFA is known, and thus the pixels in the CFA image can be mapped to the corresponding channels of a color image of the same size [2]. For example, a CFA image z acquired using the Bayer CFA with GR phase in odd rows and BG phase in even rows corresponds to a color image $\mathbf{x}$ with RGB pixels $\mathbf{x}_{(r,s)} = [z_{(r,s)}, 0, 0]^T$ for (odd r, even s), $\mathbf{x}_{(r,s)} = [0, z_{(r,s)}, 0]^T$ for (odd r, odd s) and (even r, even s), and $\mathbf{x}_{(r,s)} = [0, 0, z_{(r,s)}]^T$ for (even r, odd s). Two zeros in $\mathbf{x}_{(r,s)}$ indicate two missing components in order to denote their portion to the coloration of the image $\mathbf{x}$ shown in Figure 1.4b which is a color version of the CFA image shown in Figure 1.4a. These two missing components per pixel location must be determined from the adjacent pixels using a digital image processing solution called *demosaicking* [2], [18], which restores the full-color information.[2] Thus, demosaicking is an integral step in the *single-sensor imaging pipeline*. Depending on the employed algorithm, the demosaicked image, such as the one shown in Figure 1.4c, may suffer from color

[2]Other terms known from the literature are demosaicing, color interpolation, and CFA interpolation.

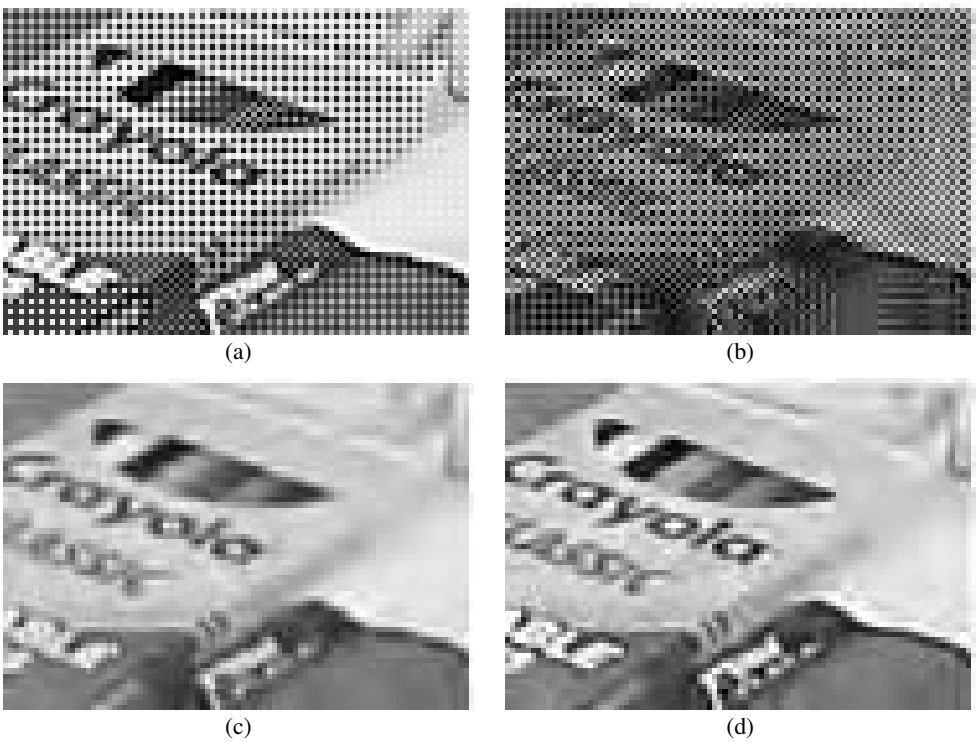


FIGURE 1.4 (See color insert.)

Single-sensor imaging: (a) grayscale mosaic image, (b) color version of the mosaic image, (c) demosaicked full-color image, and (d) postprocessed demosaicked image with improved visual quality.

shifts, aliasing effects, blur, and other visual artifacts. These effects can be suppressed, if not eliminated, via *demosaicked image postprocessing* [19], [20], [21]. Figure 1.4d shows an example of a postprocessed demosaicked image. Apart from demosaicking and demosaicked image postprocessing which constitute, respectively, *mandatory* and *optional* steps in the single-sensor imaging pipeline, the rest of the pipeline is more or less identical to the pipeline in a three-sensor digital camera. Chapter 3 discusses the digital camera image processing chain design in detail.

Another important factor in CFA design is the arrangement of color filters in the array, because it suggests potential cost-effectiveness of color reconstruction by demosaicking, immunity to color artifacts and color moiré, reaction of the array to image sensor imperfections, and immunity to *optical and electrical cross talk* between neighboring pixels. A small *basic repetitive unit* in the CFA usually allows for relatively simple demosaicking. For example, the Bayer CFA shown in Figure 1.3 is constructed by repeating a 2×2 square composed of one red filter, one blue filter, and two green filters located on the diagonal. Given that the wavelength of the green color band is close to the peak of the human luminance frequency response,[3] many CFAs have the higher number of green filters compared

[3] In the literature, G components are often referred to as the luminance whereas R and B components are known as the chrominance.

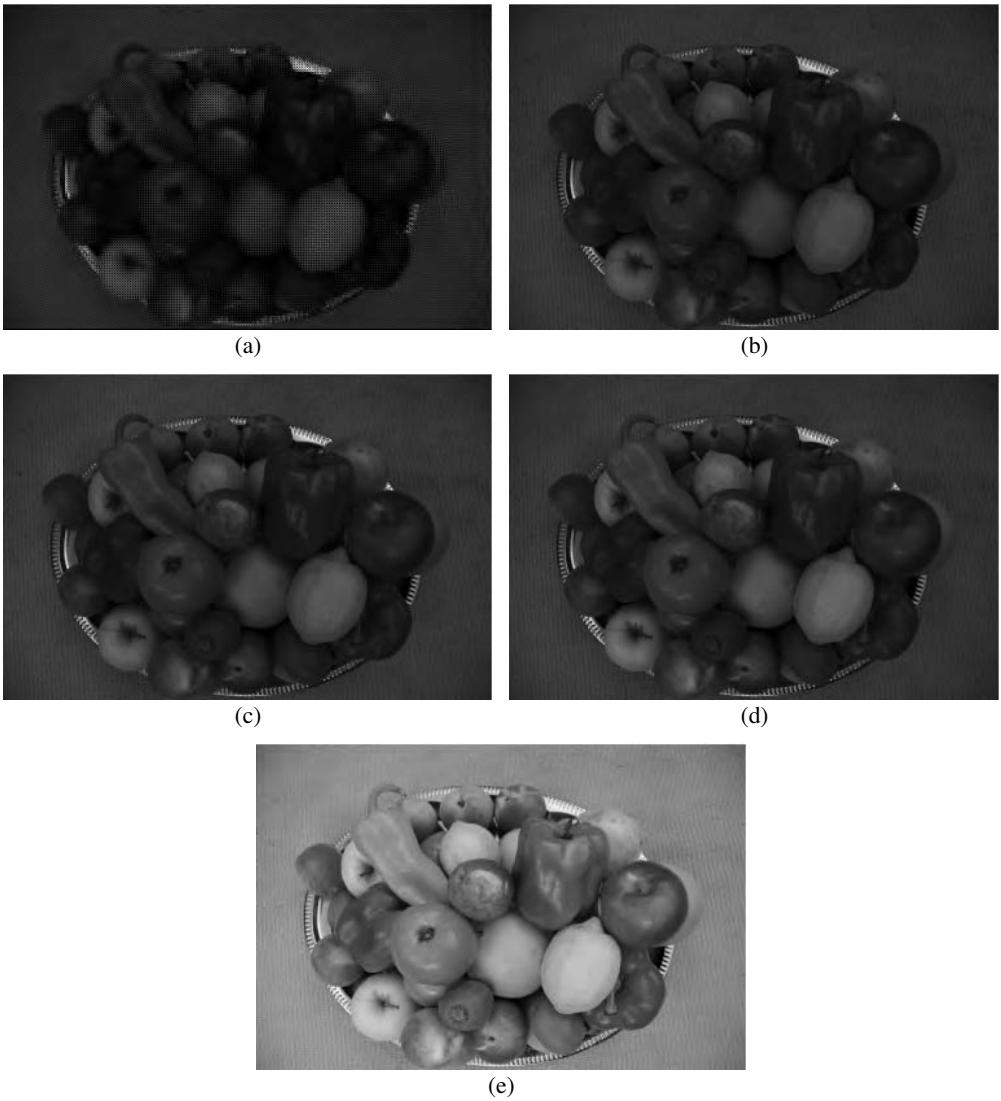
to the amount of other filters in the array in order to reduce the amount of demosaicking artifacts [22]. The sensitivity of the array to color artifacts in the demosaicked image can be reduced by surrounding each color filter with color filters of all other types, thus minimizing the size of the local neighborhood and performing demosaicking in a more local fashion. Introducing some degree of *randomness* or aperiodicity into the arrangement of color filters can make CFAs more robust to color moiré effects; unfortunately, this comes at the expense of increased computational complexity due to the larger size of the minimum repetitive pattern. Image sensor imperfections are typically observed along rows or columns of the sensor cells, suggesting that a diagonal version of the Bayer CFA as well as various diagonal stripe patterns can be more immune. Finally, CFAs with the fixed number of neighbors corresponding to each type of color filters in the CFA are usually more robust against optical and electrical cross talk between neighboring pixels than pseudo-random patterns. This concludes an overview of current CFA design issues. A novel view on the problem of designing CFAs is presented in Chapter 5.

1.3 From Raw Sensor Data to Digital Photographs

As previously discussed, once the sensor image has been acquired, it can be stored as raw data or passed through the camera image processing pipeline to produce a digital photograph. A number of digital cameras, typically digital *single lens reflex* (SLR) cameras, use the former approach by following the *Tagged Image File Format for Electronic Photography* (TIFF-EP) [23]. By applying *lossless compression* to raw sensor image data and storing compressed image data together with *metadata* containing information about the camera settings, TIFF-EP allows for developing high-quality digital photographs on a companion personal computer (PC) using sophisticated solutions, under different settings, and reprocessing the image until certain quality criteria are met. Thus, this approach may have quite different design and performance characteristics compared to the latter one where the sensor image, immediately after its acquisition, undergoes in-camera *real-time* image processing to produce the final image to be typically stored using *Joint Photographic Experts Group* (JPEG) compression [24] in the *Exchangeable Image File* (EXIF) format [25] together with the metadata. Chapter 13 discusses camera formats in detail.

1.3.1 Pipelining Image Processing Solutions

The way an imaging pipeline is constructed can vary significantly between camera and imaging software manufacturers, depending on many factors such as the selection and order of pipelined image processing steps, preferences on visual appearances of digital photographs, implementation constraints, etc. Typically, early processing stages [26] aim at detecting defective pixels caused by a failure of individual sensor photo-elements and correcting them using the concept of image interpolation. A linearization step may be needed if the captured data resides in a nonlinear space due to the involved electronics. In low-exposure images, where both signal and noise levels may be comparable, it is essential to



(a) (b) (c) (d) (e)

FIGURE 1.5 (See color insert.)

Images stored at different stages of the single-sensor camera image processing pipeline: (a) mosaic CFA image, (b) demosaicked image, (c) white-balanced image, (d) color-corrected image, and (e) tone / scale-rendered image.

compensate for dark current noise which is introduced into the signal through thermally generated electrons in the sensor substrate. The rest of the pipeline consists of image processing steps which critically influence the visual quality of final digital photographs.

Figure 1.5 shows the images produced by typical components of the imaging pipeline. Namely, the *CFA mosaic image* is shown in Figure 1.5a. Figure 1.5b shows the full-color image restored by *demosaicking* CFA mosaic data — the step which distinguishes the single-sensor camera image processing pipeline from other camera pipelines. Figure 1.5c shows the image after *white balancing* which is the process of adjusting the image values

to compensate for the scene illuminant, recovering the true scene coloration [27]. Since the spectral sensitivities of the camera are not identical to the human color matching function [26], the pipeline uses *color correction* to adjust the values of color pixels from those corresponding to accurate scene reproduction to those corresponding to visually pleasing scene reproduction. Figure 1.5d shows a color-corrected image. Finally, Figure 1.5e shows the image achieved after subsequent *tone / scale rendering* which transforms the color image from unrendered spaces with twelve to sixteen-bit representations to a rendered (mostly sRGB [28]) space with eight-bit representation, as is required by most output media. This step also makes the tonality of a finished image match the nonlinear characteristics of the human visual system. Additional information on white balancing can be found in Chapter 10, and on color correction and tone / scale rendering in Chapter 3.

The purpose of Figure 1.5 is to illustrate the basic camera image processing chain. The processing steps included in the chain are essential for producing visually pleasing images and can be found in practically all digital cameras. Depending on available resources and camera manufacturer preferences, the processing chain can be extended by adding a series of image quality enhancement steps. As shown in Figure 1.4, visual quality of the demosaicked image can be improved through its *postprocessing*. In order to enhance structural content, such as edges and color transitions, a sharpening step [29] should be included whereas noise and insignificant details can be suppressed or removed via *denoising* or *low-pass filtering* [30], [31]. As visually pleasing appearances of photographs also depend on proper exposure settings, there is also a need for the inclusion of *exposure correction* in the pipeline. Chapter 12 describes popular exposure correction solutions for digital cameras.

It should be mentioned here that both professional photographers and advanced digital SLR camera users prefer no sharpening and denoising in order to preserve the natural appearance of photographs. In addition, professional photographers tend to control white balancing and exposure correction manually in order to achieve desired visual effects. On the other hand, slim compact digital cameras, image-enabled mobile phones and personal digital assistants (PDAs) have to rely on denoising due to the poorer noise characteristics of miniature sensors compared to large-size sensors in SLR cameras. Functionalities of the camera image processing pipeline are often further enhanced by adding optional steps such as *image resizing* [32], [33] to alter the spatial resolution of captured images.

1.3.2 Design Alternatives

As already noted, the order of processing steps has great impact on the overall performance of the imaging pipeline. Unfortunately, there is no ideal way of pipelining the processing steps because the choice of a solution for each particular image processing step has a great impact on both *design* and *performance* characteristics of the imaging pipeline. To simplify the problem of designing the single-sensor camera imaging pipeline, the position of processing steps under consideration can be related to the position of demosaicking. Practically any processing step can be used *before* or *after* demosaicking. Performing steps before demosaicking can allow for significant computational savings due to the grayscale nature of CFA image data, as opposed to performing the same operation on demosaicked color data which basically increases the number of calculations three-fold.

(a) (b) (c)

FIGURE 1.6 (See color insert.)

Image sharpening. Finished images generated by the pipeline depicted in Figure 1.5: (a) original pipeline, i.e., *no sharpening*, (b) original pipeline with added *image sharpening after demosaicking*, and (c) original pipeline with added *image sharpening before demosaicking*.

Figure 1.6 shows the influence of image sharpening on the visual quality of digital camera images. Figure 1.6a is a cropped area from the image in Figure 1.5e. Inspection of the results shown in Figure 1.6a and Figure 1.6b reveals that enhancing the pipeline by adding an image sharpening step after demosaicking increases sharpness of the camera image while amplifying color noise and artifacts present in the image after demosaicking. On the other hand, as can be seen in Figure 1.6c, performing the same image sharpening operations before demosaicking can produce even more significant sharpening effects than the previous approach. In this case, however, the sharpening step may amplify sensor noise present in the captured CFA image data.

When exploring TIFF-EP and EXIF storage formats, a typical example of the processing task considered to be placed before or after demosaicking in the single-sensor imaging pipeline is image compression. Refer to Chapters 3, 14, and 15 for details on compression schemes. Also, image resizing, denoising, sharpening, white balancing, and tone-scale rendering are implementable in either case. Some processing operations can also be implemented in a *joint* manner with demosaicking, thus potentially reducing cost of implementation, enhancing performance of processing tasks, and producing higher visual quality. Usually, implementing various processing steps in a joint process is possible if they employ similar digital signal processing concepts. A good example of that is demosaicking and image resizing which both basically perform interpolation [32], [34], [35]. Another example of the joint process can be constructed by treating demosaicking and denoising from the signal estimation perspective [30]. Details on these two joint processes and performance comparisons between joint implementations and the corresponding traditional cascades can be found in Chapters 9 and 17. Table 1.1 lists other demosaicking-based processing configurations implementable in today's imaging pipelines.

It should be noted at this point that some integration can also be done apart from demosaicking. Examples include simultaneous white balancing and color correction, and tasks based on the filtering concept, such as simultaneous image smoothing and sharpening [31], [36] or simultaneous image resizing and edge enhancement [37], both achieved by combining filters with *low-pass* and *high-pass* characteristics. Integrating and performing different processing steps in the camera imaging pipeline jointly is interesting, in particular from the design point of view. However, depending on the nature of the processing steps to be inte-

TABLE 1.1
Feasibility of some interesting processing configurations for today's imaging pipelines.

processing task	*before demosaicking*	*after demosaicking*	*joint with demosaicking*
image compression	✓	✓	
gamma correction	✓	✓	
white balancing	✓	✓	✓
denoising	✓	✓	✓
image sharpening	✓	✓	✓
image resizing	✓	✓	✓
superresolution reconstruction	✓	✓	✓
digital stabilization		✓	
red-eye detection / removal		✓	
face detection	✓	✓	
face recognition		✓	
digital rights management	✓	✓	✓
image encryption	✓	✓	
forensics	✓	✓	

grated this may be impractical due to the increased memory requirements and complexity of the design, as well as reduced flexibility and high number of calculations to be repeated if some of the steps have to be rerun with different settings.

Some designs can flexibly accommodate various image, video and multimedia processing operations which are occasionally used or may be required in the near future as default components of the imaging pipeline. An example of the former is red-eye removal [38], [39] which aims at detecting and correcting defects in photographs caused by the reflection of the blood vessels in the retina due to strong and sudden light, and *video stabilization* needed to compensate for undesired camera movements [40], [41]. For details refer to Chapters 16 and 20, respectively. The latter includes emerging *CFA video compression* used to store captured video data by reducing spectral and spatiotemporal redundancies present in single-sensor captured image sequences [42], *video-demosaicking* which restores full-color video from its CFA sampled version [43], [44], [45], and *resolution enhancement* which aims at producing images or frames of higher resolution compared to the input [44], [46]. Detailed discussions on these topics can be found in Chapters 15, 18, and 19, respectively. In terms of *multimedia*, many digital cameras support *audio*, either sole or in conjunction with video. Image-enabled consumer electronic devices that greatly benefit from audio and video support are mobile video phones and PDAs. It is quite common to employ in-camera *face detection* [47], [48] in order to improve auto-focusing and to optimize exposure and flash output. Face detection can also help to improve performance of automatic red-eye removal and to produce visually pleasing photographs with enhanced color and tonal quality by setting the optimal white balance and color correction. In addition to face detection, *face recognition* [49], [50] is used in digital photo-archives to organize and retrieve photos. Finally, as in other areas dealing with digital media, there are already certain needs in digital camera imaging for *digital rights management* (DRM) [51], [52], [53], *encryption* [54], [55], and *forensics* [56], [57] in order to ensure digital photograph integrity, secure transmission of photos in public communication networks and protect intellectual property rights.

1.4 Visual Artifacts in Digital Camera Images

Digital images are often corrupted by *noise* and various *artifacts* which significantly degrade the value of captured visual information, decrease the perceptual fidelity of an image and complicate many image processing and analysis tasks. In practice, relations between sources of these defects are very complex. Reducing or eliminating one type of defect can have significant implications on the appearance of another type of defect.

The following focuses on common types of visual impairments present in single-sensor captured images. Namely, issues to be discussed include image noise, demosaicking artifacts, coloration and exposure shifts, and compression artifacts. Discussions of other, mostly optics-based types of defects, such as spherical and chromatic aberrations, vignetting and flare effects, can be found elsewhere (e.g., Reference [58]).

1.4.1 Image Noise

Noise in digital camera images usually appears as random speckles in otherwise smooth regions, altering both tone and color of the original pixels. Typically, noise is caused by random sources associated with quantum signal detection, signal independent fluctuations, and inhomogeneity of the responsiveness of the sensor elements. The appearance of noise in images varies amongst different digital camera models. Noise increases with the *sensitivity* (ISO) setting in the camera, length of the *exposure*, and *temperature*. It can vary within an individual image; darker regions usually suffer more from noise than brighter regions. The level of noise also depends on characteristics of the camera electronics and the physical size of photosites in the sensor. Larger photosites usually have better light-gathering abilities, thus producing a stronger signal and higher signal-to-noise ratio. A survey of these issues, supported by numerous examples, can be found in Reference [59].

According to the *tristimulus theory* of color representation, the RGB color vector $\mathbf{x}_{(r,s)} = [x_{(r,s)1}, x_{(r,s)2}, x_{(r,s)3}]^T$ is uniquely defined by its length (magnitude) $M_{\mathbf{x}_{(r,s)}} = \|\mathbf{x}_{(r,s)}\| = (x^2_{(r,s)1} + x^2_{(r,s)2} + x^2_{(r,s)3})^{-1/2}$ and orientation (direction) $D_{\mathbf{x}_{(r,s)}} = \mathbf{x}_{(r,s)} / \|\mathbf{x}_{(r,s)}\| = \mathbf{x}_{(r,s)} / M_{\mathbf{x}_{(r,s)}}$ in a three-dimensional vector space, where $\|D_{\mathbf{x}_{(r,s)}}\| = 1$ denotes the unit sphere. Thus, both *direction* and *magnitude* of $\mathbf{x}_{(r,s)}$ significantly influence perception of its color by the human observer and both are affected by noise [13]. Since noisy pixels deviate from their noise-free neighbors, the evaluation of magnitude and directional differences between vectors in a local image area constitutes the basis in a number of *noise filtering* techniques. Reference [31] surveys popular filtering approaches in detail.

It is well known that magnitude and direction of a color vector correspond to its *luminance* (intensity) and *chrominance* (color) characteristics. Thus, noise can be seen as fluctuations in intensity and color, and can be handled separately in the luminance and chrominance domain. The relative amount of color and luminance noise differs significantly amongst digital camera models. As shown in Figure 1.7, *color noise* can be completely eliminated; however, suppressing *luminance noise* can result in unnatural looking images and excessive blur. For additional discussions on noise suppression in digital camera images refer to Chapters 3 and 9.

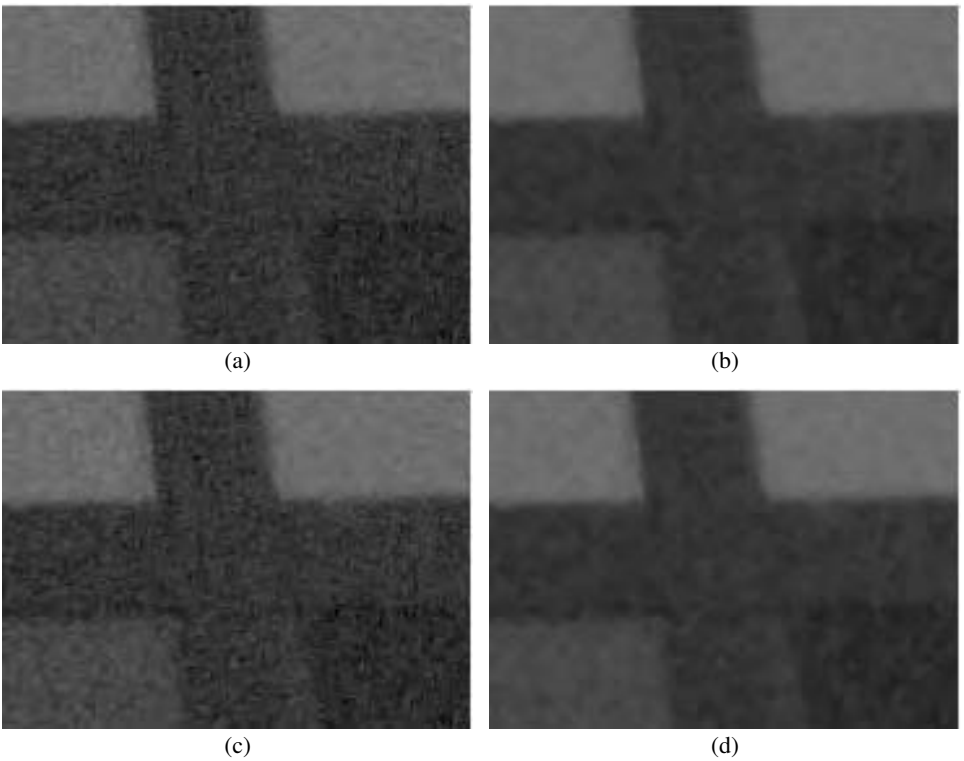


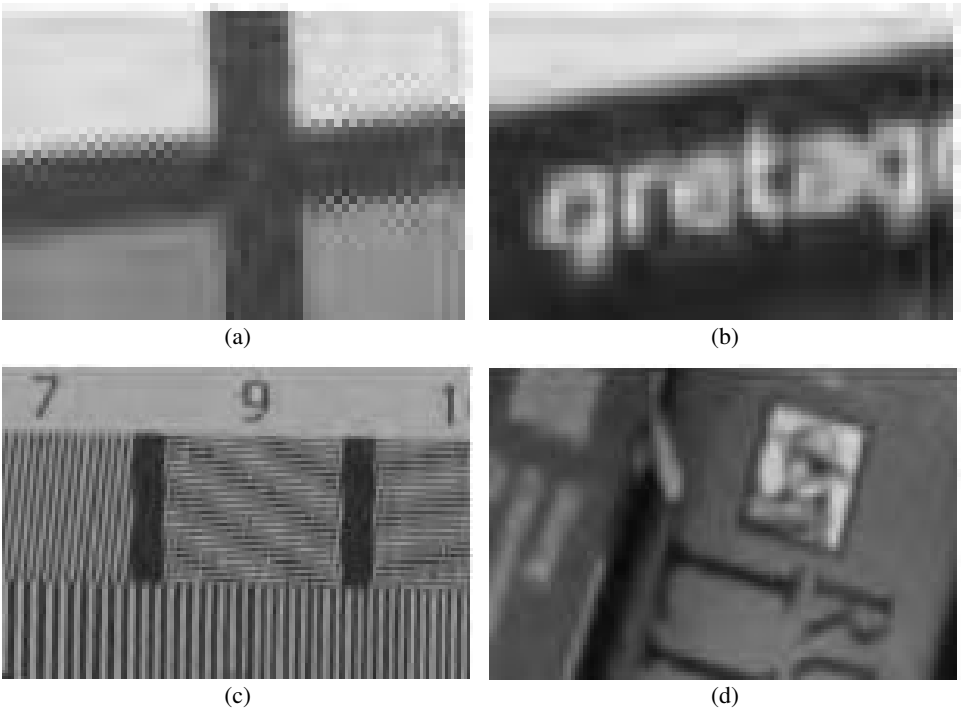
FIGURE 1.7 (See color insert.)

Cropped parts of a color checker image captured with ISO 1600 setting: (a) captured noisy image, (b) luminance noise suppression, (c) color noise suppression, and (d) both luminance and color noise suppression.

1.4.2 Demosaicking Artifacts

Demosaicking is the core of image processing in single-sensor digital color cameras. Since its goal is to restore both color and structural content of an image from mosaic sensor data, the quality of demosaicking significantly influences the amount of detail and artifacts in finished digital photographs.

Figure 1.8 shows several examples of typical issues in demosaicking. Namely, Figure 1.8a shows an image which suffers from *zipper effects*. These effects can usually be seen along abrupt edges when pixels from both side of an edge are used in demosaicking. Figure 1.8b shows an image with isolated color artifacts which are often introduced in regions rich of details due to the lack of spectral and structural information during demosaicking. Due to their localized nature, both defects described above are not as apparent as other impairments when printing or displaying an image at its natural resolution. As can be seen in Figure 1.8c, this is not the case of *aliasing* artifacts or *color moiré* patterns which usually constitute large, visually annoying regions. These artifacts cannot be therefore removed using traditional low-pass filters which rely on local image characteristics. Aliasing artifacts appear in areas where the resolution limit of the sensor has been reached and where color sampling prevents correctly detecting orientations of edges in an image. This is particularly true in fine texture regions where aliasing artifacts often take the form



(a) (b) (c) (d)

FIGURE 1.8 (See color insert.)

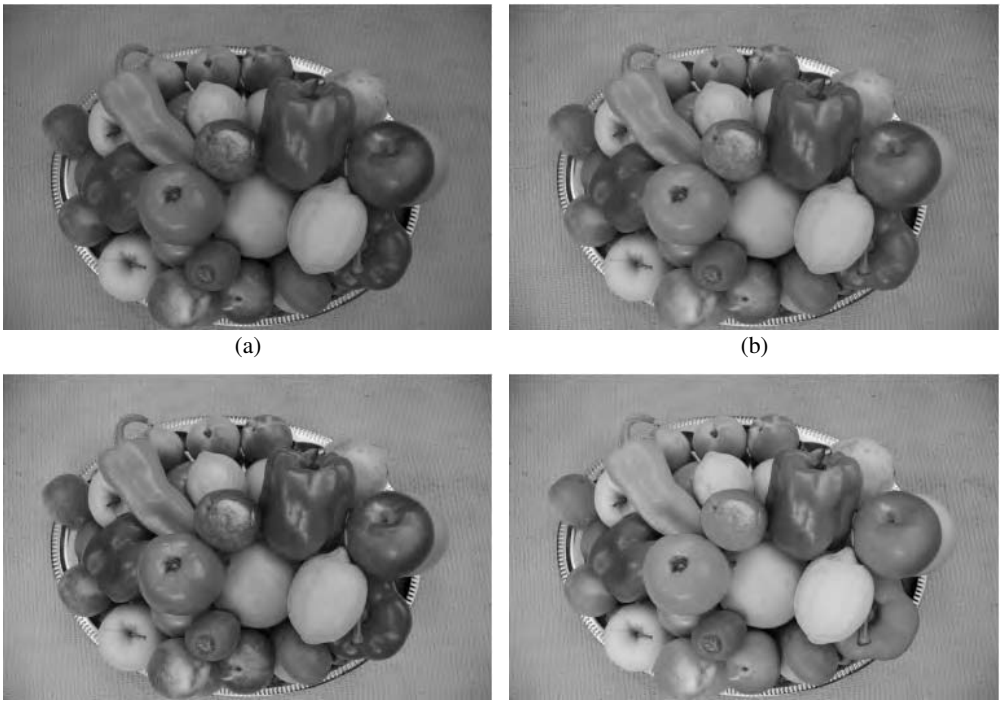
Typical demosaicking defects: (a) zipper effects, (b) color shifts, (c) aliasing artifacts, and (d) blur effects.

of repeating patterns of false colors. Finally, Figure 1.8d shows an image with apparent resolution loss due to excessive *blur* of its structural content during demosaicking with insufficient edge-preserving characteristics.

As discussed above, demosaicking artifacts vary in their characteristics, appearance and size. Considering the complexity of the problem of reversing color sampling in areas with difficult structural content, demosaicking artifacts may never be fully avoided in real-life situations. Therefore, many digital camera designers focus on achieving trade-offs between noise, image sharpness, demosaicking artifacts and processing time rather than emphasizing any of these issues.

1.4.3 Coloration Shifts

Image sensors are calibrated for certain light characteristics. Whenever an image is shot under light of a different color temperature from those for which sensors were calibrated, the image coloration is shifted from the perceived coloration of a scene. This is well observable in the case of neutral (i.e., achromatic) colors, particularly white, which is one of the most recognizable colors due to high and approximately equal contributions of all three color primaries. Unlike noise and demosaicking artifacts which have a localized nature, setting an incorrect white balance affects the appearance of the whole image. Therefore, white balance is considered by many as the most important characteristic of captured images.



(a) (b) (c) (d)

FIGURE 1.9 (See color insert.)

Coloration shifts due to incorrect white balance settings: (a) cool appearance, (b) warm appearance, (c) grayish appearance, (d) saturation effects. See also Figure 1.5e corresponding to an *as-shot* white balance setting.

To produce photographs with a natural *color tint*, an image processing operation called white balancing is performed on captured images. Digital SLR camera users usually set the white balance parameters in their camera based on the shooting situation. Alternatively, some adjustments to color balance can also be made by using software for processing camera images stored in a raw format. Detailed discussions on popular white-balancing approaches can be found in Chapter 10.

Figure 1.9 shows examples of incorrect white balancing. As demonstrated, images often appear bluish (Figure 1.9a) or reddish (Figure 1.9b) which are often referred to as *cool* or *warm*, respectively. Some algorithms produce images with a grayish appearance (Figure 1.9c) or with saturated colors (Figure 1.9d). Obviously, such images are not as visually pleasing as images obtained using an *as-shot* white balance setting (Figure 1.5e).

1.4.4 Exposure Shifts

An *exposure* setting is another important characteristic which affects the global appearance of captured images. Through opening and closing the aperture, the camera basically controls the amount of light reaching the sensor. By deciding how long to leave the shutter open, it controls the period for which the sensor is exposed to the light to collect photons. Finally, adjusting the ISO also has an effect on the exposure. Depending on exposure settings, the appearance of images can range from dark, which is the effect known as *un-*

(a) (b) (c)

FIGURE 1.10 (See color insert.)

Influence of exposure settings on image quality: (a) underexposure, (b) normal exposure, and (c) overexposure.

derexposure, to bright, which is referred to as *overexposure*. Figure 1.10 shows images of the same scene captured with different exposure settings.

To ensure proper exposure, digital cameras use various compensation methods which first measure the amount of light reaching the image sensor and then adjust the exposure accordingly. These methods, however, often fail in complex scenarios with different subjects having different reflectivity. Therefore, cameras and PC software also allow manual control of the exposure by simply multiplying and dividing sensor readings by factors of two. Refer to Chapter 12 for additional information on exposure compensation methods.

1.4.5 Image Compression Artifacts

Finished digital camera images are commonly stored in EXIF format using JPEG compression to reduce their file size and allow more pictures to fit on a memory card. Depending on compression settings, JPEG can reduce the size of original files ten-fold, and for images with solid color backgrounds even more. Due to lossy coding, JPEG-compressed images typically have a blocky appearance which is often referred to as image compression artifacts. These artifacts (and basically also compression abilities of JPEG) result from casting away neighboring pixels with similar luminance and chrominance components in a manner which prevents recovering their original values. Since JPEG and other lossy compression formats ruin fine details and edges, compression artifacts are considered by many a bigger problem than sensor noise.

Fortunately, many of today's digital cameras allow for storing captured images in a raw format following the TIFF-EP standard. Some raw formats do not use compression at all whereas others apply lossy compression to CFA image data, very often using quantization and filtering which results in loss of the resolution. However, most raw formats rely on lossless compression to reduce the size of the files without affecting image quality.

Figure 1.11 demonstrates the effect of compression on the image quality. As shown in Figure 1.11a, compressing full-color data using JPEG produces block artifacts and reduces original structural content. It also suppresses potential demosaicking artifacts due to the low-pass nature of lossy compression. Applying the same compression scheme to CFA mosaic data using a structure conversion [60], [61] results in less blocky appearance of the final image shown in Figure 1.11b while it may be accompanied with higher level of noise and demosaicking artifacts due to performing demosaicking after lossy compression. Finally, Figure 1.11c shows that compression artifacts can be avoided by using lossless coding. Chapters 14 and 15 discuss image compression issues in detail.

(a) (b) (c)

FIGURE 1.11 (See color insert.)

Influence of data compression on image quality for the same demosaicking method: (a) lossy compression of a full-color image, (b) lossy compression of a CFA image, and (c) lossless compression.

1.5 What Is Really Important in Digital Camera Image Processing?

In order to prevent the introduction of various artifacts to finished digital camera images, well-designed processing solutions should be able to follow *image characteristics* as much as possible. Focusing on still digital photography, this relates to the utilization of spatial, structural, and spectral characteristics. In digital video capture, additional temporal characteristics should be considered.

Obviously, the need for *spatial* characteristics results from the fact that spatially neighboring pixels are usually highly correlated. *Structural* characteristics should be followed because edges and fine details convey essential information about a scene. Using *spectral* characteristics is essential since a typical natural image exhibits significant correlation among its R, G, and B color planes. Finally, *temporal* characteristics are good indicators of scene changes and object motion in digital video.

To support the above discussion, Figure 1.12 illustrates the importance of using structural and spectral characteristics during demosaicking. As can be seen from this example, the omission of any of these characteristics usually results in excessive blur, color shifts, and aliasing effects.

1.5.1 Spatial Characteristics

Natural images are nonstationary due to noise and blur processes encountering the image formation. The presence of edges and fine details results in additional variations between neighboring regions. To reduce the processing errors, many camera image processing solutions (e.g., demosaicking, image resizing, noise filtering, edge sharpening, etc.) operate in small localized image areas, each of which can be treated as stationary. Such small areas are localized by placing the supporting window [13], usually centered in the pixel location (r,s) under consideration. The window, defined as the set ζ of pixel locations (i,j) in a

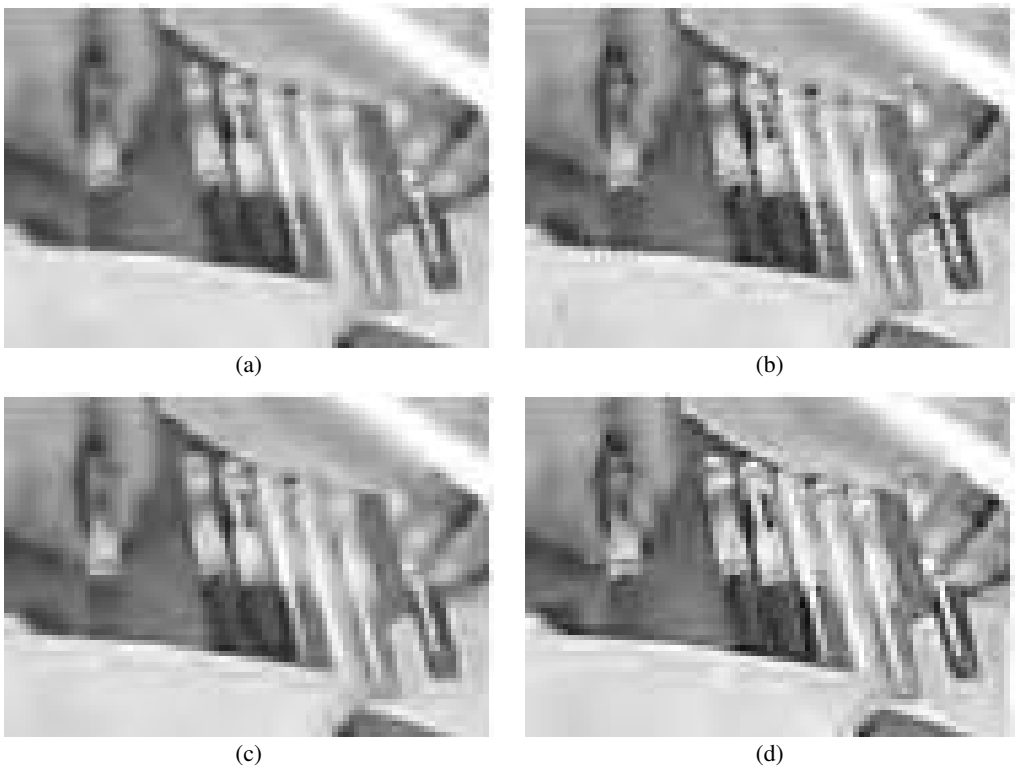
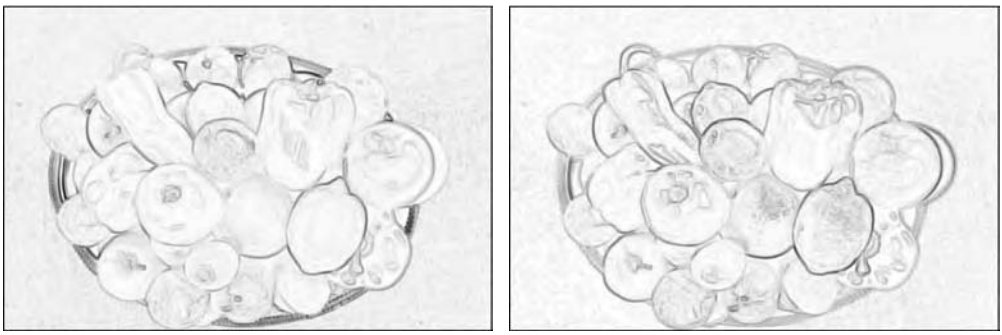


FIGURE 1.12 (See color insert.)

Different visual quality of images demosaicked using: (a) spatial characteristics, (b) spatial and spectral characteristics, (c) spatial and structural characteristics, and (d) spatial, structural, and spectral characteristics.

local neighborhood of (r,s), slides over the entire image placing, successively, every target pixel at the center of a local neighborhood denoted by ζ. The procedure generates the output of the processing operation as a function $f(\cdot)$ — defined as $f(z_{(i,j)};(i,j)\in\zeta)$ for CFA image or $f(\mathbf{x}_{(i,j)};(i,j)\in\zeta)$ for full-color data of samples identified by ζ. In this way, processing solutions can minimize local distortion in the output image.

The performance of a processing solution based on the *windowing* concept is generally influenced by the size of the local area and the actual number of samples in it used as input of the processing solution. Note that these two often refer to two different things. Namely, the former can be seen as an indicator of memory requirements, as it suggests how many image lines (or columns) should be buffered. The latter indicates the processing speed, as it denotes how many normalized operations (e.g., additions, multiplications, etc.), as dictated by function $f(\cdot)$, have to be performed on pixels localized by ζ. For example, it is common in demosaicking to use $\zeta=\{(r-2,s),(r-1,s),(r,s-2),(r,s-1),(r,s),(r,s+1),(r,s+1),(r+1,s),(r+2,s)$. Implementing such demosaicking may require buffering five image lines and in these lines operate in five columns to read all necessary values. In the literature, such demosaicking is often referred as 5×5 demosaicking, although in fact only nine of twenty-five available samples are used in $f(\cdot)$.



(a) (b)

FIGURE 1.13 (See color insert.)

Correlation characteristics of the image shown in Figure 1.5e. Brighter values correspond to higher correlations inside the supporting window. (a) RGB values denote *spatial correlations* in the red, green and blue channel of the original image, respectively. (b) RGB values denote *spectral correlations* between red and green, green and blue, and red and blue channels of the original image, respectively.

Pixels inside the supporting window usually exhibit significant *spatial correlation*, meaning that their values are very similar. As shown in Figure 1.13a, this is truly the case of flat and slowly changing regions. On the other hand, in areas rich of edges and fine details, spatial correlations are usually weaker. Thus, spatial characteristics play in an important role in many image processing operations, such as image compression, demosaicking, filtering and resizing.

1.5.2 Structural Characteristics

Edges and *flat regions* constitute the basic structural content of a digital image. Edges can be seen as discontinuities in the vector space representing the color image. Edges split image regions of different color or intensity; thus, they are essential for human perception. It is therefore important to preserve edges and fine details during image processing.

Popular edge operators process information contained in a local image neighborhood, which is determined by supporting window ζ. Fast and robust *edge detection* can be achieved with using image-agnostic edge operators able to perform with no prior information about the image structure. Edge detectors for color images can be divided into two basic classes: *scalar operators* which process each channel of a color image separately or require color-to-luminance conversions before edge detection, and *vector operators* which fully utilize the spectral correlation and process pixels in a color image as vectors. Both approaches are surveyed in References [31] and [62].

The trade-off between computational complexity and good detection performance makes scalar edge operators ideal candidates to support various camera image processing steps. Typical steps which rely on some form of edge detection are demosaicking, image resizing, denoising, and image sharpening. Edge detection may also help in image compression and white balancing if in these steps more localized processing is needed. Scalar edge detectors use the concept of *gradients* — the first-order directional derivatives of the image — to determine the edge contrast used in edge map formation, or the concept of *zero-crossing*

— the second-order directional derivatives — to identify locations with zero crossings. Through gradients of the image function $U_{(r,s)}$:

$$\nabla U_{(r,s)} = \left[\frac{\partial U_{(r,s)}}{\partial r}, \frac{\partial U_{(r,s)}}{\partial s} \right] \tag{1.1}$$

the first derivative uses the *gradient magnitude*

$$\left| \nabla U_{(r,s)} \right| = \sqrt{\left(\frac{\partial U_{(r,s)}}{\partial r} \right)^2 + \left(\frac{\partial U_{(r,s)}}{\partial s} \right)^2} \tag{1.2}$$

to provide information on the rate of change of image intensity and the *gradient direction*

$$\theta_{(r,s)} = \arctan \left(\frac{\partial U_{(r,s)}}{\partial r} / \frac{\partial U_{(r,s)}}{\partial s} \right) \tag{1.3}$$

to determine the orientation of an edge. Since the second derivative is zero when the first derivative achieves a maximum, it is also possible to localize edges by evaluating the zeros of the second derivatives of $U_{(r,s)}$. One possible implementation of second-order derivatives is given by

$$\Delta U(r,s) = \nabla^2 U(r,s) = \frac{\partial^2 U(r,s)}{\partial r^2} + \frac{\partial^2 U(r,s)}{\partial s^2}. \tag{1.4}$$

With respect to single-sensor imaging, edge-detection can be performed on mosaic CFA data or full-color data. In either case, popular edge operators can be implemented as follows:

$$m_{(r,s)} = \mathbf{w} * U_{(r,s)} = \sum_{(i,j)\in\zeta} \{ w_{(i,j)} u_{(i,j)} \} \tag{1.5}$$

where $m_{(r,s)}$ is the detector response, $u_{(\cdot,\cdot)}$ is image data, and $w_{(\cdot,\cdot)}$ denotes the coefficient of the *convolution mask* $\mathbf{w}$ used to approximate the first- or second-order directional derivative edge operator. Each coefficient $w_{(i,j)}$ is associated with spatial location (i,j). The term $U_{(r,s)} = \{u_{(i,j)}; (i,j) \in \zeta\}$ denotes the set of (CFA or full-color) pixels used as input by performing detection in the pixel location (r,s) under consideration.

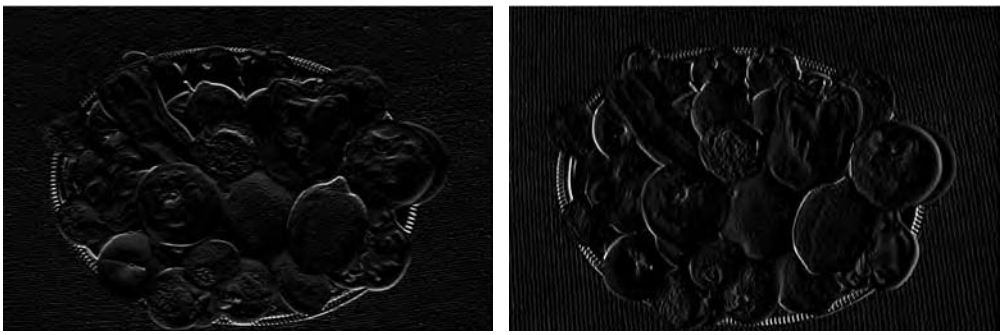
Typically, for the first-order directional derivative operators, masks are defined enabling the determination of the gradient magnitude in each of two orthogonal directions. For example, the convolution masks for the well-known *Sobel operator* are defined as follows:

$$\mathbf{w} = \left\{ \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \right\}, \text{ or } \mathbf{w} = \left\{ \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}, \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \right\} \tag{1.6}$$

The first pair of convolution masks allows for edge detection in horizontal and vertical directions, whereas diagonal edges can be located using the second pair.

A similar approach is to perform edge detection using second-order directional derivative operators. Equation 1.4 represents the well-known *Laplacian operator* which is approximated in practice for a four- and eight-neighborhood, respectively, as follows:

$$\mathbf{w} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \text{ or } \mathbf{w} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{1.7}$$



(a) (b)

FIGURE 1.14 (See color insert.)

Structural content of the image shown in Figure 1.5e: (a) horizontal edges, and (b) vertical edges.

Each of two above convolution masks is separable into a few directional filters with coefficients $[1,-2,1]$. This is an important property, because the determination of edge orientations is essential in various interpolation and filtering steps to avoid processing errors.

Figure 1.14 shows the result of edge detection in horizontal and vertical directions. As can be seen, the directional edge operators differ significantly in their response to structural content in an image. Therefore, a number of image processing algorithms aim at detecting edge orientation first and then performing the processing operation along an edge in order to preserve it. This concept is quite common in camera image processing such as demosaicking, image resizing, and filtering.

1.5.3 Spectral Characteristics

It was already shown in Figure 1.13a that images consist of regions where neighboring pixels are well correlated in a spatial sense. It is also common that small regions in natural images are well correlated in a spectral sense; meaning that the different color channels in these regions exhibit similar dynamics which basically relates to structural content of the image. As shown in Figure 1.13b, significant spectral correlations can typically be observed in slowly changing regions whereas spectral correlations are lower in regions with color edges. A number of camera image processing steps, such as those based on interpolation, filtering, and color manipulation concepts, operate based on the assumption of significant spectral correlation among RGB planes of natural color images, utilizing spectral characteristics of the captured image during its processing.

Chromaticity is one of the most characteristic features of color pixels. It relates to the directional characteristics of the three-component vectors representing color pixels in an RGB space [63], [64]. Thus, it is reasonable to assume that two color vectors $\mathbf{x}_{(r,s)}$ and $\mathbf{x}_{(i,j)}$ occupying spatially neighboring locations (r,s) and (i,j) have the same chromaticity characteristics if they are collinear in the RGB color space [2]. Based on the definition of dot product $\mathbf{x}_{(r,s)}.\mathbf{x}_{(i,j)} = M_{\mathbf{x}_{(r,s)}} M_{\mathbf{x}_{(i,j)}} \cos\left(\langle \mathbf{x}_{(r,s)}, \mathbf{x}_{(i,j)} \rangle\right)$, where $\langle \mathbf{x}_{(r,s)}, \mathbf{x}_{(i,j)} \rangle$ denotes the angle between RGB color vectors $\mathbf{x}_{(r,s)}$ and $\mathbf{x}_{(i,j)}$, the following can be implied:

$$\left\langle \mathbf{x}_{(r,s)}, \mathbf{x}_{(i,j)} \right\rangle = 0 \Leftrightarrow \frac{\sum_{k=1}^{3} x_{(r,s)k} x_{(i,j)k}}{\sqrt{\sum_{k=1}^{3} x_{(r,s)k}^2} \sqrt{\sum_{k=1}^{3} x_{(i,j)k}^2}} = 1 \tag{1.8}$$

The above concept can be further extended by considering both magnitude and directional characteristics of color vectors, as they are both essential for human perception. It was shown in References [65] and [66] that this can be achieved by enforcing the underlying modelling principle of identical color chromaticity on linearly shifted variants of vectors $\mathbf{x}_{(r,s)}$ and $\mathbf{x}_{(i,j)}$:

$$\left\langle \mathbf{x}_{(r,s)} + \gamma\mathbf{I}, \mathbf{x}_{(i,j)} + \gamma\mathbf{I} \right\rangle = 0 \Leftrightarrow \frac{\sum_{k=1}^{3} (x_{(r,s)k} + \gamma)(x_{(i,j)k} + \gamma)}{\sqrt{\sum_{k=1}^{3} (x_{(r,s)k} + \gamma)^2} \sqrt{\sum_{k=1}^{3} (x_{(i,j)k} + \gamma)^2}} = 1 \tag{1.9}$$

where $\mathbf{I} = [1,1,1]^T$ is a unity vector and $x_{(\cdot,\cdot)k} + \gamma$ is the k-th component of the linearly-shifted vector $[\mathbf{x}_{(\cdot,\cdot)} + \gamma\mathbf{I}] = [x_{(\cdot,\cdot)1} + \gamma, x_{(\cdot,\cdot)2} + \gamma, x_{(\cdot,\cdot)3} + \gamma]^T$. The linear-shift γ is a design parameter which controls the influence of directional and magnitude characteristics of color vectors during processing. It can be shown that any component of RGB vector $\mathbf{x}_{(r,s)}$ can be derived from Equation 1.8 or Equation 1.9 based on two other components of $\mathbf{x}_{(r,s)}$ and all three components of $\mathbf{x}_{(i,j)}$ by solving the quadratic equation problem. Using full-color information in calculations makes the solution attractive and potentially highly accurate. Unfortunately, the required number of multiplications may prevent implementation of the solution in current real-time camera imaging pipelines.

To reduce the complexity, the concept behind Equation 1.9 can be applied to two-component vectors which can be created from RGB color vectors by omitting one of their three components. It is straightforward to show that for two-component vectors, each comprising one luminance (G) component and one chrominance (R or B) component, the approach reduces to the following:

$$\frac{x_{(r,s)k} + \gamma}{x_{(i,j)k} + \gamma} = \frac{x_{(r,s)2} + \gamma}{x_{(i,j)2} + \gamma} \tag{1.10}$$

where $k = 1$ refers to R components whereas $k = 3$ refers to B components. This constitutes the *normalized color-ratio model* [67].

Setting a design parameter to $\gamma = 0$ and $\gamma \to \infty$, respectively, allows for further simplification of the above expression. Namely, for $\gamma = 0$, it reduces to the *color-ratio model* of Reference [68]:

$$\frac{x_{(r,s)k}}{x_{(i,j)k}} = \frac{x_{(r,s)2}}{x_{(i,j)2}} \tag{1.11}$$

whereas for $\gamma \to \infty$, the *color-difference model* of Reference [69] is approximated:

$$x_{(p,q)k} - x_{(i,j)k} = x_{(p,q)2} - x_{(i,j)2} \tag{1.12}$$

Both Equation 1.11 and Equation 1.12 constitute early, yet still popular spectral modelling approaches due to their relatively good performance and high computational efficiency, which is particularly true for Equation 1.12.

Spectral models presented in this section can be used in a number of color image processing operations. The most typical ones are those based on interpolation and filtering

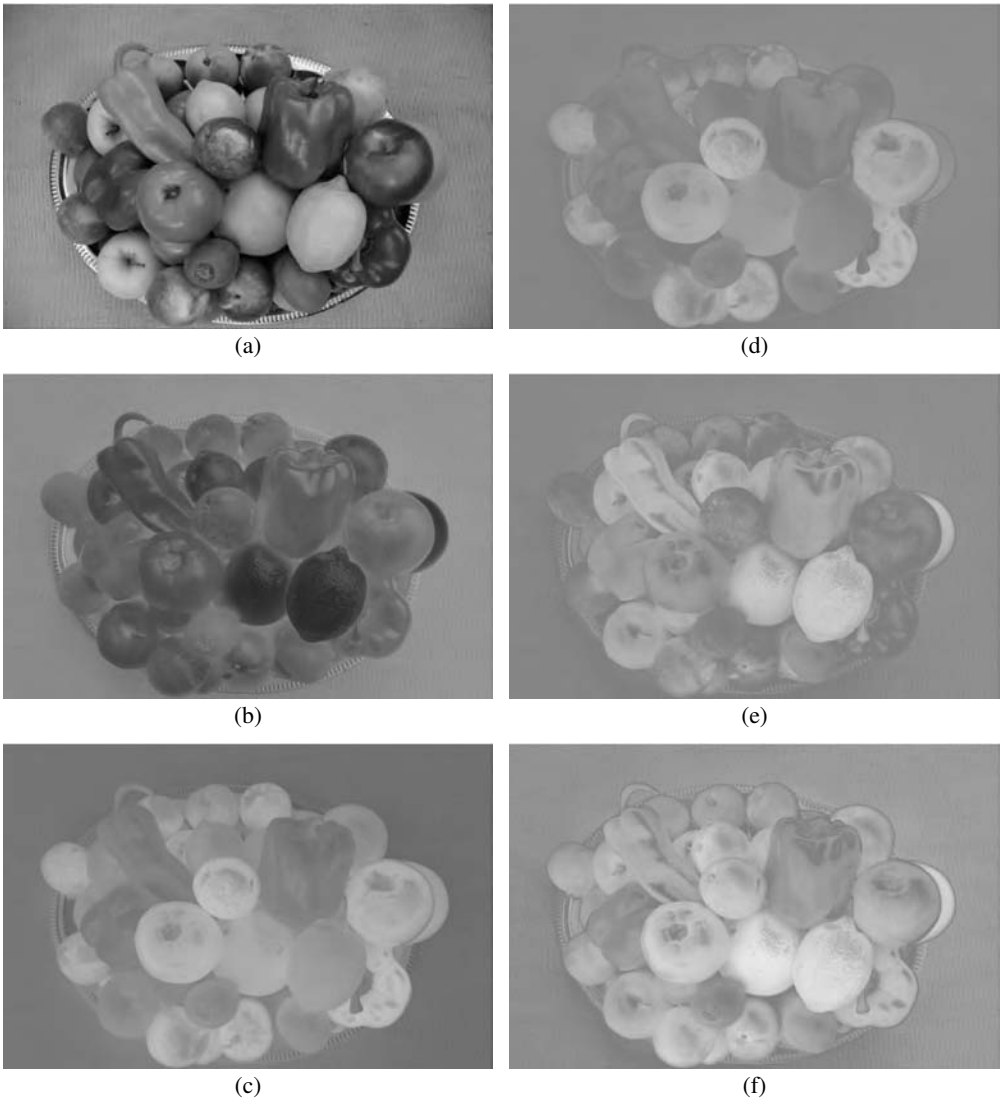


FIGURE 1.15

Typical color-based approaches in digital camera image processing: (a-c) luminance-chrominance approach and (d-f) color-difference approach. These approaches were applied to a color image shown in Figure 1.5e to obtain: (a) luminance image, (b,c) two chrominance images, (d) green-red color difference image, (e) green-blue color difference image, and (f) red-blue color difference image. Middle intensities in images of (b) to (f) correspond to zero values of displayed signals; low and high intensities correspond to negative and positive signal values, respectively.

concepts; however, spectral modelling also helps in white balancing and image compression. In general, any spectral modelling driven processing operation can be defined as follows:

$$\mathbf{x}_{(r,s)} = \Lambda^{-1}\left(\mathbf{x}_{(r,s)}, f(\Lambda(\mathbf{x}_{(i,j)}); (i,j) \in \zeta)\right) \tag{1.13}$$

where $\Lambda(\cdot)$ and $\Lambda^{-1}(\cdot)$ denote, respectively, spectral modelling and inverse spectral mod-

elling functions and $f(\cdot)$ is the processing operation performed in the spectral domain over the samples in the neighborhood defined by ζ. In many demosaicking solutions, $f(\cdot)$ takes the form of an averaging-like operation performed on color difference quantities achieved via $\Lambda(\mathbf{x}_{(i,j)})$, for $(i,j) \in \zeta$. In this case, $\Lambda(\cdot)$ can perform the subtraction of the chrominance component from the luminance component of $\mathbf{x}_{(i,j)}$, thus implying that $\Lambda^{-1}(\cdot)$ adds the chrominance component of $\mathbf{x}_{(r,s)}$ to the output of $f(\cdot)$ in order to obtain the resulting luminance component of $\mathbf{x}_{(r,s)}$.

Spectral modelling is not the only color-based approach used in digital camera image processing. A number of processing steps, such as demosaicking, filtering and compression, can operate by first transforming the color data into one luminance and two chrominance components, and then performing the processing operation on luminance and chrominance signals separately. The final RGB color image is obtained through the inverse transform of processed luminance and chrominance images. Figure 1.15 allows for comparison of both approaches. As can be seen in Figure 1.15a, the luminance image contains almost complete structural content of the color image shown in Figure 1.5e. On the other hand, as seen in Figure 1.15b to Figure 1.15f, both chrominance and color-difference images lack most details and small edges, suggesting that these signals have a low-pass nature. Processing images with such reduced structural content can mitigate the processing error and allow for higher performance compared to when operating directly on RGB color channels.

1.5.4 Temporal Characteristics

Compared to images, digital video has an additional dimension, as it consists of two-dimensional images or frames captured in a certain period of time. This suggests that processing digital video may require an extension to traditional approaches for still images for more dimensional signals or using special approaches to effectively deal with its unique temporal characteristics. These additional characteristics can be expressed as changes between consecutive frames. Typically, most of these changes are caused by the motion, either of objects in the scene or both camera and objects. Therefore, popular video processing techniques aim at deriving motion information first and then adjusting the processing operation accordingly in order to prevent motion blur and artifacts and to increase performance. To avoid problems found when processing still images, temporal characteristics should be used together with spatial, structural and spectral characteristics, taking advantage of different types of correlations. Detailed discussions on typical video-processing issues in digital cameras can be found in Chapters 18 to 20.

1.6 Conclusion

This chapter aimed at summarizing the fundamentals of single-sensor color imaging and digital camera image processing. The concept of sampling visual information using a color filter array placed on top of a monochrome image sensor became one of the most important developments in the history of digital imaging due to its good performance, effectiveness,

and relatively low cost. Therefore, it is no surprise that this concept plays a key role in popular consumer electronic devices with image-capturing capabilities, such as digital still and video cameras, mobile phones, and personal digital assistants, and that many color imaging applications, for example, digital photography, printing, visual communication, machine vision, digital cinema, medical imaging, and astronomy, benefit from using such devices.

As described in this chapter, the data captured by a color filter array sensor architecture has to undergo a number of image processing steps in order to produce digital photographs. These steps constitute the digital camera image processing pipeline and can be implemented in imaging devices or performed on personal computers, thus flexibly providing the user with a number of options. In either case, each processing step has its own design, performance, and implementation challenges. Most of these challenges relate to effectively using information available in captured visual data in order to produce visually pleasing finished images in the output of the pipeline. Thus, the utilization of the spatial, structural, spectral, and even motion characteristics is essential in modern imaging systems which attempt to mimic human perception of the visual environment. Meeting such objectives requires large efforts in designing the pipeline which achieves the best collaborative effect of its components and is reasonably robust in order to deal with the infinite amount of variations in the visual scene.

References

[1] K. Parulski and K.E. Spaulding, *Digital Color Imaging Handbook*, ch. Color image processing for digital cameras, G. Sharma (ed.), Boca Raton, FL: CRC Press, 2002, pp. 728–757.

[2] R. Lukac and K.N. Plataniotis, *Color Image Processing: Methods and Applications*, ch. Single-sensor camera image processing, R. Lukac and K.N. Plataniotis (eds.), Boca Raton, FL: CRC Press / Taylor & Francis, October 2006, pp. 363–392.

[3] G. Wyszecki and W.S. Stiles, *Color Science, Concepts and Methods, Quantitative Data and Formulas*, 2nd Edition, New York: John Wiley, 1982.

[4] P.L.P. Dillon, D.M. Lewis, and F.G. Kaspar, "Color imaging system using a single CCD area array," *IEEE Journal of Solid-State Circuits*, vol. 13, no. 1, pp. 28–33, February 1978.

[5] B.T. Turko and G.J. Yates, "Low smear CCD camera for high frame rates," *IEEE Transactions on Nuclear Science*, vol. 36, no. 1, pp. 165–169, February 1989.

[6] H.K. Burke and G.J. Michon, "Charge injection imaging: Operating techniques and performances characteristics," *IEEE Journal of Solid-State Circuits*, vol. 11, no. 1, pp. 121–128, February 1976.

[7] C. Anagnostopoulos, "Signal readout in CID image sensors," *IEEE Transactions on Electron Devices*, vol. 25, no. 2, pp. 85–89, February 1978.

[8] A.J. Blanksby and M.J. Loinaz, "Performance analysis of a color CMOS photogate image sensor," *IEEE Transactions on Electron Devices*, vol. 47, no. 1, pp. 55–64, January 2000.

[9] D. Doswald, J. Haflinger, P. Blessing, N. Felber, P. Niederer, and W. Fichtner, "A 30-frames/s megapixel real-time CMOS image processor," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1732–1743, November 2000.

[10] R.F. Lyon and P.M. Hubel, "Eying the camera: Into the next century," in *Proceedings of the Tenth IS&TSID Color Imaging Conference*, Scottsdale, AZ, USA, November 2002, pp. 349–355.

[11] P.M. Hubel, J. Liu, and R.J. Guttosh, "Spatial frequency response of color image sensors: Bayer color filters and Foveon X3," Technical Report ID 6502, Foveon, San Antonio, TX, USA, March 2002.

[12] G. Sharma and H.J. Trussell, "Digital color imaging," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 901–932, July 1997.

[13] R. Lukac, B. Smolka, K. Martin, K. N. Plataniotis, and A. N. Venetsanopulos, "Vector filtering for color imaging," *IEEE Signal Processing Magazine*, Special Issue on Color Image Processing, vol. 22, no. 1, pp. 74–86, January 2005.

[14] J. Adams, K. Parulski, and K. Spaulding, "Color processing in digital cameras," *IEEE Micro*, vol. 18, no. 6, pp. 20–30, November 1998.

[15] B.E. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 1976.

[16] FillFactory, "Technology - image sensor: The color filter array faq." Available online: http://www.fillfactory.com/htm/technology/htm/rgbfaq.htm.

[17] R. Lukac and K.N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1260–1267, November 2005.

[18] B.K. Gunturk, J. Glotzbach, Y. Altunbasak, R.W. Schaffer, and R.M. Murserau, "Demosaicking: Color filter array interpolation," *IEEE Signal Processing Magazine*, Special Issue on Color Image Processing, vol. 22, no. 1, pp. 44–54, January 2005.

[19] R. Lukac and K. N. Plataniotis, "A robust, cost-effective postprocessor for enhancing demosaicked camera images," *Real-Time Imaging*, Special Issue on Spectral Imaging II, vol. 11, no. 2, pp. 139–150, April 2005.

[20] R. Lukac, K. Martin, and K. N. Plataniotis, "Demosaicked image postprocessing using local color ratios," *IEEE Transactions on Circuit and Systems for Video Technology*, vol. 14, no. 6, pp. 914–920, June 2004.

[21] W. Lu and Y.P. Tang, "Color filter array demosaicking: New method and performance measures," *IEEE Transactions on Image Processing*, vol. 12, no. 10, pp. 1194–1210, October 2003.

[22] B. Gunturk, Y. Altunbasak, and R. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 997–1013, September 2002.

[23] Technical Committee ISO/TC 42, Photography, "Electronic still picture imaging - removable memory, part 2: Image data format - TIFF/EP," ISO 12234-2, January 2001.

[24] "Information technology - digital compression and coding of continuous-tone still images: Requirements and guidelines." ISO/IEC International Standard 10918-1, ITU-T Recommendation T.81, 1994.

[25] Japan Electronics and Information Technology Industries Association, "Exchangeable image file format for digital still cameras: Exif Version 2.2," Technical report, JEITA CP-3451, April 2002.

[26] R. Ramanath, W.E. Snyder, Y. Yoo, and M.S. Drew, "Color image processing pipeline," *IEEE Signal Processing Magazine*, Special Issue on Color Image Processing, vol. 22, no. 1, pp. 34–43, January 2005.

[27] R. Lukac, "New framework for automatic white balancing of digital camera images," *Signal Processing*, vol. 88, no. 3, pp. 582–593, March 2008.

[28] International Electrotechnical Commission, *Colour Measurement and Management in Multimedia Systems and Equipment - Part 2-1: Default RGB Colour Space - sRGB*. IEC 61966-2-1, 1999.

[29] R. Lukac and K.N. Plataniotis, "A new image sharpening approach for single-sensor digital cameras," *International Journal of Imaging Systems and Technology*, Special Issue on Applied Color Image Processing, vol. 17, no. 3, pp. 123–131, June 2007.

[30] K. Hirakawa and T.W. Parks, "Joint demosaicing and denoising," *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2146–2157, August 2006.

[31] R. Lukac and K.N. Plataniotis, *Advances in Imaging and Electron Physics*, ch. A taxonomy of color image filtering and enhancement solutions, pp. 187–264, San Diego, CA: Elsevier / Academic Press, February/March 2006.

[32] R. Lukac, K. Martin, and K.N. Plataniotis, "Digital camera zooming based on unified CFA image processing steps," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 15–24, February 2004.

[33] R. Lukac, K. N. Plataniotis, and D. Hatzinakos, "Color image zooming on the Bayer pattern," *IEEE Transactions on Circuit and Systems for Video Technology*, vol. 15, no. 11, pp. 1475–1492, November 2005.

[34] K.H. Chung and Y.H. Chan, "A low-complexity joint color demosaicking and zooming algorithm for digital camera," *IEEE Transactions on Image Processing*, vol. 16, no. 7, pp. 1705–1715, July 2007.

[35] L. Zhang and D. Zhang, "A joint demosaicking-zooming scheme for single chip digital color cameras," *Computer Vision and Image Understanding*, vol. 107, no. 1-2, pp. 14–25, July/August 2007.

[36] R. Lukac, B. Smolka, and K.N. Plataniotis, "Sharpening vector median filters," *Signal Processing*, vol. 87, no. 9, pp. 2085–2099, September 2007.

[37] R. Lukac, "Methods for fast enlargement of digital images," U.S. Patent, submitted, December 2007.

[38] M. Gaubatz and R. Ulichney, "Automatic red-eye detection and correction," in *Proceedings of the IEEE International Conference on Image Processing*, Rochester, NY, USA, September 2002, vol. 1, pp. 804–807.

[39] F. Volken, J. Terrier, and P. Vandewalle, "Automatic red-eye removal based on sclera and skin tone detection," in *Proceedings of the IS&T Third European Conference on Color in Graphics, Imaging and Vision*, Leeds, UK, June 2006, pp. 359–364.

[40] A. Engelsberg and G. Schmidt, "A comparative review of digital image stabilising algorithms for mobile video communications," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 591–597, August 1999.

[41] S.C. Hsu, S.F. Liang, and C.T. Lin, "A robust digital image stabilization technique based on inverse triangle method and background detection," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 335–345, May 2005.

[42] L. Zhang, X. Wu, and P. Bao, "Real-time lossless compression of mosaic video sequences," *Real-Time Imaging*, Special Issue on Multi-Dimensional Image Processing, vol. 11, pp. 370–377, October-December 2005.

[43] R. Lukac and K.N. Plataniotis, "Fast video demosaicking solution for mobile phone imaging applications," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 675–681, May 2005.

[44] S. Farsiu, M. Elad, and P. Milanfar, "Multiframe demosaicing and super-resolution of color images," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 141–159, January 2006.

[45] X. Wu and L. Zhang, "Improvement of color video demosaicking in temporal domain," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 3138–3151, October 2006.

[46] S.G. Narasimhan and S.K. Nayar, "Enhancing resolution along multiple imaging dimensions using assorted pixels," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 27, no. 4, pp. 518–530, April 2005.

[47] P. Viola and M.J. Jones, "Robust real-time object detection," Tech. Rep. CRL 2001/01, Compaq Cambridge Research Laboratory, Cambridge, Massachusetts, February 2001.

[48] R.L. Hsu, M. Abdel-Mottaleb, and A.K. Jain, "Face detection in color images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, May 2002.

[49] S.Z. Li and A.K. Jain (eds.), *Handbook of Face Recognition*. New York: Springer, 2005.

[50] K. Delac and M. Grgic (eds.), *Face Recognition*. Vienna, Austria: I-Tech, 2007.

[51] F. Bartolini, A. Tefas, M. Barni, and I. Pitas, "Image authentication techniques for surveillance applications," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1403–1418, October 2001.

[52] P. Blythe and J. Fridrich, "Secure digital camera," in *Proceedings of the Digital Forensic Research Workshop*, Baltimore, MD, USA, August 2004.

[53] R. Lukac and K.N. Plataniotis, "Secure single-sensor digital camera," *IEE Electronics Letters*, vol. 42, no. 11, pp. 627–629, May 2006.

[54] H. Cheng and X. Li, "Partial encryption of compressed images and videos," *IEEE Transactions on Signal Processing*, vol. 48, no. 8, pp. 2439-2451, August 2000.

[55] K. Martin, R. Lukac and K.N. Plataniotis, "Efficient encryption of wavelet-based coded color images," *Pattern Recognition*, vol. 38, no. 7, pp. 1111–1115, July 2005.

[56] A.C. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images," *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3948–3959, October 2005.

[57] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, June 2006.

[58] P. van Walree, "Photographic optics." Available online: http://www.vanwalree.com/optics.html.

[59] S.T. McHugh, "Digital camera image noise." Available online: http://www.cambridgeincolour. com/tutorials/noise.htm.

[60] C.C. Koh, J. Mukherjee, and S.K. Mitra, "New efficient methods of image compression in digital cameras with color filter array," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1448–1456, November 2003.

[61] R. Lukac and K.N. Plataniotis, "Single-sensor camera image compression," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 299–307, May 2006.

[62] K.N. Plataniotis and A.N. Venetsanopoulos, *Color Image Processing and Applications*. New York: Springer Verlag, 2000.

[63] P.E. Trahanias, D. Karakos, and A.N. Venetsanopoulos, "Directional processing of color images: Theory and experimental results," *IEEE Transactions on Image Processing*, vol. 5, no. 6, pp. 868–881, June 1996.

[64] B. Tang, G. Sapiro, and V. Caselles, "Color image enhancement via chromaticity diffusion," *IEEE Transactions on Image Processing*, vol. 10, no. 5, pp. 701–707, May 2001.

[65] R. Lukac and K.N. Plataniotis, "Vector concepts-based spectral modelling, in *Proceedings of the 19th Annual Canadian Conference on Electrical and Computer Engineering*, Ottawa, ON, Canada, May 2006, pp. 2009–2012.

[66] R. Lukac and K.N. Plataniotis, "Demosaicking using vector spectral model," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, Toronto, ON, Canada, July 2006, pp. 1185–1188.

[67] R. Lukac and K.N. Plataniotis, "Normalized color-ratio modeling for CFA interpolation," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 737–745, May 2004.

[68] D.R. Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal," U.S. Patent 4 642 678, February 1987.

[69] J. Adams, "Design of practical color filter array interpolation algorithms for digital cameras," *Proceedings of SPIE*, vol. 3028, pp. 117–125, February 1997.

2

Reusable Embedded Software Platform for Versatile Single-Sensor Digital Cameras

Wen-Chung Kao, Hung-Hsin Wu, and Sheng-Yuan Lin

2.1 Introduction

Many versatile point-and-shoot digital cameras, which support attractive functions with satisfactory system performance, have been announced in the consumer electronics market [1], [2], [3], [4]. However, camera system designers still suffer from the difficulty of lacking hardware architecture standards and good software design methodology. According to the product definition, camera designers must carefully select several key hardware components such as camera signal processor (CSP) [5], [6], [7], [8], [9],[10], [11], lens module [12], [13], [14], [15], [16], charge-coupled device (CCD) [17], [18], [19] or complementary metal oxide semiconductor (CMOS) image sensor [20], [21], [22], analog front end (AFE) chip [23], [24], and liquid crystal display (LCD). Among these components, CSP plays the most important role for the entire system. A typical CSP consists of an embedded microprocessor (EMP), hardware engines, peripherals, and other programmable computing units such as digital signal processors (DSPs) for real-time image/video processing. The scheduling as well as the allocation for these heterogeneous computational resources is quite a complex issue in embedded software design.

In recent years, the functions demanded for digital cameras have grown very numerous and complex. Development time is forced to decrease and maintaining the entire software system becomes more and more difficult. Many integrated signal processors have been proposed for simplifying the design of digital still cameras (DSCs), but very few systematic software platforms are available to support the development of multi-function DSCs [25], [26]. The lack of a standard software platform defined in industry remains a significant obstacle in developing digital cameras. The embedded software team should construct several software systems for a variety of camera models that are designed with specific CSPs. It is estimated that up to 60% of the development time may be spent in software coding and thus the software design phase has become a bottleneck in developing a DSC.

A modern digital camera is no longer a simple imaging system. Other than some basic functions such as capture, playback, display and storage, a high performance camera supports many attractive functions while satisfying miscellaneous timing constraints and the power budget. Namely, the supported features include digital zoom in both capture and playback mode, continuous shots with instant audio recording, MPEG video/audio compression and real-time data streaming, direct print that incorporates output formatting and color postprocessing, and parsing or preparing configuration files for the captured pictures such that they can be mailed out and printed automatically from a personal computer (PC). Some of the functions listed above tightly depend on the resolutions of the image sensor and display device. Without a good software architecture design, changing part of these devices may result in effects that the current software system might not be able to accommodate.

Apart from camera functionality, performance is also important for a camera system. The speed of capturing an image as well as the response time of human machine interface affects the evaluation of a camera. An advanced consumer DSC supports fast continuous shots, recording audio at the same time, and runs fast auto-exposure (AE) [27], [28], [29],

auto-focus (AF) [30], [31], [32] between successive shots. Such a camera has to handle start and stop of audio recording, immediate capture of the next picture, and data storage while avoiding accidentally recording the sound of the mechanical shutter. On the other hand, the user may want to have fast response from the camera whenever they press buttons. It is particularly critical to have a short delay from pressing the shutter button to actually taking an instant image. The challenge behind this performance index is that many tasks are executed simultaneously, including periodic AE and automatic white balancing (AWB) in preview mode, pre-capture exposure/focus estimation, fine resolution image raw data readout, audio recording, image processing, key scan task, and real-time streaming of compressed data.

A robust embedded software platform, that has good flexibility in accommodating these changeable hardware components and software modules, would be key in producing versatile digital cameras. In addition to advanced functions supported by cameras and the performance requirements, from the viewpoints of system designers, the most important features of an ideal embedded software platform include flexibility, reusability [33], [34], [35], [36], and calibration/diagnosis capability [37]. This chapter describes an integrated embedded software platform for developing consumer DSCs. Although different camera models possess diverse graphic user interface (GUI) or adopt different key hardware components, many functions are common for all cameras, even if they might be implemented with different operations. The platform should be reusable when some of the hardware components are replaced or the image resolution is changed. Furthermore, an ideal software platform must utilize common functions for both normal operation mode and engineering modes that can calibrate camera parameters and diagnose/verify the system specifications. Each camera instance can be calibrated in the production line without connecting to any computers, thus the number of computer based test fixtures can be drastically reduced.

2.2 Hardware Platform Overview

The hardware organization of a typical consumer camera system is shown in Figure 2.1. The image captured from the CCD/CMOS sensor is stored in synchronous dynamic random access memory (SDRAM) first and then processed by the CSP. On the other hand, audio is recorded from the microphone and converted into digital signals through an analog-to-digital converter (ADC). The audio/video data that pass through signal processing as well as compression are finally stored in the internal flash memory or external flash cards. The color LCD provides a friendly GUI for viewing the pictures to be taken in the field. A few DSCs support the functions of printing the pictures directly to printers through USB interface in the host or slave mode. This direct print mode enables connection to some specific printers and makes the print operation simpler without having to clumsily set the print mode.

In the following subsections, several key components of a camera system are introduced: the lens module, the image sensor, and the color image signal processor. Meanwhile, the design considerations of AF and AE algorithms are also discussed.

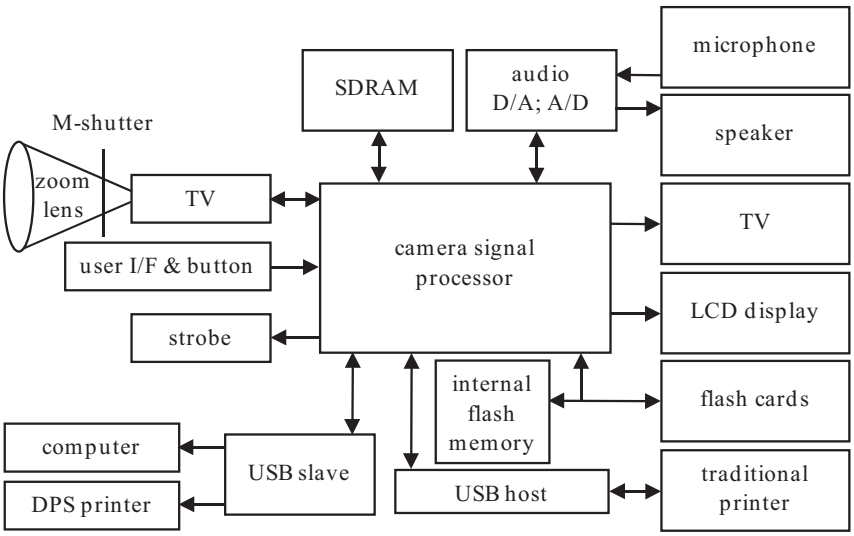


FIGURE 2.1

The hardware organization of a consumer digital camera. © 2005 IEEE

2.2.1 Zoom Lens Module and Auto Focus Control

A zoom lens allows the photographer to change view angles at the same spot. Using zoom lens cameras, the photographer does not need to bring many lenses with different focal length while taking pictures in different view angles. In general, with longer focal length, the view angle is narrower and more image details in a smaller area can be captured clearly. On the other hand, with shorter focal length, a wider scene can be captured but the resolution for the same object becomes lower.

Since every lens has limited depth of field (DOF), the distance in the object space that can be projected clearly on the image plane is limited. In order to capture scenes at different distances, the image sensor position has to be adjusted to make the projected image of the main subject fall accurately on the sensor. This action is called auto focus (AF), and in practice the lens is moved instead of the image sensor because controlling the lens motor is much easier.

2.2.1.1 Overview of Zoom Lens Control

A zoom lens is usually composed of several groups. As the individual groups of the lens move along the optical axis and change the relative distance among them, the effective focal length (EFL) of the lens will be changed. The relation of the zoom position and the corresponding EFL is inherently nonlinear, but lens makers are always able to provide a look-up table (LUT) of EFL versus motor positions for the software designer to use. To simplify the zoom lens control, most manufactures split the range of zoom motor motion into a few steps and the control LUT contains only the data for these steps. In practice, it is possible to design some test patterns to calibrate the EFL values versus the motor positions.

The so-called zoom ratio is actually the ratio of the longest EFL to the shortest EFL. For example, if the range of the EFL for a digital camera is 5.8 - 17.4mm, the zoom ratio

is $17.4/5.8 = 3$. It is then marked as a $3\times$ zoom lens. When the subject is far from the camera, it is approximately true that when the EFL is switched from 5.8mm to 17.4mm, the image width is increased to about three times. Note that changing the EFL of a zoom lens also changes the effective aperture value. Longer EFL results in smaller aperture and larger F-number. The main issue with different F-numbers is that the light intensity falling on the image sensor will be different and the AE algorithm needs to take this change into account while changing zoom steps. For consumer digital still cameras, the AE algorithm relies on the image sensor to sense and analyze the brightness of the scene. This issue is automatically taken care of and thus changing F-number is not a big problem for AE control.

Changing EFL also affects the characteristic of lens fall-off. The problem of lens fall-off means that the light intensity at the corners of a lens is typically lower than the intensity at the center. Lenses with shorter EFLs have obvious fall-off problems. An approach to dealing with this issue is to perform lens fall-off compensation in the image processing pipeline, in which different compensation factors can be applied at different zoom steps.

The relationship between EFL and lens position is usually nonlinear. As the zoom motor is moved, the focus position will be also changed such that the focus needs to be readjusted. However, enabling both zoom motors and focus motors at the same time may cause temporarily very high power consumption. Hence zoom motor and focus motor are not enabled simultaneously in practice, and a fine AF adjustment is executed to achieve a more accurate focus after the zoom motor stops at the target position.

2.2.1.2 Design Considerations of Auto Focus Algorithms

Auto focus is usually achieved by moving the lens to multiple focus positions and taking several test images to evaluate the focus conditions. A better focus condition means that an image keeps more image details. The focus condition is either called focus index (FI) or figure of merit (FOM), which are usually implemented with a two-dimensional high-pass digital filter.

In traditional photography, a small region of interest (ROI) is defined at the center of the image and AF program only evaluates the FOM at this ROI. The photographer must point the camera to the main subject to make correct focus. The camera scans several focus motor positions and the FOM value is calculated for each position to generate a focus curve as shown in Figure 2.2. The peak of the curve is usually regarded as the best focus position.

Several ways have been proposed to speed up the process of finding the best focus position. One is to estimate the slope of the curve and determine the next point of focus trial with a better increment. The coarse search step in AF is according to the DOF of the camera in the current zoom step. When the curve changes direction, it means a local peak has been found and the remaining process is to fine search the peak position in the focus curve. A modified approach is to use a second order polynomial curve fitting to fast predict the peak of the focus curve after coarse search. This approach provides a promising search result in general conditions and has been widely applied in designing commercial cameras. On the other hand, advanced digital still cameras frequently use a larger ROI for AF. The ROI is split into a few smaller regions and the FOM value of each region is calculated individually. Then these FOM values are compared and a best focus is selected.

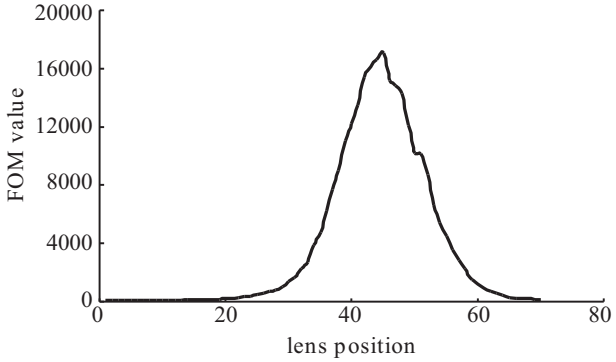


FIGURE 2.2

The figure of merit (FOM) curve in auto focus.

2.2.2 Image Sensor and Auto Exposure

2.2.2.1 Overview of Image Sensors

There are two popular types of commercial image sensors, charge-coupled device (CCD) and complimentary metal-oxide-semiconductor (CMOS) image sensor. In both types of image sensor, the transducer is a photo diode coupled with a capacitor. The difference between them is the way of transferring the image signals to the outside circuit. When light falls on the sensor, some photons are absorbed by the silicon and electron-hole pairs are generated. The sequence of electron-hole pairs forms an electric current, which is in general proportional to the light intensity falling on the photo diode. The charge accumulated in the capacitor is the integration over time of the current flowing into it. Since the current is proportional to image light intensity and the output voltage is proportional to the accumulated charge, it can be deduced that the output voltage of a photo diode is proportional to the product of image light intensity and the integration time. The integration time of a photo diode is controlled by its reset timing. The control circuit periodically sends a reset pulse to clear the charge accumulated in the capacitor. When the effective exposure starts, the control mechanism simply stops the reset signal and the newly generated charge will be accumulated. This exposure control mechanism is also called electronic shutter.

The end of the exposure procedure is achieved by connecting the capacitor to a sense and read amplifier through another transistor. After the voltage is sensed and converted to a digital value, the photo diode capacitor is cleared again and the next exposure procedure can be started. This kind of electronic shutter is very precise and controlling extremely short durations is possible. The shortest exposure time of a traditional mechanical shutter is around 1/2000 second, while electronic shutter can easily achieve 1/50000 second of exposure time resolution.

There are several different architectures of CCD design, including frame transfer, full frame transfer, frame interline transfer, and interline transfer. The most popular architecture adopted in consumer cameras is the interline transfer which is shown in Figure 2.3. The pixels are arranged as a two-dimensional array. Between neighboring vertical lines of the photo diodes, there is a line of vertical CCDs which are masked by a metal layer and will not

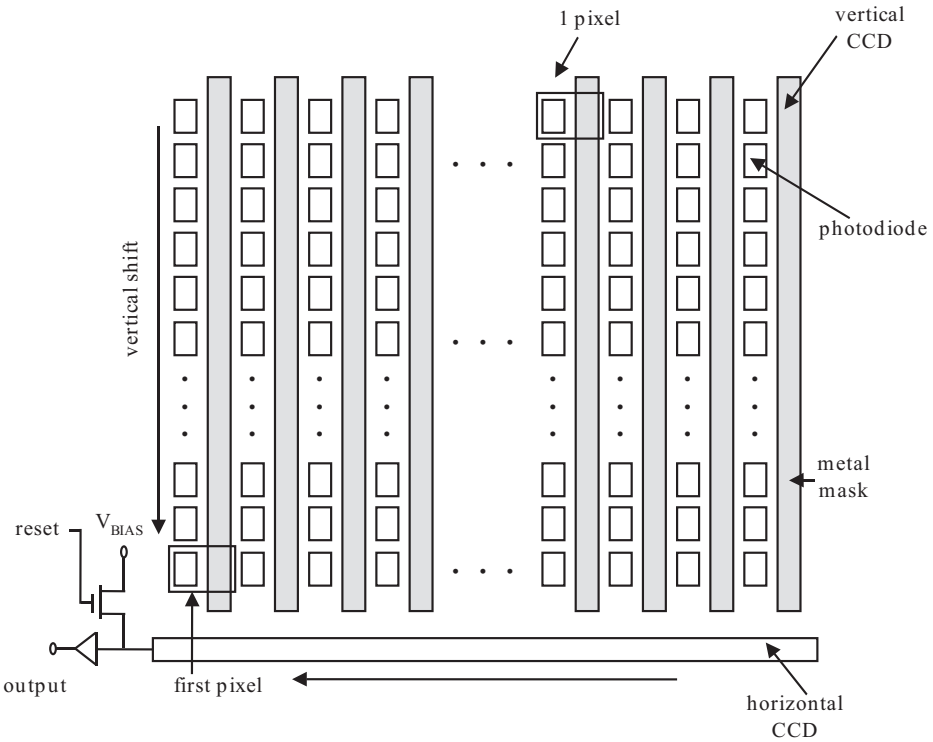


FIGURE 2.3

Basic interline transfer CCD architecture.

respond to incident light. The pixels in the entire frame are exposed and the accumulated charges in the photo diodes are transferred to the neighboring vertical CCDs at the same time. The pixel charges in each horizontal line are then shifted vertically downward to the horizontal CCDs. Finally, the signals stored in the horizontal CCDs are amplified and sent out in pixel rate.

The main difference between CCD and CMOS image sensors is that CMOS sensor lacks charge buffer as the readout mechanism. Once the exposure is completed, the charge is read out immediately. Since there is at most one row of ADCs at the bottom of the pixel array and the data readout is also sequential, only one row of the pixels can be read out at a time. The exposure of the second row must be completed exactly at the instant of the first row readout completed and so on for the remaining rows. In order to keep the exposure time for all rows of pixels to be the same, the start of exposure for each row also has to be adjusted accordingly. As a result, the entire image array is not exposed at the same time. The exposure mechanism is called rolling exposure, with the upper rows exposed earlier and lower rows exposed later. The impact of this rolling exposure is that the image captured will be distorted if the subject moves horizontally. That is, a vertical line will become slanted. This effect is especially significant when the resolution of the image is larger, because the time lag from the first row to the last row is longer than in low resolution sensors.

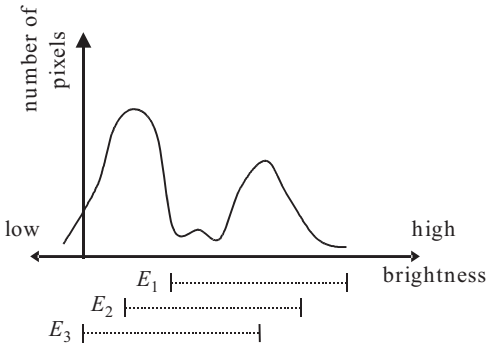


FIGURE 2.4

The dynamic range of the image sensor and the problem of auto exposure control.

2.2.2.2 Design Considerations of Auto Exposure Algorithms

Although high quality CCD/CMOS sensors have a larger dynamic range, most commercial imaging systems have narrower exposure latitude than color negative films. The dynamic range of a camera system is usually limited by the noise of the sensor as well as the precision of the ADC. The luminance level in the brightest area can be millions of times of that in the darkest area. When taking pictures in typical scenes, the dynamic range, which refers to the ratio of the highest and lowest level of light intensities, is usually higher than what most image sensors can measure.

Auto exposure (AE) is to determine the suitable aperture size, gain setting, and exposure time such that the exposure value best fits the dynamic range of the current scene. As shown in Figure 2.4, the luminance distribution of the scene is wider than the dynamic range of the sensor. Consequently, exposure needs to be adjusted in order for the sensor to capture proper range of luminance data. E_1, E_2, and E_3 represent three different exposure conditions. If selecting E_1, many pixels corresponding to the dark area are underexposed so that their output codes are zero. On the contrary, selecting E_3 may produce lots of saturated pixels whose values are clipped to the maximum output of the ADC. The brightness information is also lost. Therefore, a proper adjustment of exposure is needed to acquire more brightness information from the scene.

Typical AE algorithm adopts the additive system of photographic exposure (APEX) system [38]. Under APEX system, the relationship between the shutter time, aperture, luminance level and camera sensitivity is stated as follows:

$$A_f^2/T_s = B_s S_f/K \tag{2.1}$$

where A_f represents the F-number of the optical lens, T_s is the electronic shutter time (or exposure time), B_s is the scene brightness, S_f is the sensitivity of the camera, and K is a scaling factor. In practical system design, a log_2 data system is used for computational considerations. The unit under this data system is usually expressed as exposure value (EV), which means a difference of 1 EV is equivalent to change the shutter time by two times. The task of AE control is interpreted as follows: Given a good brightness measurement B_s of the current image, find a proper combination of A_f, T_s, S_f, and B_s to capture another image such that the exposure in the new image is optimal.

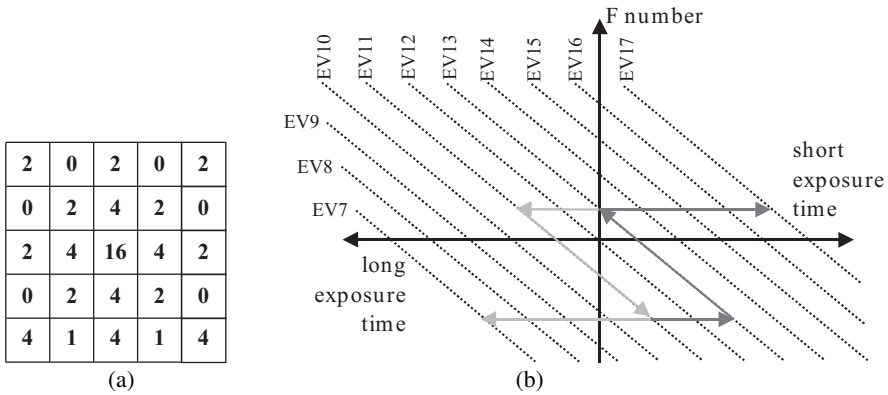


FIGURE 2.5

Auto exposure: (a) an example of the weighting matrix, and (b) auto exposure control curve. © 2005 IEEE

In preview mode, the brightness is measured based on the raw data of CCD sensor operating in draft mode in which only a few data lines rather than the entire sensor data are read out. Since the importance of different areas in the scene usually varies, a popular approach is to evaluate the importance index by analyzing the stability of brightness. As shown in Figure 2.5a, the raw image is further divided into 5×5 regions and the final measurement result is calculated based on the weighted values.

The adjustment of exposure time and gain setting must be performed before the next frame starts its exposure. A typical adjustment algorithm is based on the lookup curve shown in Figure 2.5b. In this figure, the slanted lines represent different scene brightness and the AE algorithm should move along the AE curve to find the proper combination of aperture size, exposure time, and gain setting. In the example, when the scene brightness increases, the algorithm will follow the solid trace, initially keeping the large aperture while reducing the exposure time. Then the algorithm switches to a smaller aperture at EV13 and increases the exposure time in order to maintain the same exposure. When the scene brightness increases further, the algorithm will keep with the same smaller aperture but further reduce exposure time. The reverse is similar but slightly different. As the scene brightness decreases, the algorithm will hold the same small aperture while increasing the exposure time to match the exposure. However, the algorithm will not switch aperture setting until it reaches EV11. In practice, the exposure time in this instance is 1/60 second to prevent the image blur problem. Note that the sensor sensitivity as well as register setting is usually different for draft mode and fine mode. The final exposure value for capturing still image must consider this factor which should be provided in the specification of the image sensor.

2.2.3 Camera Signal Processor

A CSP is designed for fast processing of massive image, video, and audio data and providing various peripherals to support user interaction and various IO extensions. Versatile camera features are realized by utilizing these hardware resources together. Figure 2.6 shows a block diagram of a typical CSP which includes EMP, DSP, peripherals and other

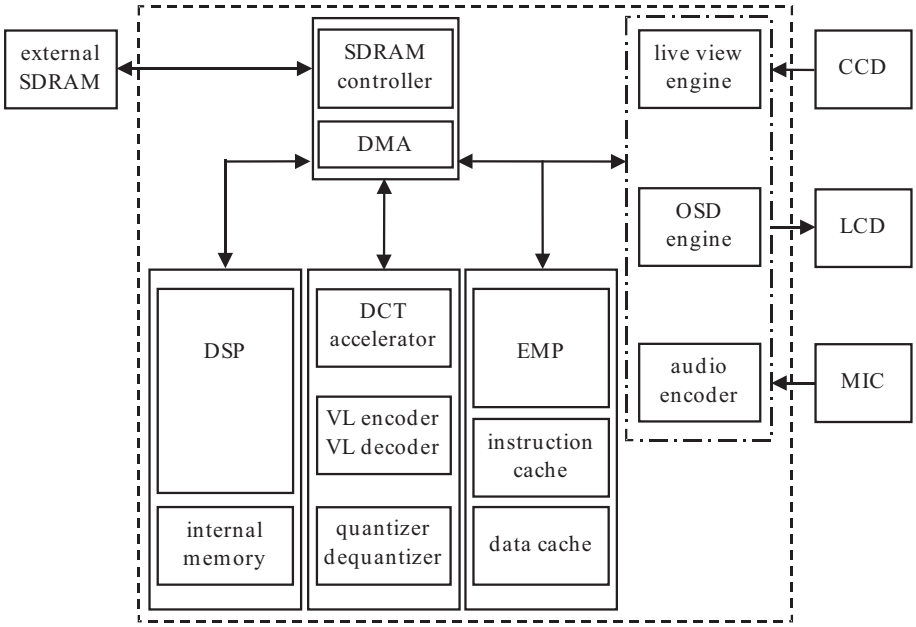


FIGURE 2.6

A typical color image signal processor (CSP).

engines. The EMP hosts the entire camera system, such as OS scheduling, interrupt handling, and interfacing to IO connection of the DSC system. The DSP, as its name shows, excels in executing signal processing programs. A multi-core solution provides the possibility of running jobs in parallel. Although performance is boosted in parallel processing with the multi-core architecture, the difficulty in controlling the cores also rises. The designer should carefully study the inherent property as well as limitation for each hardware resource.

An EMP is especially suitable for executing conditional branches, such as if-else, or switch-case statements. Most EMPs used in digital cameras are designed with a reduced instruction set computer (RISC) architecture. Although the EMP is able to execute most camera functions, it still cannot outperform a programmable DSP while processing image and video data. The EMP requires more instruction cycles to process visual data, while a DSP is particularly efficient for processing the data with the same properties by the same algorithm. A typical DSP has four or more high speed multiply-accumulator (MAC) units, hardware looping, and several on-chip memory buses. This is why multiple image and video data streams can be processed simultaneously in a DSP.

In addition to a programmable DSP, several dedicated engines might be included in a CSP for compressing image/video data following popular standards. Each engine can only execute one type of computation. They may include discrete cosine transform (DCT), quantizer/dequantizer, or variable length (VL) coding engines. Although the programmable DSP can also do these computations, providing these engines definitely provides the advantage of realizing parallel processing of visual data in pipeline stages.

Previewing scenes at 30 frames/second is a basic requirement for a DSC. However, the performance of a programmable DSP may not be high enough to satisfy this requirement. A specific functional module called preview or live view engine/accelerator is often built into the CSP to support camera preview and video recording modes. Since the live view engine processes raw CCD data to a formatted image, it is actually a fairly simple image pipeline. When the formatted data are generated from the live view engine, the OSD engine manages the display, and writes the data to LCD driver for display.

The role of the SDRAM controller and direct memory access (DMA) controllers is to manage data I/O among the processors, peripherals, and external SDRAM. Many hardware modules, including the live view engine, on-screen display (OSD) engine, DSP, and EMP, are connected to the SDRAM controller. However, transferring a large amount of data under the control of the EMP is too slow to satisfy the timing requirements. A CSP is often equipped with several DMA units to solve the bottleneck problem regarding data transfer among different hardware devices. The DMAs efficiently transfer blocks of data between SDRAM and the DSP or peripherals.

2.3 Embedded Software Platform

Although most camera functions are executed by several specialized hardware engines, software plays an increasingly important role in the design of camera systems. By increasing the range of software layout, it is possible to gain several advantages, like flexibility to change hardware components or add new user operations. The use of software facilitates reuse of previously designed software modules, independent from the selected hardware platform. The objective of design reuse can be achieved by designing the software modules at a processor and real-time operating system (RTOS) independent abstraction level.

In this section, an embedded software platform for developing versatile camera systems is described. For improving the programmability and reusability, the platform is divided into three programming layers and two interfaces: application layer, functional layer, system layer, application program interface (API), and device driver interface (DDI), as shown in Figure 2.7. To keep the application-specific features independent from the functional operations, program development of the application layer is based only on those APIs which are supported by the modules in the functional layer. No other lower level functions can be called by the modules in the application layer. This prevents the top-level routines from directly executing lower level operations or accessing hardware resources directly without being protected by any mechanisms. Direct function calls may result in an unpredicted state transition that may make the system hang in an unknown state. Even the modules in the functional layer can only call the service from other modules through API calls.

2.3.1 Software Programming Layers

In the application layer, two modules — GUI and manufacture/calibration interface (MCI) — are executed independently with their own control flows. The state diagram,

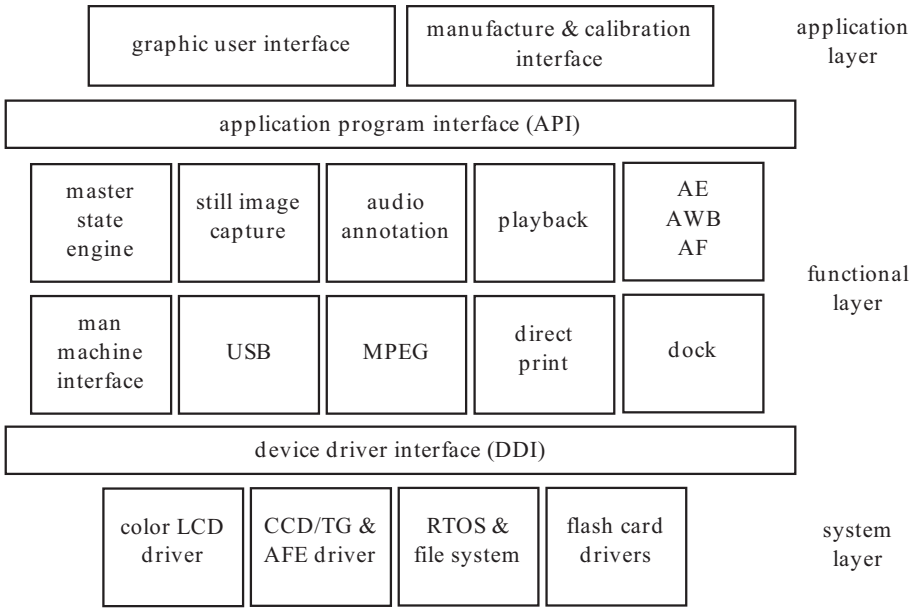


FIGURE 2.7

An example of an embedded software platform for digital cameras.

data flow, and dynamic behavior of the GUI module are customized by the GUI design specification. A typical GUI design specification defines the operational modes as well as their state diagrams. The common modes include system initialization, preview, still image capture, playback, video recording, direct print, and connection to a computer [39]. Although the GUI module is designed as several state machines, each of them is scheduled by one dedicated task. Once the user presses the shutter button, the GUI task will receive a message, and the message will be passed to a specific submodule in the GUI module.

The operational flow design of the MCI module is based on the manufacturing process adopted in the production lines. Its objective is to achieve high throughput, good product quality, and better production reliability. A good approach to designing MCI is to define common interfaces between computer and camera for the manufacture and calibration related APIs. Hence the MCI modules can be controlled by computer through a USB cable or operated by function calls inside the camera. Based on this approach, production engineers or line operators can monitor the manufacture/calibration progress and collect production data.

In the functional layer, some modules are designed to support versatile camera functions such as master state engine (MSE), still image capture, audio annotation, playback, man machine interface, USB, MPEG video and audio, and direct print. Each module runs under the scheduling of one or more tasks, which contain their own message boxes, state diagrams, and data flows. Each particular operation, such as still image capture with audio annotation or MPEG video and audio recording, is implemented with APIs supported by one or several functional modules. The communication between two modules is through message boxes or event flag settings.

In the system layer, the major design objectives are hardware abstraction and software programming environment construction. Since many cameras have similar hardware architecture with different types of image sensors, color LCDs, and flash cards, the software related to these key components should be designed as a configurable or replaceable one. Most of the code in the upper programming layers is reusable. In addition, by running the system with an RTOS, the programming environment can separate upper level functional operations from the detailed timing scheduling and file allocation mechanisms.

2.3.2 Software Design Reuse

The difficulties of embedded software reuse come from the following facts. i) The hardware components, such as zoom lens, CCD sensor, color LCD modules, analog front end chip and the type of storage cards, might be replaced while the next generations of digital cameras are being developed. ii) The CSP or RTOS may be changed for performance or cost considerations. iii) The design of the GUI flow is unique for each camera series in order to differentiate their product lines. The design team must develop a new GUI module for a new camera series. iv) A robust camera development platform must provide manufacture and diagnosis interfaces such that the design engineers can monitor device calibration parameters and system performance indices.

In order to maximize software reuse, the embedded system platform is abstracted at a level where the basic functional modules can utilize a device-independent interface to the hardware. The application software, which implements various GUI specifications, is designed by simply ignoring the implementation details of timing, scheduling, and resource allocation on the physical hardware platform.

Figure 2.8 summarizes the basic idea of a camera embedded software platform. Given a new camera system specification, designers first map the specification onto the hardware platform by choosing a suitable family of hardware key components. Then the designers develop drivers for specific components and link them with those predefined DDI functions. In the meantime, the application programmers analyze the GUI specification and design their state machines, menu system, and artwork. The key to concurrent hardware and software development is these predefined API/DDI functions, which abstract the behavior of functional modules and isolate hardware specific features and their implementation details.

2.3.3 Application Program Interface

The application program interface (API) provides a unique abstract representation of the functional modules with the implementation details hidden. With such APIs, the application software can easily be reused while developing new products. It is also possible to change the program modules located in application and functional layers for accommodating new applications features. A simple way to implement an API is:

```
API_ModuleName_FunctionName()
{
     SendMessage (ModuleName, FunctionName);
     WaitMessage (ModuleName, Finished);
}
```

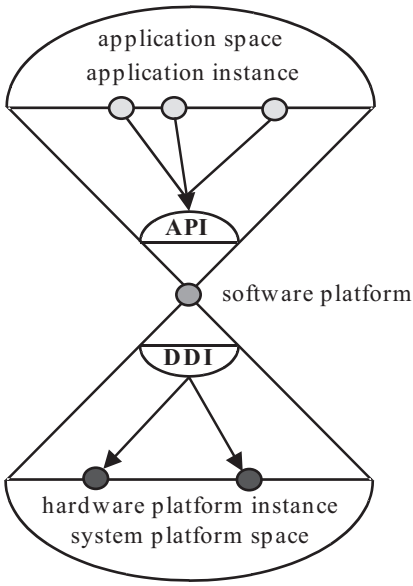


FIGURE 2.8

The design reuse of a camera embedded software platform. © 2005 IEEE

In such an implementation style, an API may not include the exact execution code to the functions it needs to execute, but it only sends a message to a specific module that executes the requested function. The module receiving the message will be invoked and switched into the ready state to carry out the task. It is worth noting that a task calling such an API will be switched to the blocked state and can only be reactivated after it receives the task-finishing message.

One problem that might occur while a task calls such an API is that the task cannot run any other code until it receives the task-finishing message. A periodic task, which is automatically invoked by a timer interrupt, monitors camera status and can send requests to stop the running processes. This task only triggers other modules but does not wait for responses. Therefore, it should not be blocked by any RTOS mechanisms.

2.3.4 Device Driver Interface

Similar to API design, the device driver interface (DDI) provides an abstract representation of the hardware device or hardware platform instances. The DDI isolates device-specific features and unifies device behaviors such that the program in the functional layer can execute without considering the types of device (e.g., storage media) in the system. The only thing needed to be modified is the low level driver that directly controls the storage cards. All program functions that access data from storage cards should be kept the same or only have minor changes.

2.4 Software Design Methodology

2.4.1 Task Scheduling by Real-Time Operating System

A modern camera executes several functions simultaneously to fully utilize all available resources on the hardware platform. Embedded software equipped with a real-time operating system (RTOS) to schedule all jobs is common in most of digital cameras. Unlike other powerful real-time kernels, the RTOS adopted in digital cameras is usually a scalable one in which only those services actually used by the camera are brought into the run-time image. The RTOS is loaded and executed by the EMP, yet jobs scheduled by RTOS are not only those running on the EMP but also those on the DSP or hardware engines. The RTOS achieves the illusion of concurrent processing by rapidly switching EMP among tasks and each task manages some jobs in a software module or controls the allocation of a hardware engine. Resource control mechanisms such as semaphores or event flags provided by RTOS are commonly used for maintaining the execution sequence of the required jobs.

As shown in Figure 2.9, a task can be in the running, blocked, or ready state, and four transitions are possible among these three states. Transition A occurs when a task is waiting for an occupied resource or a required event flag that has not been triggered. When the resource is available or the event occurs, transition B occurs. If no other task is running at the instant, transition C will be triggered and the task will immediately enter the running state. Transition D occurs only in preemptive scheduling scheme. It happens when the scheduler decides that the running task has occupied a resource long enough, and it is time to release the resource and activate another task. Note that transition D will not happen in non-preemptive scheduling. The task can only release the resource after it is finished or it is blocked when transition A occurs.

Preemptive scheduling is more powerful than non-preemptive, but the analysis of state transitions and resource allocations becomes more difficult than in non-preemptive scheduling. This is due to the fact that the running sequence for a specific operation scenario is unpredictable. It is more difficult to predict and meet timing constraints if using preemptive scheduling on a camera. Although RTOSs may allow assignment priorities for a task, it is infeasible to dynamically change the priority of a task to meet the timing constraint for different scenarios.

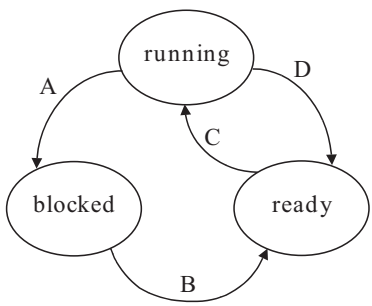


FIGURE 2.9

The states of a task.

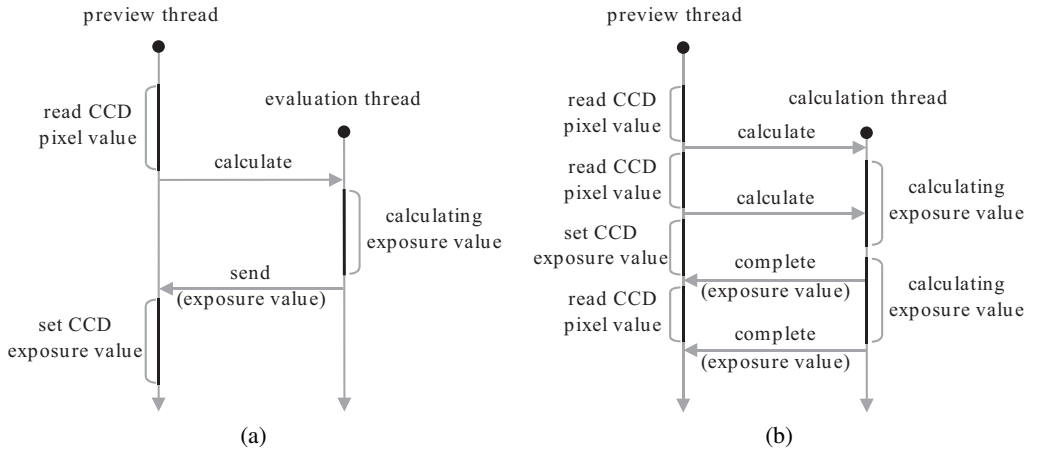


FIGURE 2.10

Sequence diagrams: (a) the sequence diagram of auto exposure in preview mode, and (b) the modified sequence diagram of auto exposure in preview mode.

Non-preemptive scheduling with the addition of some exception handling mechanisms is another good approach to digital camera design. Non-preemptive scheduling can cope with the difficulties of designing, debugging, reliability testing, and timing optimization. The main drawback of non-preemptive scheduling is that the task may occupy EMP resource if it is trapped in an unknown state or an infinite loop. Fortunately, such software issues or bugs can easily be detected in the development and verification phase. The timing constraint problem can be solved by adding extra code to interrupt routines with time-consuming jobs.

The software modules located in functional or application layers are managed by one or more dedicated tasks. With scheduling by RTOS, these tasks run independently. The initial state of a task is set to the block state and it can only be invoked by a message sent from another module. A typical example is where one task sends a specific message to another task to trigger a specific action. Figure 2.10a shows an example of the interactions of auto exposure with sequence diagram in unified modelling language (UML) notation. The preview task sends a message called CALCULATE to trigger the evaluation task after the CCD raw data readout is completed. After the evaluation task finishes the calculation, it sends the evaluated exposure value back to the preview task. The preview task sets the new CCD exposure time according to the evaluated exposure value.

Figure 2.10a shows a simple sequence diagram for auto exposure, which is executed by the EMP but the DSP is not involved. An alternative design is that the evaluation task running on the EMP establishes communication between the EMP and the DSP. The task does not evaluate the exposure value. Instead, it only handles the DSP resources and enables the DSP to calculate exposure value. A modified sequence diagram shown in Figure 2.10b shows the parallelism of running the EMP and the DSP simultaneously. While the DSP is calculating the exposure value for the previous frame, the preview task, which is executed by the EMP, sets the new exposure and gets the next frame data.

2.4.2 DSP Subsystem Management

Due to the program memory size limitation of the DSP subsystem, the DSP program is partitioned into several sections which include preview and capture, playback, and MPEG video/audio recording. Only one section is loaded into the DSP at a time according to the current camera operation mode. The interactions between the EMP and the DSP are through interrupts, where both the EMP and the DSP have corresponding interrupt service routines (ISRs) to handle the communication.

Since a software module running on the EMP is driven by messages or events, the ISR triggered by the DSP will send messages to the corresponding modules. As shown in following sample code, the module may load the corresponding DSP code first and then the task will be blocked on waiting for the DSP response. After checking the semaphore, the routine LoadDSP loads new DSP code. In addition, the event flag was set for controlling the execution of the task. The RTOS function call WaitDSPEventFlag in routine ProcessStillImage will block the corresponding task until the DSP finishes its job. This task can be reactivated only after the corresponding event flag is set again in the ISR. The ISR is triggered by the DSP subsystem once the assigned jobs are finished.

```
ProcessStillImage()
{
      WaitSemaphore (DSP) // Wait for DSP resource freed
      SetSemaphore (DSP); // Get the DSP resource
      ClearEventFlag (DSPFree); // DSPFree is the event flag indicating
                                // whether the DSP is free or not
      LoadDSP (Capture); // Load DSP codes of capture section
      StartImageProcess (); // Triggering DSP to run image processing
      WaitDSPEventFlag(DSPFree);// Waiting for the completion event flag
}
```

2.4.3 Hardware Accelerator Management

Several additional hardware accelerators such as a live view engine for scene previewing, a variable length encoder and decoder, and a quantizer and dequantizer for fast image processing may be included in a CSP. These accelerators play important roles in real-time image/video processing. The configuration registers in these accelerators are set through the EMP which also manages the data flow and protects these hardware accelerators from access by several software modules.

Controlling a hardware accelerator includes the following two steps:

1. The EMP first writes parameters into the corresponding registers of the accelerator. Then the mode of accelerator is switched from idle state to active state by setting the activating register. The setting of registers may not take effect immediately, but is usually activated with a predefined synchronizing signal.

2. The accelerator changes its state from active to idle after completing the job. Two popular mechanisms are used for checking the state of an accelerator: polling and

interruption. With the polling method, the EMP runs an infinite loop to check the ready bit of a control register in the accelerator. This may occupy all computational resources of the EMP. Using an interrupt would be more efficient than polling because the EMP can execute other tasks until the accelerator interrupts the EMP.

2.4.4 Dynamic Memory Management

The system may violate real-time constraints if the software uses too many system calls to dynamically allocate memory from the system heap. When the task executes dynamic memory allocation routines, the current execution task enters the blocked state for waiting the service of RTOS. Then RTOS searches suitable free space and allocates the required memory. For memory size and execution efficiency considerations, most RTOSs adopted in cameras are quite small. These systems usually lack a sophisticated garbage collection mechanism to compact memory fragments.

An alternative approach to dynamic memory allocation is handling it in the application software itself instead of the RTOS. One reason for this is that the memory arrangement for miscellaneous operation modes is different. Another reason is that the memory allocation and free operations in the RTOS may not achieve acceptable performance. To solve these problems, several basic memory allocation strategies can be adopted, including analyzing the memory usage for each camera mode such that the memory maps are customized for each mode, or statically assigning memory addresses for larger memory buffers and leaving the remaining space for dynamic memory allocation.

As shown in Figure 2.11, all camera operations are assigned to four basic modes with different memory maps in the camera software platform. In these maps, a few large memory buffers are assigned at fixed contiguous locations and the remaining space is used as heap memory for dynamic allocation. Since a typical RTOS cannot dynamically change the heap size and location, the only way to realize the memory allocation strategies is to develop a dedicated memory management routine that manages the heap size and location itself. It is particularly useful for USB connectivity mode, because several user functions are

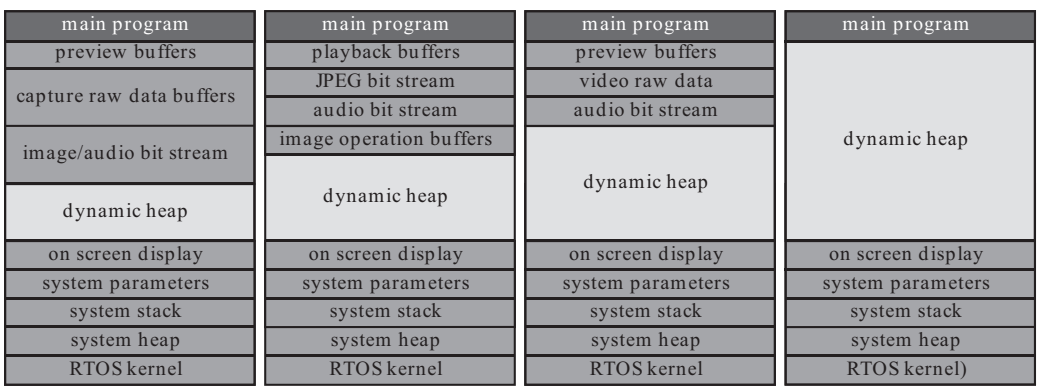


FIGURE 2.11

Examples of the memory maps corresponding to the preview/capture mode, playback mode, MPEG mode, and USB connectivity mode (from left to right). © 2005 IEEE

supported when the camera is connected to a personal computer or printer through a USB cable. The required memory size as well as the partitioning is unpredictable. For example, the memory buffer can be used for processing several printing pictures with different aspect ratios or layout formats. It can also process commands for setting an electronic mail with pictures. With this USB connectivity mode, memory can be allocated flexibly with higher efficiency.

2.5 Software Module Design Guidelines

The previous section focuses on the general design methodology of an entire software platform. The remaining important thing is how to design each software module to fully utilize all available hardware resources such that all jobs are completed within the specified timing constraints. Studying what hardware resources are available for accomplishing the desired operations is the first step in the design of a software module. As described above, a modern CSP integrates several powerful processors and hardware engines into a single integrated chip. The software running on the embedded microprocessor plays an important role in the coordination of the execution of all hardware resources.

The programming environment for traditional computers is different from that of camera systems in some aspects. Although both use operating systems to manage resources, in a digital camera, several computational resources can be used and many timing constraints are requested. The camera software designer should analyze available hardware resources and assign jobs to suitable hardware modules.

2.5.1 Available Hardware Resources Analysis

Analyzing available hardware resources for each camera operation would be the first step of software module design. The challenge of this step is to assign suitable hardware resources for specific jobs and to consider the data flow for camera operation. The hardware engines are controlled and configured through the EMP. The DSP subsystem can perform operations only after a program has been uploaded into internal program memory. The program on the DSP can be dynamically changed when the camera is operated in different modes, and some jobs can be executed in both the EMP and the DSP. The designers should optimize the system performance by balancing the loading of the EMP, DSP, and other accelerators. In addition to these engines, a DMA controller and SDRAM controller also play important roles in sharing data among them.

In order to explain the available hardware resource analysis strategies for software module design, two simple examples, previewing image on a display device and MPEG audio and video playback are illustrated in the following subsections.

2.5.1.1 Previewing Images on Color LCD

The function of the camera preview mode is to show real-time images with exposure and white-balance corrections. The live view engine is designed to support camera preview and

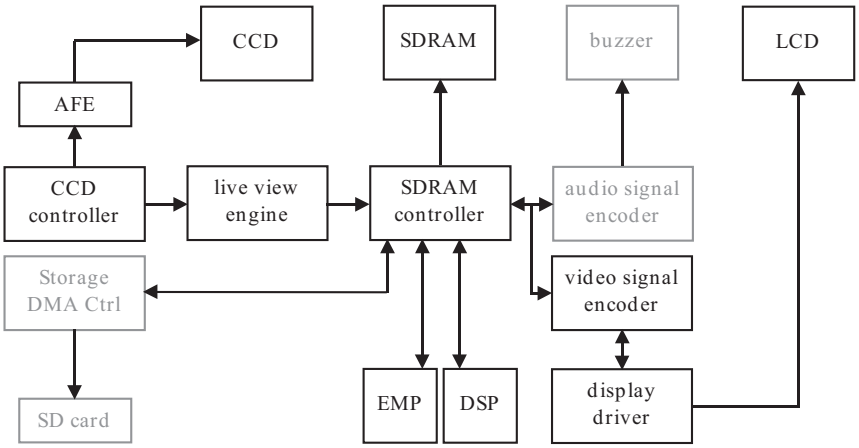


FIGURE 2.12

The hardware resource allocation in preview mode.

video recording mode. EMP and DSP are only involved in some low complexity calculations such as auto-exposure and auto-white-balancing that are performed only two or three times per second. As shown in Figure 2.12, several hardware resources are enabled for processing image signals from the image sensor: CCD/CMOS sensors, CCD timing controller, live view accelerator, SDRAM and its controller, EMP, DSP, video encoder, and color LCD display. To analyze the resource allocation, a data flow graph is designed in which arrowed lines represent the data flow of the captured image signals inside the camera:

1. The voltage signals of the CCD are first amplified and digitized by the AFE chip. By setting appropriate parameters in the CCD controller or timing generator, only the image data corresponding to the active area of the image sensor are read out and passed to the next stage.

2. The live view engine processes the image data that are screened by the CCD controller. A simple image pipeline, that performs color interpolation, white balance, noise filter, color correct, and tone and gamma correction, is included in the live view engine. This special pipeline aims for fast processing of the data without various image enhancement steps. It is rarely used for processing final pictures because the engine does not include a powerful noise filter, tone and color reproduction that are typical in CSPs.

3. A few frames are used for AE and AWB measurement each second. The DSP executes measurement of scene brightness and color temperature based on the raw frame data. The EMP reads the estimated parameters, adjusts exposure time of the CCD, and periodically sets new white balance gains in the engine.

2.5.1.2 MPEG Audio / Video Playback

Since the complexity of decoding MPEG video bit streams is still too high for the EMP, dedicated hardware modules for inverse quantization and inverse discrete cosine transform

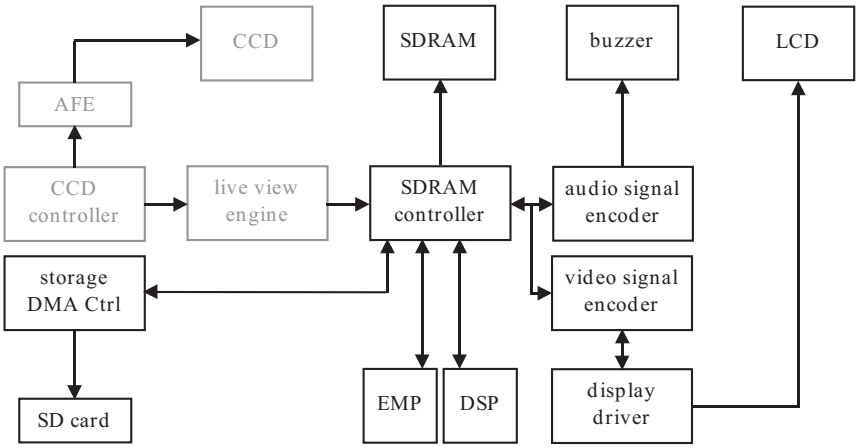


FIGURE 2.13

The hardware resource allocation in MPEG playback mode.

(DCT) may be included in the CSP. The key points of hardware resource allocation and data flow design include how to fully utilize DMA for transferring data in a background process and the approach to synchronizing EMP, DSP, and other hardware modules. The MPEG playback data flow from retrieving the MPEG bit stream to displaying video frames on the LCD is shown in Figure 2.13 and can be summarized as follows:

1. The compressed MPEG audio and video bit streams are transferred from the storage device to SDRAM through a DMA channel, which is managed by the DMA controller.

2. The compressed audio and video data stored in SDRAM is uncompressed by the DSP subsystem. Since the complexity of audio and video decoding is much less than in the encoding process, both audio and video decodings are usually performed by the DSP.

3. The decoded data is transferred from the DSP to SDRAM through a DMA channel. The DMA controller will signal an interrupt once video and audio data is decoded completely.

4. The decoded video data is displayed on the LCD while the audio data are played through an audio codec and speaker.

2.5.2 Job Scheduling and Resource Allocation

In a camera system, several software modules may execute simultaneously. The design needs to consider adopting several modelling tools such as finite state machines, Petri nets, and the collaboration and sequence diagram. This section introduces one of the tools for designing digital cameras rather than discussing the technology details of various modelling tools.

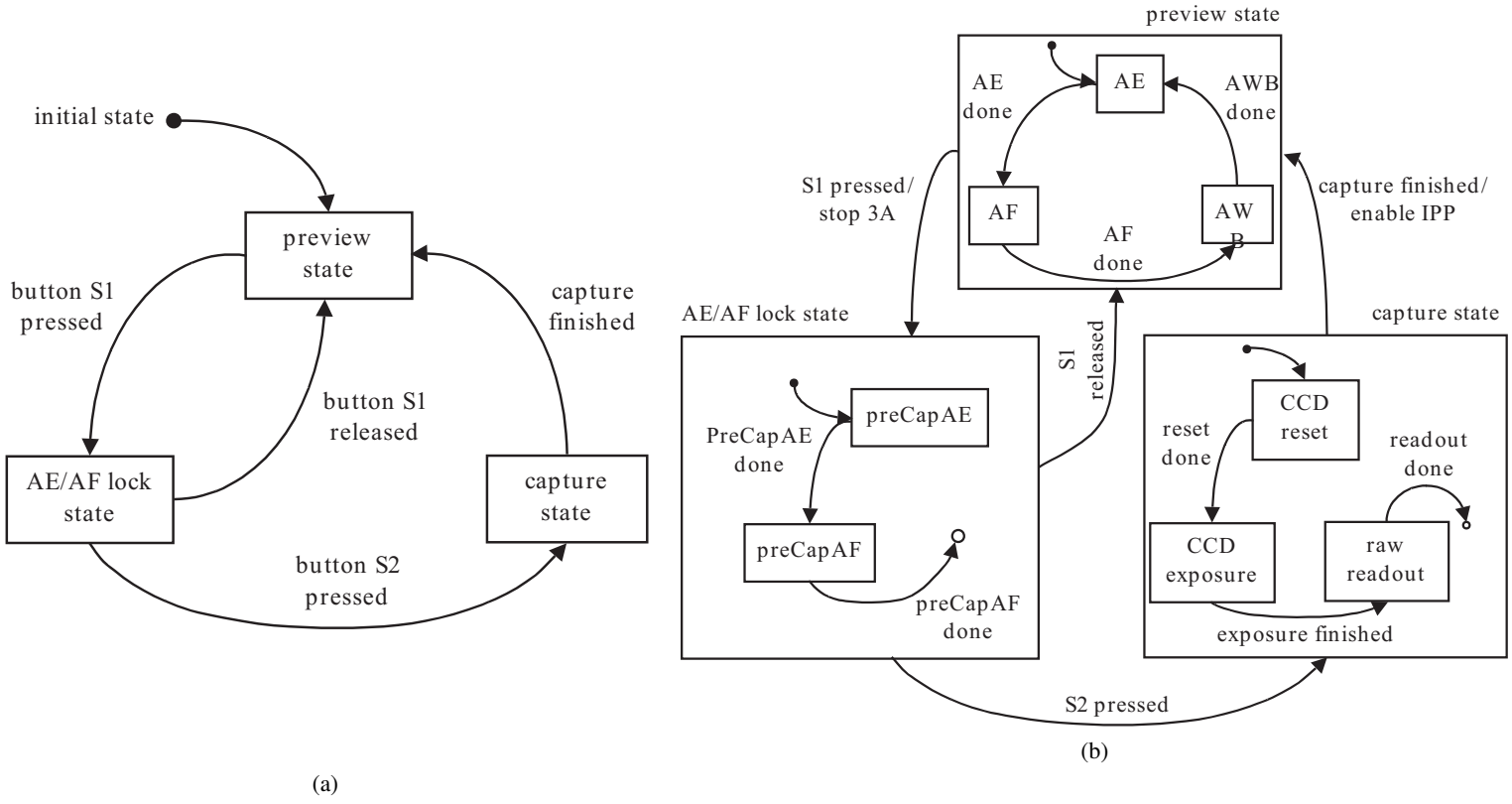


FIGURE 2.14

State diagram based on: (a) user's point of view, and (b) designer's point of view.

Job scheduling and resource allocation in cameras can be modelled by a state diagram. A complete state diagram must have state definitions, input events, state transitions and output actions. The design of state diagrams may have two alternatives based on user's and designer's points of view. Figure 2.14a shows the design of state diagrams from user's point of view. The default state is the preview state in which the camera shows real-time images with proper exposure and white-balance correction. To explain how to use state diagram for modelling jobs, a scenario for capturing still images is presented as an example. A digital camera has two stages of shutter button called S1 and S2. When a user presses button S1, the camera enters AE/AF lock state (also known as pre-capture state) in which the camera measures scene brightness and object distance. The state diagram shown in Figure 2.14a is easily understood by end-users but it is too rough to represent the detailed control flow of the software module.

Figure 2.14b shows a refined state diagram of the image capture module. There are three composition states, and each of them is decomposed into several sequential substates. The default state is the preview state in which the camera performs 3A (AE, AWB, and AF) periodically. When the S1 shutter button is pressed, the key scan task should detect that S1 shutter button is being pressed and send a message (S1 Pressed) to the image capture module. The image capture module stops the current 3A control loop and other activities of the DSP, and switches to AE/AF lock state after it receives this message. Note that the preview AE/AWB/AF, pre-capture AE/AF, and image processing pipeline are typically executed in the same DSP, but only one of them can be executed in the DSP at any one time. Since the preview AE algorithm is totally different from pre-capture AE, the capture module task calls Stop3A() routine to stop the execution of periodic 3A task and runs pre-capture AE one time. When calling Stop3A, the EMP sends an interruption to the DSP no matter what actions are executed on it. After that, DSP switches to the codes of pre-capture AE metering immediately. This diagram does not show the execution details of the image processing pipeline (IPP) but only sends an enabling message to the IPP task by calling the EnableIPP routine. Execution of IPP is not a real-time job, instead it is only a background progress. Each of the running jobs in the background processing task can be interrupted by a shutter button press event. The current status for these jobs will be temporarily saved. The remaining unprocessed data are queued into buffers until the DSP resource is freed.

The modules located in functional or application layers are allocated by one or more tasks. With scheduling done by the RTOS, these tasks run independently and their initial states are set to idle. An idle task can only be invoked after receiving a trigger message sent from another module. The button press events or hardware and software timer interrupts are the triggering source of the entire camera system. In order to help illustrate the module design concept, an example of the programming structure is shown below, in which partial codes of the module ImageCaptureModuleTask are listed.

The module includes an infinite loop, which waits for messages from the RTOS with the routine WaitMessage(). The received message will be dispatched to other functions that handle different messages accordingly in order to perform right actions in current operation mode. This coding style is the fundamental mechanism of task scheduling without using preemptions in the software system. For other modules in the application and functional layers, a similar programming structure is adopted such that tasks do not occupy computation resources until they receive messages. Note that messages or events are normally

triggered by external hardware signals or internal timer interrupts. The corresponding interrupt service routines send messages to specific modules. The hardware interrupt sources include key-press or key-release events, ADC/DAC, real-time clock (RTC), CCD timing generator and the DSP subsystem.

```
void ImageCaptureModuleTask()
{
      while (true) {
            Message = WaitMessage ();
            switch (CurrentState) {
                  case PreviewState:
                      PreviewStateMsgFunc (Message);
                      break;
                  case AEAFLockState:
                      AEAFLockStateMsgFunc (Message);
                      break;
                  case CaptureState:
                      CaptureStateMsgFunc(Message);
                      break;
                  ...
            }
      }
}
void PreviewStateMsgFunc (MSG Message)
{
      switch (Message) {
            case S1_Pressed:
                  StopDSP ();
                  EnterState (CaptureState);
                  break;
            case xxButton_Pressed:
                  ...
      }
}
```

2.5.3 Background Processing and Data Buffering

Data buffering plays a key role in background signal processing within the camera. Some camera operations have tighter real-time constraints than others. Users may be concerned about response time when they press particular buttons. Among all operations of a digital camera, color image and video processing and data compression are the most time-consuming. Fortunately, it is not difficult to design a camera that has faster response to key press events by leaving previously taken images unprocessed. All unprocessed images are first queued into a buffer. The jobs of image processing and compression are executed sequentially and scheduled by the background processing task.

The design of the background processing task must consider memory requirements, complexity of individual tasks, and the availability of hardware resources. Memory requirements are one of the critical issues in the camera and will determine the number of pictures allowed to be left unprocessed in the buffer. For example, storing the raw data for a six mega-pixel picture requires 12 MB of memory if a 10-bit or higher precision ADC is adopted. The maximum number of queued images will be less than five if 64 MB is installed in the camera. Another application case is to record audio/video with unlimited recording length within the capacity of the storage card. Two ping-pong buffers can be used to realize parallelism of data processing and data streaming rather than allocating a huge memory pool in such cases. Since the required memory size for ping-pong buffers is more flexible, allocating these buffers is much easier and may produce better memory usage. In the following text, two design cases are stated explaining the concepts of continuous still image capture and real-time MPEG audio/video recording.

2.5.3.1 Continuous Still Image Capture

An advanced digital camera can take several pictures continuously. Users are usually concerned about how fast pictures can be taken in continuous mode. The time frame between two successive shots may be too short to finish the image processing or compression for a picture. In practice, the image capture task will immediately switch back to preview mode after raw data readout is done. As described in the previous section, the image capture task sends an enabling message to the IPP task by calling the EnableIPP routine. The remaining color image processing and real-time streaming jobs are taken over by IPP task which is a background processing job. Similar to other tasks in the embedded system, the IPP task is triggered when it receives an enabling message.

As shown in Figure 2.15, the captured raw image data are first queued into buffers and then the remaining jobs are taken over by the IPP task. While the images are being processed, users may press the shutter buttons (S1 and S2) again which will result in the DSP resource being switched to execute pre-capture AE and AF. The raw data stored in the image data buffers may not be completely processed in time. Two or more images may be queued into the raw data buffers. Since Joint Photographic Experts Group (JPEG) format is the most popular image file format adopting macro blocks as a basic data unit, a straightforward way for the IPP task is to also use macro blocks which contain 16×16 pixels as a basic unit. The DSP handles raw image data within the image processing pipeline and writes the compressed bit stream to one of the ping-pong buffers. Once the current bit stream buffer is full, the DSP interrupts the microprocessor to enable the background storing task, which controls DMA for streaming data onto the storage card. The DSP continues to process raw image data if no other actions are enabled while the DMA is streaming data to the card. This realizes parallelism of color image processing, compression, and real-time streaming. Note that actions in the background processing task may be interrupted at any time when receiving the message that the user has pressed the shutter button.

2.5.3.2 MPEG Audio/Video Recording

Video signal processing, compression and real-time streaming would be the most time consuming jobs in a digital camera. Since the DSP subsystem is always tied up with motion

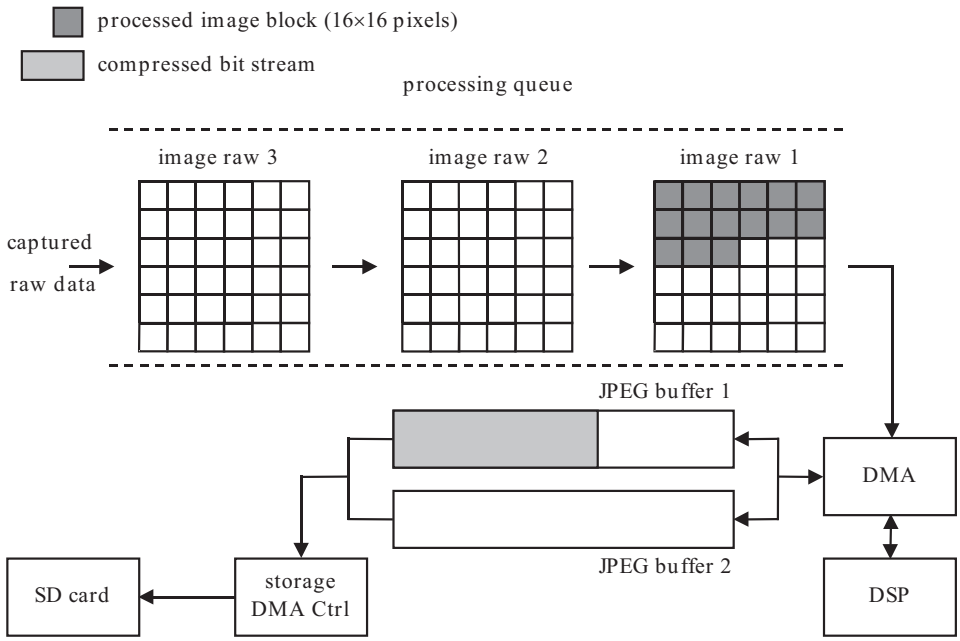


FIGURE 2.15

Background processing for still images.

estimation and motion stabilization calculation for video compression, it is better to assign other data processing jobs such as audio signal processing to the EMP. In addition to fundamental video signal processing steps, bit rate control also highly affects the final quality of recorded video. The target bit rate must be a compromise between video quality, bit stream size, and the streaming speed of the storage card. Hence, the background processing should continuously monitor the average write speed of the inserted storage card.

Audio data compression is triggered by DMA interrupts and executed in the microprocessor. The analog-to-digital converter (ADC) converts the analog signal sent from the microphone and stores the digital data into a pair of ping-pong buffers. The DMA activates an interrupt once one of the buffers is full, and then the EMP takes over the remaining audio data noise filtering and compression processes. On the other hand, access of video data is triggered by vertical sync signal of CCD. The data are processed in both the live view engine and the DSP system. Note that typical video data is sent from CCD sensors at 30 frames/s, but not all of the frames are processed due to the limitation of computation capability. In the meantime, the microprocessor runs AE/AWB adjustment based on the results sent from DSP and then synchronizes video and audio bit streams to generate MPEG files.

Due to large variations in the bit rates of the video bit stream and the write speed of different brands/types of storage cards, the task of real-time streaming audio/video data to storage cards is not as easy as that performed in personal computers, which usually enjoy much larger buffer memories and faster CPUs. In many cases, even flash cards of the same brand have different write speeds among cards of different memory size due to the effect of internal SRAM buffers. In a practical example, the design can incorporate a rate control algorithm and the described buffering scheme to automatically balance video quality and

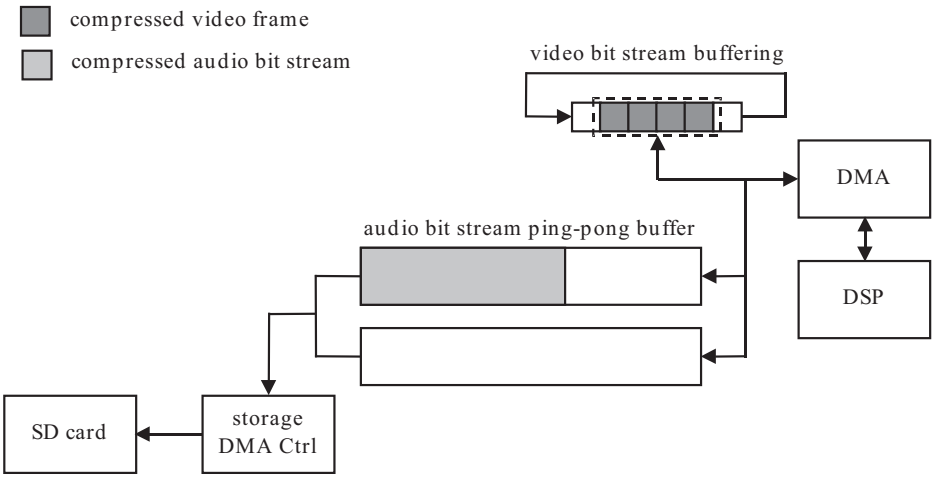


FIGURE 2.16

Audio/video bit stream buffers.

write speed of the storage card. As shown in Figure 2.16, the available SDRAM space is organized as circular and ping-pong buffers for video and audio bit streams, respectively. Given a default video compression rate, the DSP starts MPEG video compression, and DMA is enabled for streaming the data to the storage card as data crosses block boundaries. The ratio of the total number of audio and video frames can be calculated based on the recorded time stamps and the target bit rate is adjusted dynamically according to the free space in the buffers.

2.5.4 Power Aware Design

Portable devices rely on batteries to provide power and thus power consumption is an important issue in achieving long battery life. Besides selecting excellent battery technology and components with low power consumption, there is a lot that can be done with embedded software to achieve optimal performance. The events with the highest power consumption are as follows; changing zoom step, running auto-focus, enabling the mechanical shutter driver, switching aperture size, and charging the strobe capacitor. It is easy to ensure that the aperture and mechanical shutter not be activated together.

In general, there are several features of battery and power circuits that the designers should keep in mind. First, there is always an effective series resistance of the battery. When the current drawn from the battery is higher, the voltage drop across the series resistance is higher and the terminal voltage of the battery becomes lower. Lower battery terminal voltage usually results in lower power circuit efficiency and, for the same output current, draws even higher current from the battery. Higher voltage drops across the series resistance not only waste energy but also dissipate more heat. Therefore, it is always a good strategy to keep the battery current low.

Second, the effective series resistance is not constant and the characteristics of the battery change drastically close to energy depletion. As the voltage of the battery reduces, there is

a threshold voltage below which the battery will stop providing power. The software needs to be able to detect this threshold and stop the device before this point is reached. There are several things that need to be taken into consideration when developing embedded software for a digital camera: i) turn off the power to the circuits not in use, ii) monitor the battery level and adjust the power control accordingly, iii) when possible, gauge and calculate the remaining energy stored in the battery, and iv) arrange the timing such that circuits with high power consumption are not activated at the same time.

2.6 Embedded Software Design of Built-in Automatic Camera Calibration

The characteristics of some hardware components such as sensors, lens modules and mechanical shutters are usually inconsistent among different camera instances although each of them may be produced by the same manufacturers of the same component. The specifications of these components show that they can only be guaranteed within an acceptable range. Unfortunately, the final picture quality highly depends on several characteristics of the image sensors. Namely, the variations of sensitivity for different channels, the sensor saturation voltage in preview and capture modes, the effective aperture ratio for different F-numbers of the lens, the percentage of lens falloff as well as the best active window aligned with the sensor, the photo response non-uniformity among the different pixels of the same sensor, bad pixel identification, and the extra DC voltage generated by the sensor dark current and circuit offset. It is difficult to produce good images if one of the above items is not calibrated accurately or otherwise compensated for.

The design of a camera model can only be done based on a set of component parameters and variations in the hardware components must be calibrated. The best way is to incorporate an automatic calibration function into the developed software platform. Each camera can be calibrated in the production line without connecting to any computers, or where computers are required only at the start of production line for initializing calibration process. The number of computer based test fixtures needed is drastically reduced. In many cases the same test environment can accommodate several cameras and perform calibration simultaneously. If calibration data need to be collected for statistical analysis, they can either be stored on the camera or on a memory card. It is always possible to upload these data at a final quality control station. This system increases the throughput dramatically and reduces the mistakes made in the production line. The following sections first describe the system flow and then discuss detailed calibration methods for the two major calibration items.

2.6.1 Automatic Camera Calibration Flow

Figure 2.17 shows the calibration flow of the proposed system. Since the program is embedded in the camera, it can run these five steps automatically. The only equipment needed for these calibration items is a standard light box that provides stable and uniform light. The intensity and color temperature of the light box needs to be measured and cal-

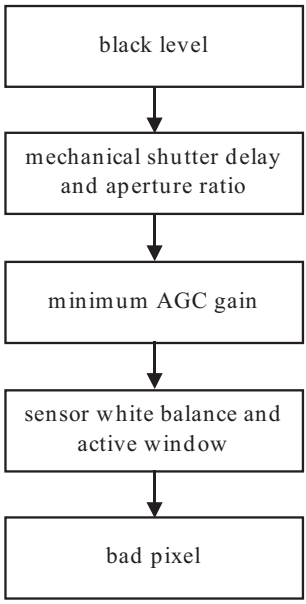


FIGURE 2.17

The calibration flow of the proposed system. © 2005 IEEE

ibrated periodically. The design methodology of this camera calibration flow is based on the following observations:

- The captured raw data contains DC components caused by sensor dark current and other offset voltages inherent in analog circuits. These components must be removed before the data can be used for other calibration processes. Even though many commercially available analog front end processors can already provide very good performance in reducing the effect of optical black level, it is still good practice to verify the result and hence this step is performed first.

- Minimum automatic gain control (AGC) setting determines the linear region of the optoelectronic conversion function (OECF) of the sensor response for preview and capture modes. This is needed for the sensor to provide highest dynamic range for image capture. However, to make sure the still image capture operation works correctly, it is necessary to calibrate the mechanical shutter delay before performing minimum AGC setting calibration.

- Mechanical shutter delay calibration must be executed for all aperture sizes, since the associated mechanical shutter delays are different. Hence the captured raw data can also be used for aperture ratio calculation.

- The sensor white balance calibration must rely on accurate AGC settings to prevent the sensor from operating in the nonlinear region. Thus, the calibration of the minimum AGC setting should be performed before calibrating the sensor white balance.

- Active window adjustment can be combined with sensor white balance calibration because the optimal exposure time and gain setting have been determined in this stage.
- The bad pixel identification stage is carried out as the last step because its exposure condition must be well-controlled.

For the black level, sensor white balance, active window, and bad pixel calibrations, it is only necessary to take one or two raw images for simple statistical analysis. This section only discusses the mechanical shutter delay and minimum AGC setting calibrations.

2.6.2 Mechanical Shutter Delay Calibration

A typical timing diagram of a CCD based digital still camera is shown in Figure 2.18. As long as the pixels are exposed to the incident light, the image charge will be accumulated and the amount of charge is proportional to the exposure time. The electronic shutter is composed of a sequence of reset pulses (RP), which will clear the image charge accumulated on the sensor elements called photosites. With consecutive RPs, the image charge stored on the photosites will be cleared until the last RP finishes. The time duration between the last RP to the time instant that the final image charge is transferred to the vertical CCDs is the so-called effective electronic shutter time.

In high resolution commercial digital still cameras, the vertical CCDs are not able to transfer the entire array of image charges at one time. Consequently, the image array is usually split into two or more fields and the entire image data is read out field by field. When the first field is being read out, the remaining fields are still exposed to the incident light if it is not blocked and this will result in uneven exposure. A popular solution is to use a mechanical shutter to block the incident light completely. The result is a combination of an electronic shutter to control the start of the exposure and a mechanical shutter for the end of the exposure.

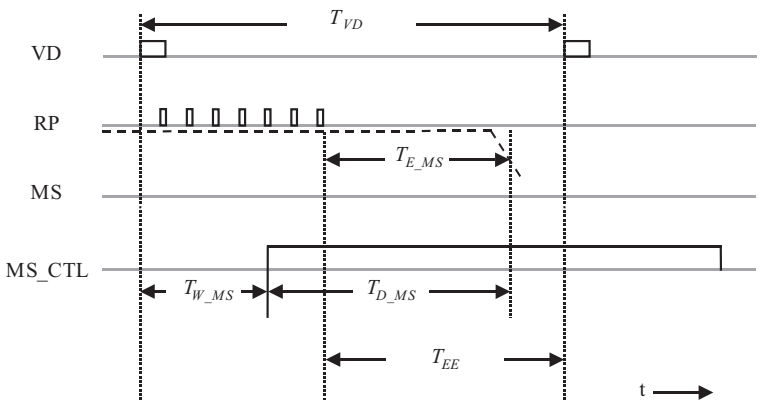


FIGURE 2.18

Typical CCD exposure timing. The term T_{D_MS} denotes mechanical shutter delay time, T_{W_MS} stands for mechanical shutter control signal wait time, T_{EE} is electronic shutter exposure time, T_{E_MS} denotes mechanical shutter exposure time, and T_{VD} denotes VD cycle time. © 2005 IEEE

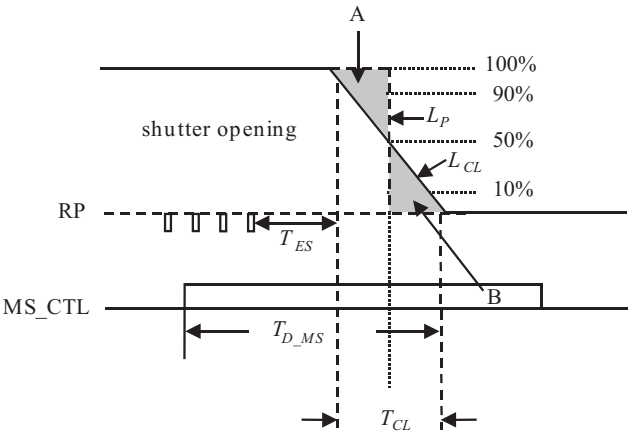


FIGURE 2.19

Shutter closure and delay time. © 2005 IEEE

A typical mechanical shutter (MS) is driven by a solenoid whose speed is finite as shown in Figure 2.19. This is because the solenoid needs to accumulate enough energy to overcome static friction and move the shutter blades. In addition, the delay time and shutter closure time also vary from component to component. In real applications, if we plot the effective opening area of the mechanical shutter as a function of time, the shutter closure curve can be approximated by a slanted line L_{CL}. The time from 90% to 10% of the full opening of the mechanical shutter is usually called the closure time T_{CL}. By assuming that area A is approximately equal to area B, the closure of the MS can be replaced with an equivalent perfect shutter at the instant of 50% of full opening, which is represented by the vertical line L_P. The effective exposure time contains two parts, one is the time T_{ES} that the mechanical shutter remains completely open, and the other is roughly $T_{CL}/2$. In normal applications, the reset pulses should not enter the shutter closure period.

Referring to Figure 2.18, the mechanical shutter is calibrated as follows. The period between the VD pulses T_{VD} is known, and is usually designed to be close to 1/30 second. Define electronic shutter time T_{EE} as the time between the falling edge of the last RP and the start of the next VD pulse. This electronic shutter time can be set in software. We can set a mechanical shutter control signal wait time T_{W_MS} and try to measure the mechanical shutter delay time T_{D_MS}. The effective shutter time composed of electronic shutter and mechanical shutter time is designated as T_{E_MS}. The relation between the above variables is shown in the following equation:

$$T_{VD} - T_{EE} + T_{E\_MS} = T_{W\_MS} + T_{D\_MS} \tag{2.2}$$

The calibration procedure is to try different values of electronic shutter time T_{EE} to find the point where the effective shutter time T_{E_MS} becomes zero. When this happens, the CCD output voltage will become zero. Under such conditions, we can obtain the following equation by substituting T_{E_MS} with zero and $T_{EE} = T_{eez}$:

$$T_{VD} - T_{eez} = T_{W\_MS} + T_{D\_MS} \tag{2.3}$$

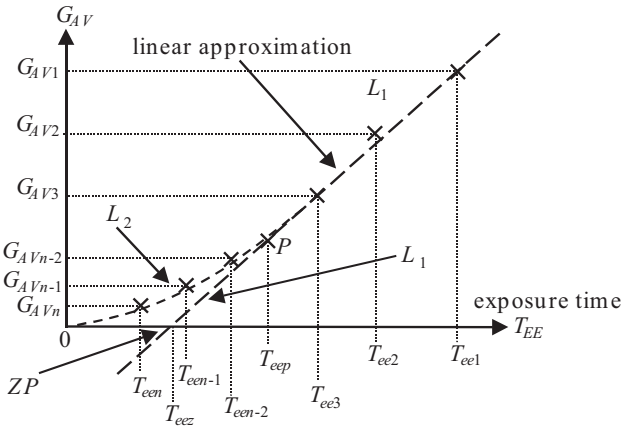


FIGURE 2.20

Shutter characteristic. © 2005 IEEE

From Equation 2.3, the mechanical shutter delay time T_{D_MS} can be calculated. It is then possible to adjust the value of T_{W_MS} in order to make the instant that the mechanical shutter actually closes fixed relative to the leading edge of VD.

The difficulty in finding the closing point ZP in Figure 2.20 is that the output level (G_{AV}) of the CCD is very low. The data tends to be corrupted by noise when the effective shutter time is extremely short. In addition, when the electronic shutter time is close to the mechanical shutter closure time, the effective exposure will be affected by the mechanical shutter tolerance. Therefore, we can only rely on the region where the effective exposure time is long enough such that the CCD output level is relatively high. In real applications with 10-bit analog-to-digital converter (ADC) output, the starting point of the electronic shutter time T_{ee1} is adjusted such that the output level G_{AV1} is approximately 800. When the data is plotted, it is possible to adopt minimum mean square error method to find linear approximation to the data points. The approximated line is represented as L_1, and it is extended so that it intersects with the horizontal axis at the crossing point ZP. The equation of line L_1 has the following form:

$$G_{AV} = \alpha \cdot T_{EE} + \beta \tag{2.4}$$

whereas the location (T_{eez}) of the point ZP can be obtained as follows:

$$T_{eez} = -\frac{\beta}{\alpha} \tag{2.5}$$

where α and β are the slope and the offset value of the line L_1, respectively.

2.6.3 Image Sensor Calibration

The OECF curve of typical CCD output usually contains linear, nonlinear and saturation regions, as shown in Figure 2.21a. Only the linear region is suitable for image processing [40]. The pixels operating outside the linear region are very difficult, if not impossible, to handle using typical image processing pipelines. The main difficulty is that this nonlinearity

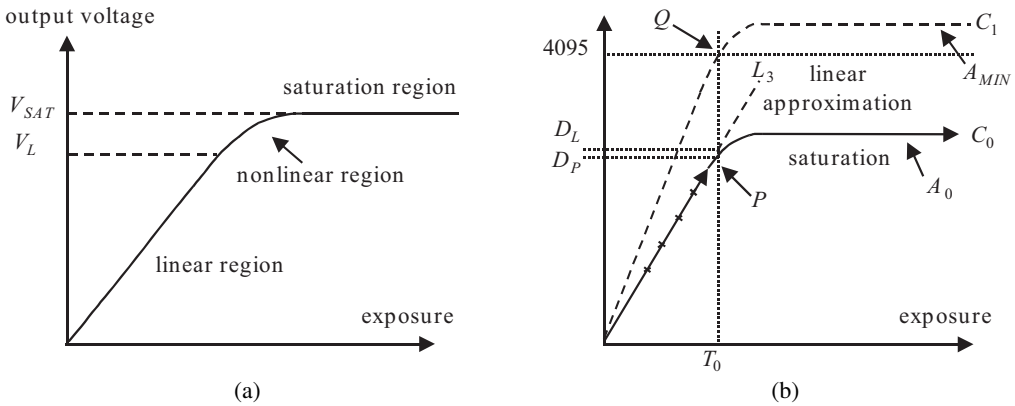


FIGURE 2.21

(a) Optoelectronic conversion function of CCD sensors. (b) Minimum AGC setting calibration. © 2005 IEEE

is usually not consistent among different sensors, thus it is difficult to characterize and utilize this nonlinearity. In order to provide better sensitivity and wider dynamic range, it is necessary to operate CCD sensor with the maximum range of its linear region. This requirement can be achieved by finding the AGC setting so that the upper edge of the linear range matches the maximum input voltage V_R of the ADC. The corresponding AGC parameter is called the minimum AGC setting A_{MIN}. Under normal lighting environments, the AGC is always set at A_{MIN}, but higher AGC settings will be used to increase the effective sensitivity in darker environments. This setting can guide the auto exposure control to prevent the sensor from operating in the nonlinear region.

The proposed calibration process relies on the ADC result for analysis. A typical example of adopting 12-bit ADC is shown in Figure 2.21b. First, a relatively low AGC gain value of A_0 is set such that the ADC output value of the saturation region is much lower than the full scale output of the ADC, which is 4095 in this case. Then a few raw images are taken with different shutter times and the center green pixels are averaged to check for the output level of the CCD. The associated OECF curve is represented by C_0. From these data points, a linear approximation line L_3 can be derived. In the next step, the shutter time is increased further while the average data of the captured raw image is compared with the data calculated from the linear approximation. At time T_0 it is found that the actual data D_P at point P is lower than the estimated data D_L by about 3%, this can be judged as the end of the linear region of the OECF. It is straightforward that additional gain needed to bring point P to point Q is $A_1 = 4095/D_P$. With the overall AGC gain value set as $A_{MIN} = A_0 \times A_1$, the resulting OECF curve is C_1.

2.7 Conclusion

A design of versatile digital cameras, which supports both attractive features in user operation mode and calibration/testing functions in engineering mode, is a complex process.

Robust embedded software platforms can make the development of digital cameras fast and shorten the design cycle time. This chapter described a camera software platform that has been successfully used in developing several consumer cameras. Both major hardware components and operation modes supported by this platform allow for easy understanding of the camera hardware architecture and practical camera design. In addition, the proposed embedded self-calibration flow and sensor/shutter calibration algorithms give a valuable reference for efficient construction of consumer cameras in mass production lines.

Acknowledgment

Figure 2.1, Figure 2.8, and Figure 2.11 are reprinted from Reference [25], Figure 2.5 is reprinted from Reference [26], and Figure 2.17 to Figure 2.21 are reprinted from Reference [37], with the permission of IEEE.

References

[1] S. Kawamura, "Capturing images with digital still cameras," *IEEE Micro*, vol. 18, no. 6, pp. 14–19, November-December 1998.

[2] N. Nakano, R. Nishimura, H. Sai, A. Nishizawa, and H. Komatsu, "Digital still camera system for megapixel CCD," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 581–586, August 1998.

[3] S. Okada, Y. Matsuda, T. Yamada, and A. Kobayashi, "System on a chip for digital still camera," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 584–590, August 1999.

[4] M.J. Loinaz, K.J. Singh, A.J. Blanksby, D.A. Inglis, K. Azadet, and B.D. Ackland, "A 200-mW, 3.3-V, CMOS color camera IC producing 352×288 video at 30 frames/s," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 12, pp. 2092–2103, December 1998.

[5] D. Talla, C.Y. Hung, R. Talluri, F. Brill, D. Smith, D. Brier, B. Xiong, and D. Huynh, "Anatomy of portable digital mediaprocessor," *IEEE Micro*, vol. 24, no. 2, pp. 32–39, March-April 2004.

[6] K. Illgner, H.G. Gruber, P. Gelabert, J. Liang, Y. Yoo, W. Rabadi, and R. Talluri, "Programmable DSP platform for digital still cameras," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, AZ, USA, March 1999, vol. 4, pp. 2235–2238.

[7] COACH-9m, *Camera-on-a-chip, Digital camera processor*. Technical Report, Zoran Corporation, 2005.

[8] S. Miyashita, H. Teshirogi, T. Sato, T. Nakajima, and K. Nishio, "A camera chipset for dual-mode mega-pixel camcoders," in *Proceedings of the IEEE International Conference on Consumer Electronics*, Los Angeles, CA, USA, June 2000, pp. 174–175.

[9] TMS320DM310, *Digital Media (DSP) Technical Reference Manual*. Texas Instruments, 2003.

[10] S. Okada, Y. Matsuda, T. Yamada, and A. Kobayashi, "System on a chip for digital still camera," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 584–590, August 1999.

[11] H. Mori, T. Hanagata, H. Nakada, N. Gamou, Y. Taura, T. Nakajima, D. Kumagai, and N. Osawa, "A digital color camera LSI chip set for multiple applications," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 3, pp. 725–731, August 1997.

[12] G.H. Smith, *Camera Lenses: From Box Camera to Digital*. Bellingham, WA: SPIE Press, 2006.

[13] S.C. Park and R.R. Shannon, "Zoom lens design using lens modules," *Optical Engineering*, vol. 35, no. 6, pp. 1668–1676, June 1996.

[14] Y. Ogata, "Zoom lens system," U.S. Patent 4 789 226, December 1988.

[15] I.A. Neil and E.I. Betensky, "High performance zoom lens system," U.S. Patent 6 122 111, September 2000.

[16] N. Nanba, "Zoom lens system and camera having the same," U.S. Patent 6 931 207, August 2005.

[17] P.L.P. Dillon, D.M. Lewis, and F.G. Kaspar, "Color imaging system using a single CCD area array," *IEEE Journal of Solid-State Circuits*, vol. 13, no. 1, pp. 28–33, August 1978.

[18] J.R. Janesick, *Scientific Charge-Coupled Devices*. Bellingham, WA: SPIE Press, 2001.

[19] J. Nakamura (ed.), *Image Sensors and Signal Processing for Digital Still Cameras*. Boca Raton, FL: CRC Press, 2005.

[20] A.E. Gamal and H. Eltoukhy, "CMOS image sensors," *IEEE Circuits and Devices Magazine*, vol. 21, no. 3, pp. 6–20, May-June 2005.

[21] K. Yoon, C. Kim, B. Lee, and D. Lee, "Single-chip CMOS image sensor for mobile applications," *IEEE Journal of Solid State Circuits*, vol. 37, no. 12, pp. 1839–1845, December 2002.

[22] E.R. Fossum, "CMOS image sensors: electronic camera-on-a-chip," *IEEE Transactions on Electron Devices*, vol. 44, no. 10, pp. 1689–1698, October 1997.

[23] Y. Fujimoto, H. Tani, M. Maruyama, H. Akada, H. Ogawa, and M. Miyamoto, "A low-power switched-capacitor variable gain amplifier," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 7, pp. 1213–1216, July 2004.

[24] M. Koen, "An analog-to-digital processor for camcorders and digital still cameras," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 570–580, August 1998.

[25] W.C. Kao, C.C. Kao, C.K. Lin, T.H. Sun, and S.Y. Lin, "Reusable embedded software platform for versatile camera systems," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1379–1386, November 2005.

[26] W.C. Kao, S.H. Chen, T.H. Sun, T.Y. Chiang, and S.Y. Lin, "An integrated software architecture for real-time video and audio recording systems," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 3, pp. 879–884, August 2005.

[27] W.C. Kao and S.Y. Lin, "Various auto exposure control strategies for digital cameras," *Images & Recognition*, vol. 12, 2006.

[28] S. Shimizu, T. Kondo, T. Kohashi, M. Tsuruta, and T. Komuro, "A new algorithm for exposure control based on fuzzy logic for video cameras," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 3, pp. 617–623, August 1992.

[29] T. Kuno, H. Sugiura, and N. Matoba, "A new automatic exposure system for digital still cameras," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 192–199, February 1998.

[30] C.M. Chen, C.M. Hong, and H.C. Chuang, "Efficient auto-focus algorithm utilizing discrete difference equation prediction model for digital still cameras," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 4, pp. 1135–1143, November 2006.

[31] J.S. Lee, Y.Y. Jung, B.S. Kim, and S.J. Ko, "An advanced video camera system with robust AF, AE, and AWB control," *IEEE Transactions on Consumer Electronics*, vol. 47, no. 3, pp. 694–699, August 2001.

[32] J.H. Lee, K.S. Kim, and B.D. Nam, "Implementation of a passive automatic focusing algorithm for digital still camera," *IEEE Transactions on Consumer Electronics*, vol. 41, no. 3, pp. 449–454, August 1995.

[33] E. Johansson, A. Wesslen, L. Bratthall, and M. Host, "The importance of quality requirements in software platform development - A survey," in *Proceedings of the 34th Hawaii International Conference on System Sciences*, Maui, HI, USA, January 2001, pp. 1–10.

[34] K.H. Kim, "APIs for real-time distributed object programming," *IEEE Computer*, vol. 3, no. 6, pp. 72–80, June 2000.

[35] A.S. Vincentelli and G. Martin, "Platform-based design and software design methodology for embedded systems," *IEEE Design and Test of Computers*, vol. 18, no. 6, pp. 23–33, November-December 2001.

[36] P.G. Paulin, C. Lien, M. Cornero, F. Nacabal, and G. Goossens, "Embedded software in real-time signal processing systems: application and architecture trends," *Proceedings of the IEEE*, vol. 85, no. 3, pp. 419–435, March 1997.

[37] W.C. Kao, C.M. Hong, and S.Y. Lin, "Automatic sensor and mechanical shutter calibration for digital still cameras," *IEEE Transactions on Electron Devices*, vol. 51, no. 4, pp. 1060–1066, November 2005.

[38] D.A. Kerr, *APEX-The Additive System of Photography Exposure*, July 2006.

[39] P. Bigioi, G. Susanu, P. Corcoran, and I. Mocanu, "Digital camera connectivity solutions using the picture transfer protocol," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 3, pp. 417–427, August 2002.

[40] H.C. Lee, *Introduction to Color Imaging Science*. New York: Cambridge University Press, 2005.

3

Digital Camera Image Processing Chain Design

James E. Adams, Jr. and John F. Hamilton, Jr.

3.1 Introduction

The transformation of digital camera raw sensor image data into a full-color fully processed image involves a complex chain of computations. The possible orderings of individual operations and associated implementation details that constitute the image processing chain can lead to a sea of permutations. However, despite this seemingly immense number of available degrees of freedom, the problem of image processing chain design is overconstrained. Image quality must be maximized while compute resource use must be minimized. It is the minimization of required computational effort that, in fact, severely restricts the number of degrees of freedom in the image processing chain design problem. Consequently, image processing operations that are highly effective may not be viable candidates for image processing chain for constrained compute environments. In the end, the process of designing an image processing chain becomes one of taking relatively simple, well-known image processing operations and staging them in a manner that produces the best synergistic effects.

This chapter begins with Section 3.2, which presents in detail a basic image processing chain that will be used as a reference for the balance of the chapter. After a discussion of cost and user sensitivity considerations (Section 3.2.1), the reference path is artificially segmented into four main stages: correcting image errors resulting from the flaws of the sensor hardware (Section 3.2.2), correcting image errors caused by the image capture conditions (Section 3.2.3), creating a standardized processed image (Section 3.2.4), and rendering an output device-specific fully processing image (Section 3.2.5). Each section is further subdivided into component operations such as dark floor subtraction, structured noise reduction, stochastic noise reduction, exposure and white balance correction, color filter array interpolation (demosaicking), stochastic color noise reduction, color correction, tone scale and gamma correction, and edge enhancement.

With the reference path fully presented, a number of variations are discussed in detail in Section 3.3. Section 3.3.1 presents the idea of splitting the chain in part along the lines of independent and interdependent luminance-chrominance image processing paths. Section 3.3.2 follows up with a similar discussion on the merits of splitting the chain along the lines of spatial frequency content. The nature of the computing environment and its impact on the image processing chain design process are discussed in detail in Section 3.3.3. In addition to the topics of internal image data management and manipulation, the section addresses the implementation details for a variety of physical computing environments. Section 3.3.4 is focused on resizing and compression which are two key image processing operations not included in the initial reference path. Finally, other important facts, such as data bit depth resolution, the use of nonlinear photometric spaces, and the issues around extended dynamic range processing, are presented in Section 3.3.5.

The bulk of this chapter is concerned with the problem of processing digital still images. Digital video imaging has been an equally important application. Section 3.4 discusses the significant changes that must be made to the image processing chain in response to this very different imaging environment. Finally, Section 3.5 summarizes the main image processing chain design ideas.

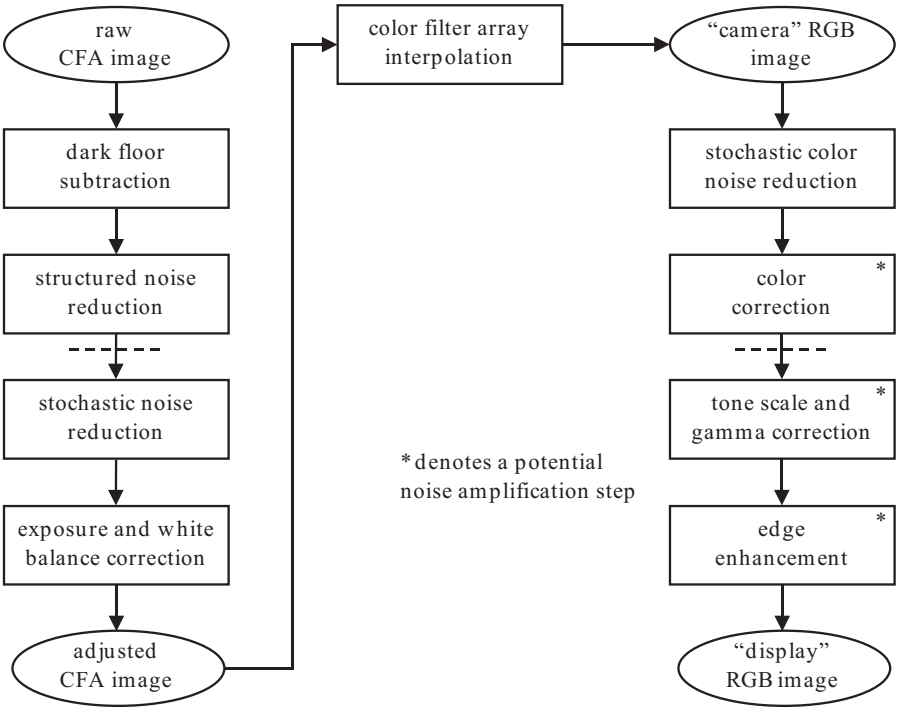


FIGURE 3.1

Reference image processing chain.

3.2 A First Image Processing Path and the Basic Building Blocks

As will be seen, the point of image processing chain design is that there are a number of plausible orderings and configurations. Indeed, each image processing chain designer will have its own preferred and valid chain configurations with supporting justifications. In this regard, any reference to a "standard" image processing path must be taken with a certain grain of salt. Still, it is a useful talking point. To this end, Figure 3.1 presents what will be the reference image processing chain for subsequent discussion. Example images achieved at different stages of the chain are presented in Chapter 1.

3.2.1 Cost and User Sensitivity Considerations

In a system as complex as an image processing chain, there will be a number of engineering decisions that must be made. Many of these decisions will have no objective basis for determining what is "right"; however, many decisions will depend on what is preferred or expected. Consider the camera used for photographing a wedding versus the other camera used to take holiday snapshots at the beach. The *user expectations* (UE) concerning the resulting image quality would be very different. This point will be raised numerous times in the following discussions. The image chain designer must make technical decisions that ultimately will be judged according to the UE of the target customer base.

3.2.2 Systematic Sensor Data Correction

Figure 3.1 begins with the raw color filter array (CFA) image data produced by the sensor. This data will typically be in a nominally linear photometric space. As a result, the pixel values will be directly proportional to scene reflectance in the scene space. The capture bit depth of the pixel values will typically be anywhere from 8 bits to 16 bits, with 8- and 10-bit systems; that is, those that capture 256 and 1024 code values per pixel, respectively, being the most common. With planning (and perhaps some luck), the desired portion of the dynamic range of the scene will fall in the range of valid code values without being significantly clipped.

3.2.2.1 Dark Floor Subtraction

A tacit assumption often made is that a pixel receiving no light will produce a code value of zero. Unfortunately, thermal noise and other nonphoton-generated noise sources will produce nonzero pixel values. As a result, the first course of action is to subtract a *dark floor* from the original CFA image. This can be as simple as applying a fixed value equally to every pixel in the image. Alternately, a spatially dependent set of values can be subtracted. While the nominal goal is to remove unwanted biases in the pixel data, some residual bias may be left in order to avoid data clipping and other quantization errors in the shadow regions.

Certainly, subtracting a single fixed value from the image is the quickest and easiest operation to implement. For a low UE situation, this will almost be an automatic decision by the image chain designer. The only subtleties in this case are deciding how much of the dark floor to subtract and then changing the dark floor as a function of camera exposure index (ISO), shutter time, and temperature. As to the first question, a quick characterization of the image noise near the photometric zero point of the image will provide a mean and standard deviation. Photon noise will exhibit a Poisson distribution, but because other noise sources may be present, a Poisson distribution is not guaranteed. Without doubt, the mean does not want to be clipped from the image data. A safe initial setting might be to subtract a value equal to the mean reduced by one or two standard deviations. These statistics can be empirically determined with either a "lens cap" shot or the capture of a good-quality matte black test card. Alternately, if the sensor has shield pixels around its perimeter, these values can be interrogated at the time of capture. This last approach has the added benefit of characterizing the dark floor at the actual circumstances of the camera at the time of capture; for example, using the information about temperature, ISO, and shutter time. Barring the convenience of shielded pixels in the sensor, a calibration of the camera in the factory will need to be performed to determine nominal values for dark floor subtraction for, at least, the ISO settings the camera supports.

3.2.2.2 Structured Noise Reduction

Unfortunately, at this point we must abandon the notion that the sensor is perfect. The assumption that the dark floor is constant across the entire image capture is a simplification that is valid only if the UE is suitably lax enough. There are many potential causes for why the dark floor may not be uniform across the extent of the sensor. Apart from physical flaws or nonuniformities in the sensor itself, the proximity to heat-producing components in the

camera body may cause one part of the sensor to be warmer than another part. Entering into the world of spatially dependent dark floor correction, the task is now to subtract a *dark floor mask* from the captured image. This mask is created either in the factory or during a dark field shot (i.e., closed shutter) that may be automatically captured during power up of the camera. As discussed above, this mask can be suitably scaled prior to subtraction. One liability of this dark floor mask subtraction operation is that while the *structured noise* in the image will be reduced, the stochastic noise in the image will be increased because of the root mean square adding of variances of the two random variables: the stochastic noise in the captured image and the stochastic noise in the dark floor map. A natural solution to address this increase in stochastic noise is to consider capturing several dark floor masks and subtracting their average from the captured image. Another solution is to fit the dark floor mask with a low-frequency polynomial model and subtract the resulting polynomial from the captured image. This route has the additional advantage that if the map is to be stored in the camera then only the coefficients of the polynomial map need to be recorded rather than data for a full image.

Somewhat implied by the previous discussion is the fact that dark floor subtraction tends to address low-frequency structured noise better than high-frequency structured noise. A significant type of high-frequency structured noise is the defective pixel. It is almost inevitable that a number of the individual pixels in any given sensor will be defective. In fact, whole columns and rows of pixels might be nonfunctional. Even if the sensor is handpicked in the factory to be free of defective pixels, over time cosmic radiation will eventually create defective pixels in the device. One can consider defective pixels falling into two broad categories. First, there are the pixels that are completely and consistently nonfunctional, being "stuck" at complete black or complete white regardless of what light may fall upon them. This is the simpler of the two categories to address. The second category is more problematic because these pixels are only partially defective. They may still respond to light, but with a significantly different gain factor from the majority pixel population of the sensor. They may also only be defective for certain camera settings (e.g., ISO) but function normally for others. When considering this second class of defective pixels, even the determination of how different a pixel must be from the main population before it is considered defective can be problematic. Defective pixels of the first category can be easily mapped out in the factory and their locations stored in the camera firmware. Defective pixels of the second category or those of the first category that are formed after the camera has left the factory are more difficult to address. Pixels that fail to white can be detected using closed shutter captures and subsequent impulse or outlier detection algorithms. Pixels that fail to black are more problematic. Ultimately, one may be forced to treat all unidentified defective pixels with stochastic noise cleaning methods [1], [2], [3].

Some defective pixels can be masked, at least partially, by the dark floor mask subtraction. However, the main method of defective pixel masking is to replace defective pixel values with the average values from known working neighboring pixels. The strategy can be as simple as performing a boxcar average of like-colored pixels over a given region or as complex as performing edge detection and selecting an appropriate directional blur kernel on a pixel-by-pixel basis [4], [5]. Isolated defective pixels are easily dealt with using the simplest of methods. Clusters and whole rows and columns of defective pixels need solutions that are more involved in order to prevent visible artifacts in the final image.

A final class of structured noise to be discussed deals with variations in the thickness of the CFA color filters across the surface of the sensor. These are usually a consequence of (unwanted) interactions of the manufacturing process with the morphology of the silicon wafer. As a result, an image of a featureless neutral field may exhibit low-frequency variations in color. Because this is a stable phenomenon, it can be mapped in the factory and stored in the camera firmware. Either a full-resolution mask can be created or the coefficients of a polynomial fit can be determined. The method of correction is similar to dark floor subtraction, except that each color channel in the CFA will have its own separate mask or polynomial.

3.2.3 CFA Data Correction

The next stage of the image processing chain is concerned with the reduction of stochastic noise and corrections for exposure and white balance errors at the time of capture. Referring to Figure 3.1, the ordering of these two operations is arbitrary and can be inverted without penalty. Unless there are highly unusual interactions present in the individual image processing implementations, these two operations can be considered independent. This is largely because stochastic noise reduction will be mainly concerned with the high-frequency spatial component of the image data while the exposure and white balance correction will be focused on using the low-frequency spatial component to the image. This notion of high-frequency / low-frequency split processing will be revisited below.

3.2.3.1 Stochastic Noise Reduction

It is well known that if a system consists of a number of operations that are signal amplifiers, then it is best to reduce noise contributions as early as possible in the chain. As will be discussed later, many of the image processing operations in Figure 3.1, notably color correction, tone scale and gamma correction, and edge enhancement, are signal amplifiers. This would argue for performing noise reduction prior to these operations. Moving further back along the chain, the nature of the CFA interpolation operation needs to be considered. This operation may or may not be a signal amplifier, depending on the composition of the missing pixel value estimators. In addition, this operation may be linear or it may be adaptive (nonlinear). In the latter case, the robustness of the algorithm's decisions can be significantly influenced by the presence of noise in the CFA image data. Therefore, it seems prudent to perform noise cleaning before CFA interpolation. Finally, we have already noted that there is no strong reason to prefer performing noise reduction before or after exposure and white balance correction.

Performing noise reduction before CFA interpolation presents its own unique set of challenges. Because there is only one color channel value at each pixel location, it becomes difficult to exploit the partial correlation between the color channels in the image. For this reason, Figure 3.1 has a separate stochastic color noise reduction block after CFA interpolation. Consequently, noise reduction before CFA interpolation generally employs the techniques of single-channel grayscale image processing. Conceptually, the CFA image data is split into three or more color channel components by collecting pixels of like color into each component. At this point, each component can be treated as an individual grayscale image and noise-cleaned in any appropriate manner. After noise reduction, the color chan-

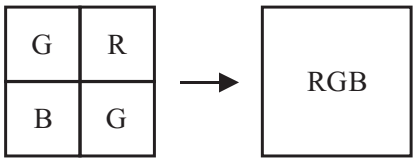


FIGURE 3.2

A 2×2 block of pixels in a Bayer mosaic image forms a single full-color superpixel.

nel components can be merged back into a (now noise-cleaned) CFA image. In practice, the CFA image is apt to be left intact rather than being formally split and merged. Instead, the pixel strides of the noise reduction operations will be adjusted to avoid unwanted mixing of color channels. Typical grayscale noise reduction methods used on CFA data include low-pass filtering, sigma filtering [6], and median filtering, although potentially any noise reduction scheme appropriate to grayscale imaging could be successfully used. The effect of noise filtering on camera images is illustrated in Chapter 1 while joint denoising and demosaicking solutions are discussed in Chapter 9.

There is a subtle challenge not to concede on addressing color channel correlation at this point of the image processing chain. While addressing correlation at the full sensor resolution must wait until after CFA interpolation, lower spatial frequency color correlation can be conditioned beforehand. Dividing the CFA image into a lower resolution array of superpixels (Figure 3.2) produces a full-color image than can now be treated with a larger set of noise reduction techniques. The discussion of these techniques will be postponed until later in the chain.

3.2.3.2 Exposure and White Balance Correction

Unlike the human visual system (HVS) that constantly and automatically adjusts the apparent exposure and white point of what we see, digital cameras have no such innate functionality. Therefore, such adjustments must be made algorithmically after image capture. The goal of such algorithms is to render neutral areas in the scene as regions of equal code values for all color channels in the final image. Additionally, midlevel grays (18% scene reflectance) should also map to mid code value range of the final image. Sometimes the processes of exposure correction and white balance correction are referred to collectively as *scene balance correction*. Detailed descriptions of exposure correction and white balance correction can be found in Chapters 10 to 12.

As with noise reduction, these adjustments can be grouped into two categories. The first category consists of adjustments in response to user input. In the direct case, although an unlikely one, the user can specify a particular exposure compensation (e.g., 1.33 stops) and also specify the type of scene illuminant (e.g., daylight or tungsten). With this explicit information, the CFA image data can be directly modified in accordance with the user input. For exposure compensation, all pixel values would be equally modified by the appropriate scale factor. For white balance correction, there would be a set of three different scale factors, one for each color channel. This white balance triplet would be characteristic of the given scene illuminant. In the more likely indirect case, a simpler input mechanism for

the user would be to click on a region in the image that is known to be neutral in color and of a middle exposure level. The algorithm can then interrogate that region of the image to determine the scale factors necessary to drive it, along with the rest of the image, to a desired code value position.

The second category of exposure and white balance adjustment is far more difficult. User input is not available. All the algorithm has is the image data itself. This is the realm of automatic exposure and white balance correction algorithms [7], [8], [9]. This is an ill-posed and unfair problem! One is asked to find the proper adjustment for a midlevel gray region that does not exist in the image. Most approaches are based on heuristic statistical models such as the gray world hypothesis. This hypothesis contends that all images, taken as a single set of pixels, average to approximately 18% scene reflectance gray (neutral). Unfortunately, this says nearly nothing about the statistics of any given single image. Slightly more robust statements can be made by restricting the application of the gray world hypothesis to only those parts of the image that lie along the edge and feature boundaries. Thus, large regions of sky or grass will not dominate the scene average. Using this and other heuristic rules, a set of code values (usually a triplet) representing 18% gray for a given image are computed. At this point, the calculation proceeds as before in the first category of exposure and white balance adjustment.

As an example of additional heuristics that can be applied, consider the problem of correcting an image with a warm (red) cast. This image could be an indoor scene at home under tungsten, or an outdoor scene in one's backyard of a sunset. In the first case, the reddish color should be rendered as neutral (i.e., more or less fully corrected) because that is how one perceives the scene. In the second case, the reddish cast should be preserved because sunsets *look* red. A heuristic that can help distinguish between these two cases is the overall brightness level that can be found from metadata describing the aperture setting, shutter time, and exposure index setting at the time of capture (a sunset is typically much brighter than an indoor light bulb) [10]. Chapter 13 describes camera image storage formats and associated metadata in detail.

Finally, it is noted that the computations just described can be done on spatially very low-resolution data. It may even be preferred to use low-resolution image data to improve the performance of the heuristic rules. This has the advantage of making the computations relatively impervious to noise and detailed scene content and composition. In addition, a small data set is computationally more tractable, allowing the use of more involved heuristic systems without undue execution time penalties.

3.2.4 Adjusted CFA Image and Image Data Calibration

At this point, we have conditioned the CFA image data to be reduced in noise, both structured and stochastic, and to represent an image that has been properly scene balanced. Most subsequent image processing operations in the chain tacitly assume these idealized conditions. It will be the responsibility of the later image processing operations to address any residual departures from this state.

The next task is to create a full-color image and then convert that image into a known, calibrated color space. Along the way, the aforementioned residual noise and scene balance errors will need to be addressed.

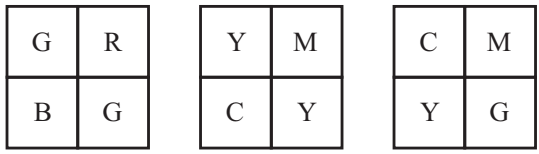


FIGURE 3.3

Minimum repeating units: (a) RGB Bayer CFA, (b) CMY Bayer CFA, and (c) hybrid CMYG pattern.

3.2.4.1 Color Filter Array Interpolation

The basic premise underlying *CFA decimation* and subsequent interpolation or demosaicking is that full-color image data is high in redundant information. Some sense of this, perhaps, can be gained by noting that spatial detail as perceived by the human visual system is predominantly based on luminance information in the scene and only to a much smaller extent on chrominance information (see page 380 in Reference [11]). Beginning with color television, most electronic imaging systems have taken advantage of this fact by recording chrominance information at a lower spatial frequency than luminance. The savings in reduced information bandwidth more than compensates for any small loss in image fidelity.

Consequently, most CFAs used for image capture subsample chrominance with respect to luminance in two ways. First, only one of three (or more) color channels is recorded at each pixel location. Second, in the color filter array *minimum repeating unit* (MRU) there are more filters assigned to sensing luminance information than there are for sensing chrominance information. The prototypical CFA is the Bayer pattern [12]. In the RGB case (Figure 3.3a), the green pixels are used to sense luminance information, and the red and blue pixels are used to sense chrominance information. In the CMY case (Figure 3.3b), the yellow pixels are used to sense luminance, and the cyan and magenta pixels are used to sense chrominance. The complexity can be further increased by adding a fourth color and reading out sums and differences of adjacent pixels to produce luminance-chrominance information. In the CMYG-based MRU (Figure 3.3c), it is common to read out two luminance (C + Y, M + G) and two chrominance (C - Y, M - G) values [13], [14].

Regardless of the CFA used, a full-color image must be produced at this point in the image processing chain. Full-color in this case means each pixel in the image has a color specification triplet. There are two general approaches to the problem of CFA interpolation. The first is to use standard linear interpolation methods. The most common approach is to combine neighboring pixel values of the same color in some straightforward method to produce an estimate for the missing pixel value. This method can take the form of a convolution operation and implement such standard practices as pixel replication, bilinear interpolation, or bicubic interpolation. If, on the other hand, there is some understanding of the cross color channel correlation of the data, more than one color may be used in this process. This latter approach is most readily accomplished by first interpolating all of the luminance pixel data and then forming color differences between the luminance and chrominance pixel values (e.g., R - G and B - G). These color difference values are then interpolated and the resulting chrominance values recovered by adding the luminance values back to the interpolated color difference values [15].

The second approach to CFA interpolation is to use nonlinear adaptive methods. These are generally heuristic in composition as they fall out of the realm of linear shift-invariant systems. With these systems, the segmentation of image data into luminance and chrominance channels becomes more important because the decisions made by the algorithm are generally keyed off the fine spatial detail in the image. As a result, the luminance channel is interpolated first by using some form of edge detection of the luminance data to determine the precise manner of interpolation from pixel to pixel. Sometimes these algorithm decisions are made immediately and irrevocably [16], [17], and sometimes the decisions are revisited and revised after their consequences are perceived later in the algorithmic process [18], [19]. Once the luminance channel is fully populated, the chrominance channels are generally treated with the linear approaches previously described, although references will be found in the literature to adaptive methods used in place of those approaches [17]. Chapters 1 and 5 to 9 discuss demosaicking issues in detail.

The result of the CFA interpolation process will be a "camera" full-color image. The color space will generally not be a standard, calibrated space. Instead, it will be defined by the spectral sensitivities of the camera image capture hardware (detector quantum efficiencies and CFA spectral responses being the biggest players). Before addressing this issue, an intermediate operation is discussed.

3.2.4.2 Stochastic Color Noise Reduction

During the stochastic noise reduction applied to CFA image data, the color channels were treated as separate and independent grayscale channels. In each channel, looking at artifacts seen in a flat field, stochastic noise gives the impression of "visual static" or "graininess" in what should be a smooth region of the image. Sufficiently reducing the visibility of the noise leaves one with the subtle fluctuations of a texture that looks "real" instead of artificial. Now that all the color channels are fully populated, another facet of stochastic noise emerges. A texture that might be acceptable in the context of a single-channel image is deemed not acceptable when matched with similar, but different, textures in the other color channels. Instead of producing a light-dark texture, residual stochastic noise in an RGB image produces unexpected *color* variations. This effect is most pronounced in neutral (gray) regions because the color fluctuations include pastels of widely divergent hue angles. Such color artifacts are far more visible and objectionable than the corresponding light-dark fluctuations in a single-channel image. The color aspect of stochastic noise reduction is now addressed.

The good news is that at this point the image data is in a well-known and well-behaved representation. Methods abound in the literature on how to noise-clean fully populated color images. The simplest approach may be to, again, treat each color channel as an independent grayscale image and then clean these components separately. However, this may not be overly effective and tends to miss the whole point. It is far better to transform the image into a luminance-chrominance representation (assuming it is not already so). At this point, the luminance and chrominance channels can be noise-cleaned in any appropriate manner. Generally, the luminance data will require a significantly different cleaning modality from that used for the chrominance data. If the same method is used, at least the tunings of the operation will be quite different.

Because this is a color noise reduction operation, the luminance channel may be left completely untouched. It is still useful as a reference for use with adaptive chrominance channel noise-cleaning operations. The chrominance channels, if sufficiently devoid of spatial edge detail, can be cleaned most simply by a low-pass (blurring) convolution operation. If the presence of low-frequency color blobs is evident, then rather than use a large convolution kernel, the chrominance channels can be decomposed into a Gaussian or wavelet pyramid. The individual pyramid components can then be convolved with smaller kernels and a noise-cleaned image reconstructed from the processed components. If there is a desire to preserve the residual edge detail in the chrominance channels, then adaptive noise-cleaning methods such as sigma filters or steerable median filters can be used. When using such adaptive methods, the luminance channel can often be used in the edge detection operations, providing improved robustness over the lower modulation edges in the chrominance channels.

If there is a reason, the luminance channel can also be noise-cleaned at this time using any method applicable to single channel grayscale images.

3.2.4.3 Color Correction

It is now time to prepare the image for its final rendered form. This requires transforming the image into a standard calibrated color space. There are a number of possible destination color spaces. Of these, the industry has standardized on *sRGB* [20], which is designed for video, or soft display, devices. However, because sRGB is itself a color transform from CIE 1931 XYZ space, the latter can be considered the first target of the color correction operation.

CIE 1931 XYZ space (see pages 101 to 110 in Reference [11]) is a color space defined by standardized $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$ color matching functions. CIE 1931 XYZ space itself is a color transform from the previously defined CIE RGB color space, although there is generally no reason to explicitly invoke that relationship in today's digital cameras. The first part of the color correction process is to transform the image data from camera color space into CIE 1931 XYZ space. Assuming an RGB camera color space, the operation becomes a 3×3 matrix multiply:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} R_{\text{camera}} \\ G_{\text{camera}} \\ B_{\text{camera}} \end{bmatrix} \tag{3.1}$$

The coefficients of the transformation matrix are computed in the factory through a regression process using measured camera RGB tristimulus values of color patches with known XYZ tristimulus values.

Once the XYZ tristimulus values have been computed, they can be transformed to sRGB tristimulus values with a standard matrix as follows:

$$\begin{bmatrix} R_{\text{sRGB}} \\ G_{\text{sRGB}} \\ B_{\text{sRGB}} \end{bmatrix} = \begin{bmatrix} 3.2410 & -1.5374 & -0.4986 \\ -0.9692 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.2}$$

Combining Equations 3.1 and 3.2 produces the color correction matrix as would be imple-

mented in the image processing chain as follows:

$$\begin{bmatrix} R_{\text{sRGB}} \\ G_{\text{sRGB}} \\ B_{\text{sRGB}} \end{bmatrix} = \begin{bmatrix} 3.2410 & -1.5374 & -0.4986 \\ -0.9692 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} a_{11}\ a_{12}\ a_{13} \\ a_{21}\ a_{22}\ a_{23} \\ a_{31}\ a_{32}\ a_{33} \end{bmatrix} \begin{bmatrix} R_{\text{camera}} \\ G_{\text{camera}} \\ B_{\text{camera}} \end{bmatrix}$$
$$= \begin{bmatrix} b_{11}\ b_{12}\ b_{13} \\ b_{21}\ b_{22}\ b_{23} \\ b_{31}\ b_{32}\ b_{33} \end{bmatrix} \begin{bmatrix} R_{\text{camera}} \\ G_{\text{camera}} \\ B_{\text{camera}} \end{bmatrix} \tag{3.3}$$

It should be noted that up to this point it has been tacitly assumed that all computations that have been performed in the image processing chain have been done in linear space. While the use of nonlinear spaces for these computations will be discussed below, the color correction computations just presented are explicitly designed for linear space data. The case of performing color correction on nonlinear space data will also be addressed below.

3.2.5 Image Space Rendering

The remaining steps in the image processing chain are targeted at producing the best image for a given image rendering. In keeping with current industry standards, this means preparing the image for display on a video device. The transformation to sRGB tristimulus values has already begun this process. The process is completed by transforming the image into a nonlinear space suited for video display devices and applying edge enhancement (i.e., sharpening).

3.2.5.1 Tone Scale and Gamma Correction

The human visual system's ability to adapt to a wide range of scene luminances is another essential capability that the digital camera must duplicate, if only in a primitive manner. In this case, the overall contrast of the scene must be adjusted so that the image as viewed on the soft display device looks similar to the original scene viewed under illumination that was typically a hundred times as bright, if not more. Added to this, the image data must be transformed to account for the nonlinearity of the video display. As in the case of color correction, the tone scale and gamma correction operation is implemented as a single transform composed of these two components.

The tone scaling operation adjusts the contrast of the image. It is usually implemented as a fixed lookup table that is applied equally to the red, green, and blue channels. It assumes the input data is in a linear space that has been properly exposure corrected. There are two general classes of tone scale transforms. The first class consists of fixed transforms that are installed in the factory and used on all images. There may be a single transform or a small family of fixed transforms, with each family member assigned to a different exposure compensation step, such as ± 2, ± 1, and 0 stops. The shape of the fixed transform curve is typically "S" shaped (Figure 3.4a) with a slope greater than unity in the middle of the input code value range and considerably less than one at the two extremes; that is, the shadows and the highlights [11]. The visual impact of applying such a curve is to increase the overall contrast of the image. Note that there is no reason that the tone scale need be symmetric in its handling of the shadows and the highlights. In order to reduce the visibility of noise in dark regions of the image, a tone scale may apply more aggressive

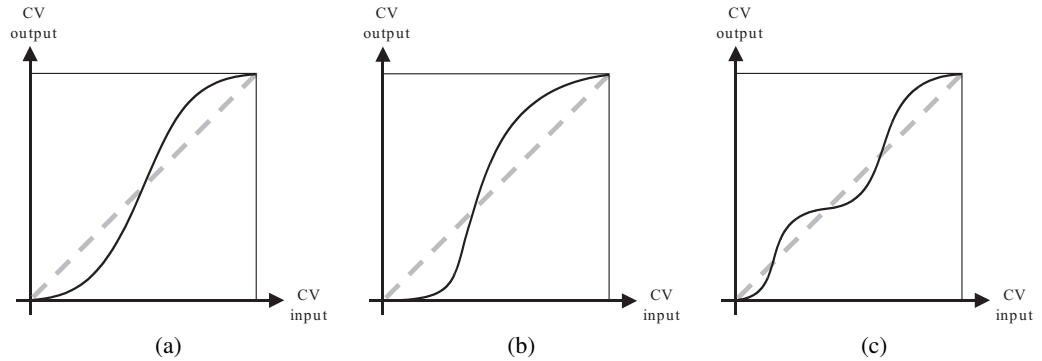


FIGURE 3.4

Tone scale functions: (a) idealized function, (b) function with suppressed shadow response, and (c) scene-specific function.

compression of shadows than highlights (Figure 3.4b). The tone scale is initially generated by the image processing chain designer based on the characteristics of the digital camera system and customer preference; for instance, professional photographers tend to prefer a lower contrast than do consumer snap shooters.

The second class consists of tone scale transforms that are generated dynamically on an image-by-image basis. Returning to the HVS, its *dynamic range* is significantly greater than any current digital camera. On a sunny day, a person can look into a blue sky and see puffy clouds, then immediately look into deep shadows and see well-defined image detail. However, a digital camera with a fixed tone scale will be forced to either saturate the sky to pure white to render details in the shadows or clip the shadows to complete black to preserve the sky. A partial solution to this dilemma is to create a custom tone scale that renders both the shadows and the sky (highlights) at the expense of the midtones, which are visually less important in such high dynamic range scenes. Figure 3.4c shows an example of such a tone scale.

There are many ways of algorithmically producing such a tone scale based on histogram analysis of the image [21], [22]. Regardless of how the tone scale is generated, it ultimately becomes a simple point transform of the image data:

$$\begin{cases} R'_{\text{sRGB}} = T\left(R_{\text{sRGB}}\right) \\ G'_{\text{sRGB}} = T\left(G_{\text{sRGB}}\right) \\ B'_{\text{sRGB}} = T\left(B_{\text{sRGB}}\right) \end{cases} \tag{3.4}$$

The second transformation to be applied is *video gamma correction*. This standard transform is also defined in the sRGB specification [20] and accounts for the fundamental photometric nonlinearity of the cathode ray tube (CRT) display. This transform is essentially a simple power relationship:

$$X''_{\text{sRGB}} = \begin{cases} 12.92X'_{\text{sRGB}} & \text{for } X'_{\text{sRGB}} \leq 0.00304 \\ 1.055X'^{(1/2.4)}_{\text{sRGB}} - 0.055 & \text{for } X'_{\text{sRGB}} > 0.00304 \end{cases} \tag{3.5}$$

where X'_{sRGB} is R_{sRGB}, G_{sRGB}, or B_{sRGB} normalized to $[0,1]$. The output is also in the range $[0,1]$ and can be subsequently scaled to any conventional data range, usually $[0,255]$.

Equation 3.5 is a point transform and can be concatenated with the tone scale correction to produce a final single point transform to perform both operations simultaneously:

$$X''_{\text{sRGB}} = V\left(X'_{\text{sRGB}}\right) = V\left(T\left(X_{\text{sRGB}}\right)\right) = G\left(X_{\text{sRGB}}\right) \tag{3.6}$$

In this expression, $V(\cdot)$ is the video gamma correction, $T(\cdot)$ is the tone scale correction, and $G(\cdot)$ is the composition correction, usually referred to casually as the gamma correction.

The systematic construction of the composite gamma correction as described here is frequently shortcut in favor of simply beginning with the sRGB video correction (Equation 3.5) and then customizing this transform until the average image has the desired contrast, dynamic range rendering, and shadow noise suppression. Such an approach is generally heuristic in nature and usually targeted at generating a fixed tone scale transform to be used for all images.

3.2.5.2 Edge Enhancement

The final block in the "standard" image processing chain is edge enhancement, more casually referred to as sharpening. The essential purpose of this operation is to amplify the high-frequency spatial components of an image to make it look sharper. Because noise has also high-frequency characteristics, attention must be given to the question of controlling noise amplification during edge enhancement.

The two main approaches to edge enhancement are *direct convolution* and *unsharp masking*. Actually, these operations are two sides of the same coin, as they produce mathematically equivalent results when confined to the world of linear shift-invariant systems. The direct convolution method consists of extracting a high-frequency record from the image via convolution with a high-pass kernel. Some scaled amount of this high-frequency record is then added back to the original image to produce the sharpened result as follows:

$$\mathbf{A}' = \mathbf{A} + k\left(\mathbf{A} * \mathbf{h}\right) \tag{3.7}$$

where $\mathbf{A}$ is the original image, $\mathbf{h}$ is the high-pass convolution kernel, k is a scale factor, and $\mathbf{A}'$ is the resulting sharpened image. In the case of unsharp masking, the high-frequency record is created by computing the difference between the image and a blurred (low-pass) version of itself:

$$\mathbf{A}' = \mathbf{A} + k\left(\mathbf{A} - \mathbf{A} * \mathbf{b}\right) \tag{3.8}$$

where $\mathbf{A}$ is the original image, $\mathbf{b}$ is the low-pass convolution kernel, k is a scale factor, and $\mathbf{A}'$ is the resulting sharpened image. From Equations 3.7 and 3.8 it can be seen that $\mathbf{h}$ and $\mathbf{b}$ are related by $\mathbf{h} = \mathbf{I} - \mathbf{b}$ where $\mathbf{I}$ is the identity matrix. In either approach, adjusting the scalar k will adjust the amount of sharpening applied to the image.

In order to control noise amplification during the edge enhancement process, the high-frequency record needs to be noise-cleaned in some manner prior to being added back to the original image. The initial impulse (no pun intended) may be to use a standard noise-cleaning operation. However, high-frequency image data is zero mean and largely devoid of low-frequency information, so low-pass filtering begins to lose its meaning. Instead, rather than performing a spatial noise-cleaning operation, an amplitude noise cleaning method is usually employed. A *coring function* is used to noise-clean the high-frequency record in

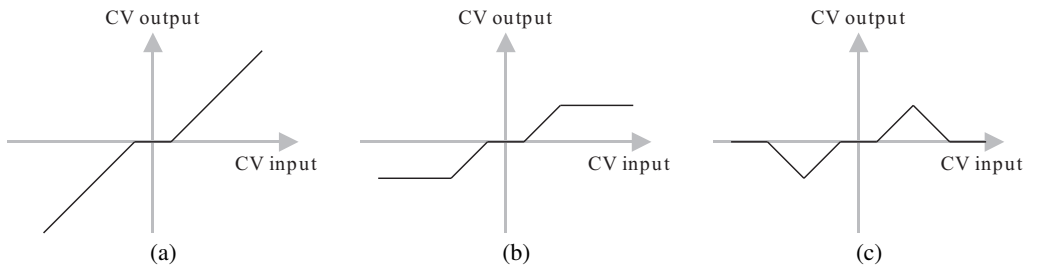


FIGURE 3.5

Example coring functions. CV stands for a code value.

this manner [23], [24]. This function is a point operation that, like the previously described tone scale correction, is usually implemented as a lookup table. Modifying Equations 3.7 and 3.8, the coring operation, $C(\cdot)$, can be added:

$$\mathbf{A}' = \mathbf{A} + kC(\mathbf{A} * \mathbf{h}) \tag{3.9}$$

$$\mathbf{A}' = \mathbf{A} + kC(\mathbf{A} - \mathbf{A} * \mathbf{b}) \tag{3.10}$$

Note that if $\mathbf{h} = \mathbf{I} - \mathbf{b}$, Equations 3.9 and 3.10 are still mathematically equivalent.

The shape of the coring function (Figure 3.5) is heuristically determined based on the fundamental noise characteristics of the digital camera system. Nominally, small amplitude values in the high-frequency record are suppressed to zero to reduce amplifying noise in flat regions of the image. Midrange amplitude values in the high-frequency record are left essentially unaltered. There is no clear consensus on how to modify large amplitude values in the high-frequency record. An entire range of possibilities can be found in practice: leave them unchanged, clip them to some maximum value, or beyond a certain input amplitude value begin to reduce the size of the cored amplitude so that the largest amplitude values in the high-frequency record are actually set to zero. The selection criteria for the shape of the coring function is a tradeoff between noise amplification, overall image sharpness, and image distortions such as the loss of three-dimensionality of strong edges.

Once the mechanisms of edge enhancement have been determined, attention can be turned to determining precisely which components of the image will be sharpened. As stated before, most of the fine spatial detail in the image is contained in the luminance component. Therefore, it is plausible to split the image into luminance and chrominance components, sharpen just the luminance channel, and then merge the components back into a sharpened image. If luminance-chrominance space is the final destination of the image, for instance, in preparation for Joint Photographic Experts Group (JPEG) compression, then this strategy is practical, as well. However, if sRGB is the final destination space, a simpler route is available.

The luminance channel, for the purposes of edge enhancement, can be adequately approximated by the green channel [23], [24]. Thus, a scaled, noise-cleaned (cored) high-frequency record is produced directly from the green channel and then added equally to the original red, green, and blue color channels of the image (Figure 3.6). The additional benefit of this approach is that the green channel of a digital camera system tends to be the least noisy of the color channels.

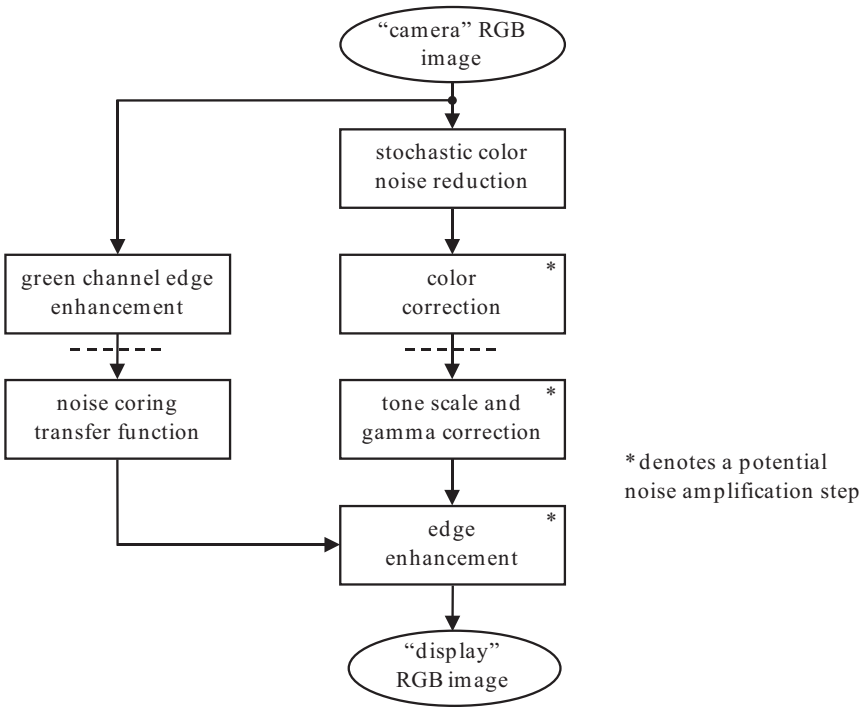


FIGURE 3.6

RGB edge enhancement.

3.3 Variations on the First Image Processing Path

It has been suggested a number of times that there are alternate and possibly better ways of configuring the reference image processing path. Many competing pressures will generally drive the ultimate configuration for a given application. One of the leading determiners is the available computing environment. From a simplistic perspective, this can be considered to be how many pixels can be processed per unit time and what the overall allowable execution time boundaries are. Complicating this interpretation is the question of sharing compute resources with other nonimaging tasks that the device, for instance, a mobile phone, may perform. In addition, an individual microprocessor may have idiosyncrasies that cause some simpler types of operations to execute slowly while other more complex types of operations execute quickly. As a result, the reference path just presented may have to be significantly modified. Section 3.4 will discuss this in some depth.

Sometimes image quality requirements, as established by the UE, require better results than can be easily achieved by simply improving the individual components of the image processing chain. The current trend towards producing acceptable images at higher and higher exposure indices (ISO ratings) is one example. As a first approach, additional noise cleaning operations may be added to the image processing chain, for instance, after the color correction or edge enhancement steps. As suggested under edge enhancement, described above, a similar noise reduction can be achieved not by formally adding more

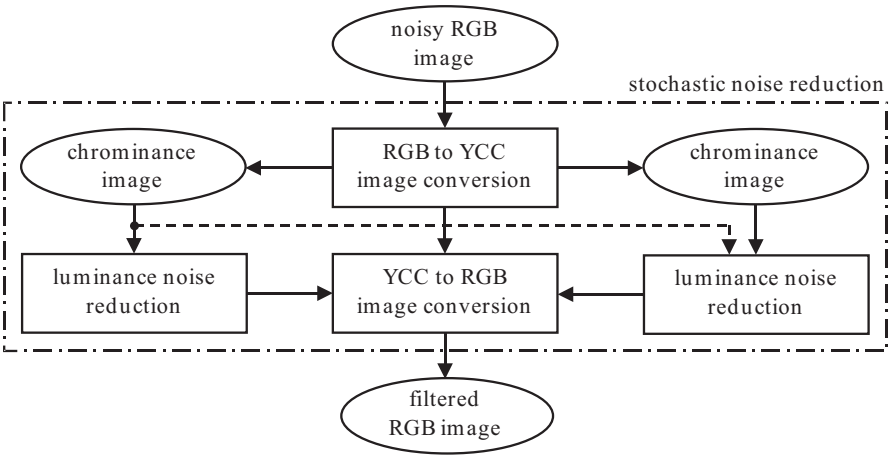


FIGURE 3.7

Luminance-chrominance stochastic noise reduction.

computations but by modifying the chain so that certain image components are processed differently in order to prevent unwanted noise amplification.

3.3.1 Luminance-Chrominance Processing

Under stochastic color noise reduction, the concept of transforming a primary color space (e.g., RGB) into a luminance-chrominance space (YCC) provided an opportunity for improved image processing results [23]. Figure 3.7 illustrates the concept.

The noisy RGB image is initially converted to YCC space. The luminance and chrominance components are now reduced in noise using methods appropriate to the type of data. Namely, luminance noise-cleaning must preserve edges and fine spatial detail, whereas chrominance noise-cleaning generally does not need to be burdened with such concerns. Figure 3.7 shows that luminance information may be used to assist the chrominance noise-cleaning process. After noise reduction, the noise-cleaned YCC components are recombined and converted back into RGB space.

The concept of YCC image processing can be extended over larger parts of the image processing chain. One classic example, as already discussed, is with edge enhancement [24]. In Figure 3.6, a side branch has been added to the image processing chain. After CFA interpolation, a copy of the luminance channel (which could be just the green channel) is routed around the stochastic color noise reduction and the color correction operations. Instead, it is fed directly into the tone scale and gamma correction operation and then the edge enhancement block. This results in a less noisy edge enhancement boost record, having avoided the noise amplification inherent in the color correction step.

3.3.2 Spatial Frequency Processing

Just as difference color channel components (e.g., luminance and chrominance) benefit from different image processing, so will the different spatial frequency band components in

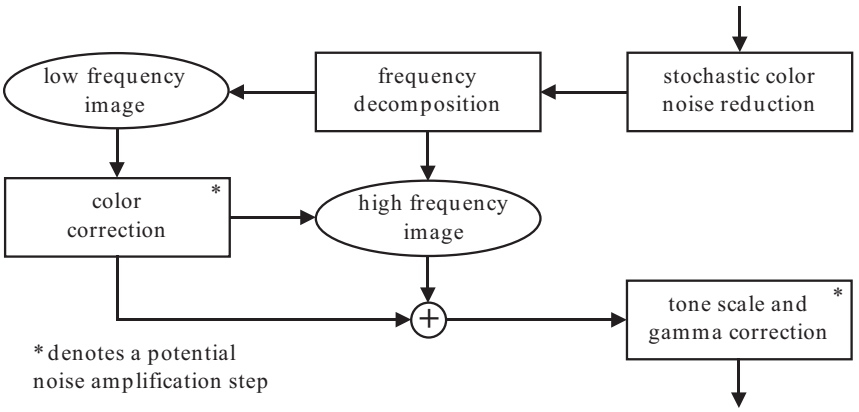


FIGURE 3.8

Color correction of the low-frequency spatial image component.

the image. This is a very powerful concept that has been exploited in a number of ways as described in the literature. The simplest approach is to split the image data into two spatial frequency bands: low-frequency and high-frequency. This splitting method has already been discussed under edge enhancement. In the current discussion, the image processing chain will once again be bifurcated around the color correction operation with only the low-frequency component being color corrected (Figure 3.8) [25].

The result of this image processing chain modification is that high-frequency noise will not be amplified by the color correction step. Because most of the color information in the image is carried by the low-frequency component of the image data, there is little, if any, visual penalty for not color correcting the high-frequency component. Noise-cleaning benefits from the same strategy (Figure 3.9).

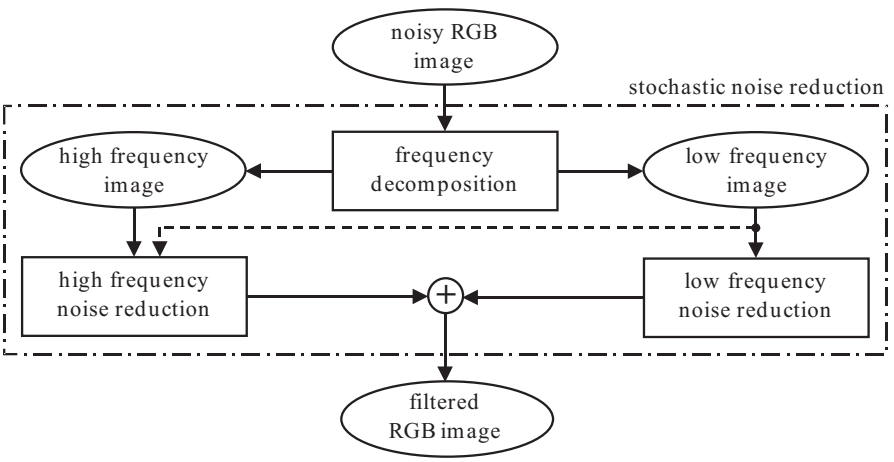


FIGURE 3.9

Low-frequency / high-frequency stochastic noise reduction.

A single low-frequency / high-frequency split of the image data allows different strategies to be used on each component. Typically, most of the objectionable high-frequency noise will be segregated into the high-frequency component leaving a relatively clean low frequency image component. Therefore, the low-frequency component can be left untouched, thus preserving the fidelity of its image content, and noise reduction performed on only the high-frequency component. As indicated in Figure 3.9, the low-frequency component can optionally be used to help drive the high-frequency noise reduction process.

There is no reason to stop at a two-component spatial frequency split. Full wavelet or Laplacian / Gaussian pyramid decompositions can also be used to great effect [26]. In these approaches after the initial split, the low-frequency component is split again into higher and lower spatial frequency components. This splitting continues until the resulting component sizes have reached some useful lower limit in size. Each component or partial reconstruction may now be processed in a custom way. Candidate operations for such custom processing are noise reduction, color correction, tone scale and gamma correction, and edge enhancement. The obvious liabilities of pyramid decomposition / reconstruction approaches are that they stress the capabilities of the compute environment with the proliferation of components, and the number of degrees of freedom mushroom with each new pyramid level, making optimum and robust tunings of such systems sometimes problematic. If there is sufficient memory available, the image can be decomposed into a pyramid representation after CFA interpolation and then reconstructed back into a single image until after edge enhancement. In the interim, each component can potentially follow an individualized image processing path best suited to its nature.

3.3.3 Computing Environments

As would be expected, the computing environment strongly dictates the nature of the image processing chain. The amount of available memory for storing both image data and intermediate results, the nature of the processor's design, and just the fundamental speed of the processing unit are just some of the significant factors that must be taken into account. Rather than trying to address all of the issues that can be largely of a computer engineering nature, only topics directly associated with image processing chain design will be discussed.

3.3.3.1 Intermediate Data Storage

There are three data structures generally used in image processing chain implementations: full frame, line buffer, and tile buffer.

• *Full-Frame Processing*
Full-frame processing is conceptually the simplest way to implement an image processing chain. The entire raw image is read into memory at the beginning of the image processing chain. Each operation in the chain then operates sequentially as an independent entity on the image data in memory. At the end of the chain, the entire image is written to storage.

Color images are generally stored in one of three ways in full frame processing. Perhaps the most common is *pixel interleaved* in which pixel value triplets are stored sequentially (see Figure 3.10 and Figure 3.11).

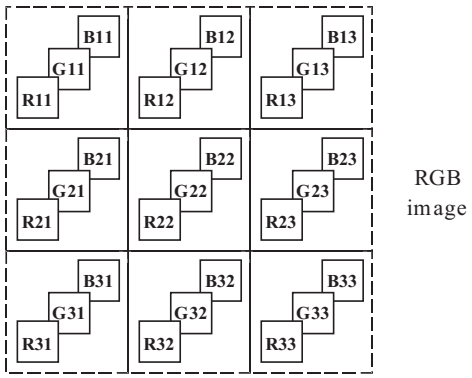


FIGURE 3.10

Original image data prior to intermediate data storage.

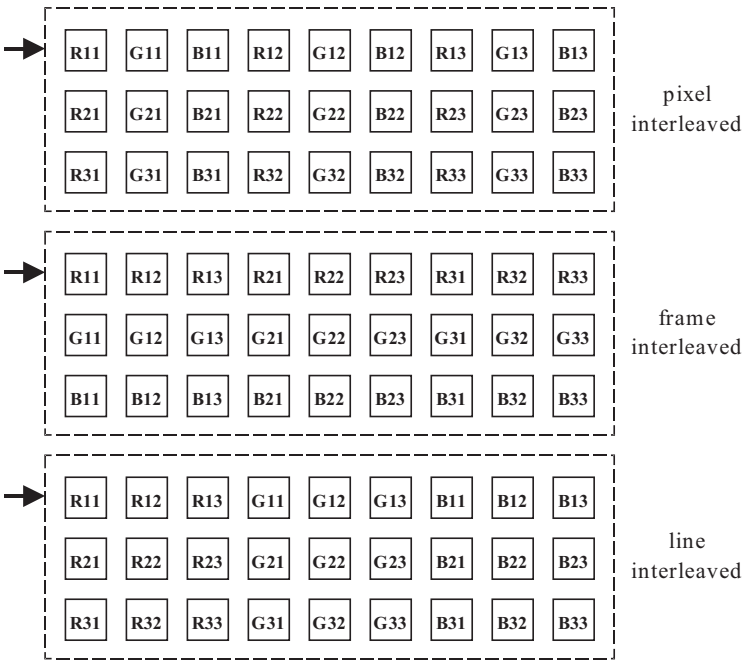


FIGURE 3.11

Image data from Figure 3.10 arranged in (top) pixel interleaved, (middle) frame interleaved, and (bottom) line interleaved formats.

This method has the advantage of keeping all pixel values associated with a given support region (pixel neighborhood) in a smaller region of the full-frame buffer. The second most common format would be *frame interleaved* in which all pixels of a given color are stored contiguously in memory. Each color channel memory block itself may or may not be stored contiguously in memory. This is perhaps the most intuitive arrangement of the image data, especially when applying independent grayscale operations to each of the color channels.

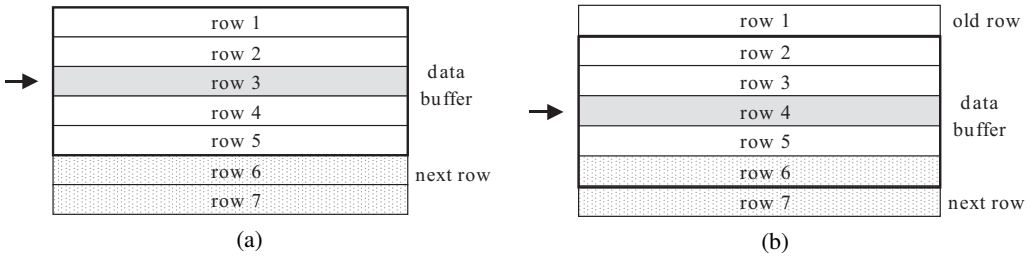


FIGURE 3.12

(a) Producing one row of fully processed data (row 3) from a line data buffer. (b) Data buffer has been rolled from the image on the left. Row 4 may now be fully processed.

The third data format is *line interleaving*, in which each line of color image data is stored as separate lines of each of the individual color channels. This last approach does not see as much application as the other two formats.

The key element that differentiates full-frame processing from the other two approaches is that the *entire image* is available to each of the image processing operations in the chain. This makes operations such as a full in-place wavelet pyramid decomposition possible. It also restricts issues associated with support region (pixel neighborhood) boundary conditions to the physical edges of the image. As an extension of this idea, all intermediate results (e.g., an edge map) are also stored completely in memory in full-frame buffers and the entirety of such intermediate results are available to each image processing operation.

The downside of full-frame processing is its use of large amounts of memory and the associated *memory cache misses* that are incurred. Even if a computing environment has sufficient RAM to hold all of the full frame image buffers, the microprocessor has only a limited amount of *cache memory* that it can directly manipulate in a rapid manner [27]. If image data is required that is currently not in the cache, then a relatively time-consuming memory swapping process must occur to store the current cache contents in slower RAM and bring the required data into the faster cache memory. Using relatively large pixel neighborhoods can quickly slow down the image processing chain attributed to the preponderance of memory cache misses.

• *Line Buffer Processing*

Line buffering was created in order to address the memory cache limitations. In line buffering, only enough lines of image data are read into memory at a time sufficient to produce one fully processed output line of image data, which is subsequently sent to storage (Figure 3.12a). Once the output line is written, the line buffer is *rolled* by the one line so that a new line of input data can be read into memory (Figure 3.12b). This process is continued until the entire image is processed.

Conceptually, in the rolling process one can think of image data physically being copied from a lower row to a higher row. In practice, a set of line data pointers would be rolled instead to avoid a large number of unnecessary data transfers.

The immediate advantage of line buffering is that far less memory is consumed, as only a few lines of the image are ever resident. This, in turn, reduces the tendency to incur memory *paging* (i.e., storing and retrieving blocks of memory from external storage) in

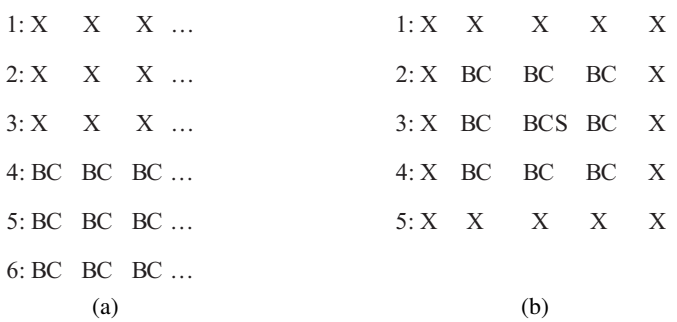


FIGURE 3.13

(a) Example line buffer processing content. (b) Example tile buffer processing content.

physical memory constrained systems [27]. The notable and significant downside is that the complexity of the image processing chain implementation increases substantially. Point processes, such as color correction, are unaffected by using a line buffering strategy. Area processes, such as any convolution operation, can become a bookkeeping exercise.

Consider Figure 3.13a. As a simplified example, the task is to produce a line of processed image data that is blurred with a 3×3 low-pass kernel, then color corrected, then sharpened with a 3×3 high-pass kernel. First, in order to perform a convolution with a 3×3 kernel a minimum of three lines of image data must be present. Therefore, three rows of unprocessed pixels (X) are read into memory lines 1 to 3. (Pixel interleaving is assumed but not explicitly shown.) This allows the blurring operation (B) to be performed and the results stored in memory line 4. The color correction operation (C), being a point operation, can now be performed in place on line 4, producing pixels marked with both B and C. At this point, the unprocessed pixel buffer (lines 1 to 3) is rolled so that line 2 is written to line 1, line 3 is written to line 2, and a new row of unprocessed pixels is written to line 3. (Again, in practice only the pointers to the memory lines would be changed.) At this point, a new blurring and color correction set of operations can be performed and the results stored in memory line 5. Lines 1 to 3 are rolled again and subsequently blurred and color-corrected to populate memory line 6. Now there are enough rows of blurred and color-corrected pixels to perform the sharpening operation, resulting in a finished row of blurred, color-corrected, and sharpened pixels that can be written to the output storage. At this point in the cycle, a steady-state situation has been achieved. In order to produce the next row of finished pixels, the unprocessed pixel buffer need only be rolled one row, the intermediate blurred and color-corrected line buffer also need only be rolled one row, and the new pixel values computed. In a similar manner, a more elaborate image processing can be analyzed and the corresponding line buffering requirements and sequencing of operations determined. A key observation in this regard is that as the number of area processing operations increases so will the required number of rows in the line buffer.

• *Tile Buffer Processing*

Some constrained computing environments will still be overtaxed by line buffering. This leads to the third alternative, tile buffering. With this method, a small two-dimensional region of pixels is read into memory, and a corresponding region of fully-processed pixel

values is produced. The size of the tile is determined by the size of the available cache memory, with the goal to minimize the number of memory cache misses that occur during the processing of a given tile. Consequently, execution time can be significantly reduced. Unfortunately, algorithm complexity continues to grow, as well.

Figure 3.13b illustrates tile buffer processing. The simple image processing chain will again be a 3×3 low-pass kernel, followed by color correction and then sharpening with a 3×3 high-pass kernel. As can be seen in the figure, the tile size has been set to 5×5. This allows the central 3×3 region of pixels to be blurred (B). These nine pixels can then be color corrected (C). Finally, the central pixel can be sharpened (S). Therefore, a 5×5 tile produces a single fully processed pixel! (As with line buffering, the details of the convolution operations here have been omitted. A second tile would be required to properly perform these convolutions.) Once the output pixel has been written, the tile is rolled in a manner similar to line buffering in order to reduce the number of computations that need to be repeated. In a similar manner, more complex image processing chains can be analyzed and implemented with tile buffering. As with line buffering, as the number of area operations increases so will the required size of the tile. It is noted in passing that hybrids between full-frame, line, and tile buffering are quite possible, depending on the nature of the computing environment.

3.3.3.2 Physical Environments

With a number of image data buffering options available, it is appropriate to discuss under which circumstances said methods are most applicable.

• *Custom Hardware Implementations*
For optimum computational efficiency, computing hardware can be explicitly designed to implement a given image processing chain. This usually takes the form of an *application-specific integrated circuit* (ASIC). Development of new ASICs is an expensive process requiring relatively high usage volumes to make this approach financially attractive. Therefore, to keep component costs at a minimum, image processing paths that use a minimal amount of memory are highly preferred. This typically results in designing image processing chains that can be implemented as "soda straw" pipelines. To this end, one will see line and tile buffering used almost exclusively. Additionally, the image processing chain itself will be relatively devoid of branch points that require intermediate results (e.g., resulting from luminance-chrominance splitting) to be kept in memory while other computations are performed. Because the size of the line and tile buffers is heavily influenced by the number of area operations and their respective support region radii in the chain, there will be a strong incentive to minimize both of these aspects. As a final observation, once constructed, the image processing chain in the ASIC cannot be changed. Therefore, the importance of careful and robust image processing chain design in an ASIC cannot be overemphasized.

• *Firmware Implementations*
The digital signal processor (DSP) provides a significant degree of freedom to the image processing chain engineer: the DSP is a programmable device. This greatly lowers the cost of the device compared to an ASIC as the same DSP can be used in a wide variety of products. Changes can also be made to the image processing chain in the DSP, which

provides the ability to upgrade the resident firmware to customize for specific applications, address bugs, or to add new features. In return for these benefits, processing speed is lost with respect to the ASIC.

Many of the considerations of using an ASIC carry over into using DSPs. Because cache memory is usually at a premium, line and tile buffering are still essential. The constitution of the image processing chain, however, may vary from the ASIC equivalent. Today's DSPs come with built-in capabilities that streamline certain essential image processing operations (e.g., convolution) making some decisions to perform "more" computations to actually result in shorter execution times. Therefore, designing an optimized image processing chain for a DSP puts the additional burden on the chain designer to understand the computational idiosyncrasies of the DSP in question.

• *Desktop Implementations*
Most image processing chains are prototyped in a desktop environment first. This can sometimes lead to a false sense of potential performance in one of the environments just discussed. Today's desktop computers have extremely large amounts of cache memory, compared to the DSP and (equivalently) ASIC environments. Therefore, area operations using larger support region sizes may run very quickly on a desktop computer while coming to a crawl in the DSP environment. Another advantage of the desktop computer is that memory is usually copiously available. This makes full-frame processing using many extra buffers for storing intermediate results practical, if not preferred.

If the ultimate compute environment of the image processing chain is a desktop computer, then there is generally no incentive to use line or tile buffering unless the sizes of the images are significant compared to the available memory. Professional and scientific digital cameras can have sensors that have in excess of 20 megapixels. When converted to full-color images, these become 60 megapixel entities. These numbers assume one byte of memory per color per pixel. For such high-end applications, it is not uncommon to require two bytes of memory per color per pixel leading to 120 megapixel images in memory. Now, consider the common scenario where it is desired to work with two or more such images simultaneously. Under these specialized circumstances, line or tile buffering may begin to be attractive.

3.3.4 Resizing and Compression

3.3.4.1 Image Resizing

One of the most common image processing operations not included in the reference chain is resizing. Resizing is simply a form of interpolation that encompasses both digitally enlarging (i.e., digital zoom) and reducing an image. The standard interpolation methods (e.g., pixel replication or subsampling, bilinear interpolation, and bicubic interpolation) are again the general workhorses of this operation. In the case of image reduction, there may also be the preliminary step of antialiasing (i.e., low-pass filtering or blurring) before interpolation to prevent the occurrence of aliasing artifacts in the resized image [28].

Conceptually, the simplest way to add resizing to the reference image processing chain is to append it at the end where it resizes the display RGB image (Figure 3.14). This is a useful position if nothing is known *a priori* about the resizing parameters (i.e., what the final size

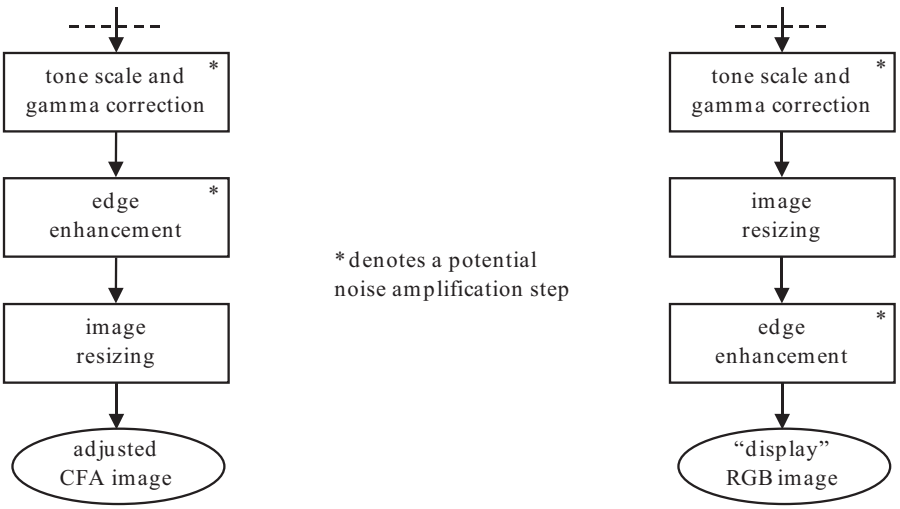


FIGURE 3.14

Image resizing performed after edge enhancement (left) and before edge enhancement (right).

of the resized image will be). This would be the scenario of a user-driven postprocessing operation, such as one might find in a kiosk or desktop application. However, even in this situation, it may be possible to find a better location in the image processing chain.

The effects of the edge enhancement operation are particularly sensitive to the final image size. The desired amount of sharpening for a standard-size video display image may produce an undersharpened or oversharpened image after resizing. The easiest way to address this situation is to perform resizing before edge enhancement (Figure 3.14). In this way, edge enhancement operates at the resolution of the final image size, providing some level of desensitization to image resizing. The correction is not ideal and further adjustments to the edge enhancement may be required.

If the resizing operation is constrained (e.g., only enlarging) or static (fixed to a specific enlargement / reduction ratio), then further economies may be possible. If the size of the final image will be reduced with respect to the original image, it makes sense to move the resizing operation to as early a stage in the image processing chain as possible. This will reduce the amount of data that is subsequently processed, decreasing memory usage and execution time. In this scenario, two locations in the chain suggest themselves: immediately after CFA interpolation (on the camera RGB image) and, further back, after the structured noise reduction. The former case is evident. The camera RGB image is a full-color, full-resolution image that is handled in one of the manners previously discussed. The latter case is a bit more problematic. The task, in this case, is to resize CFA image data. The exact details of the computations performed will be dictated by the actual CFA MRU. Formal interpolation of the individual subsampled color channels can be performed. However, the typical image reduction scenario is one that requires short execution times, for instance, for video-rate readout and real-time image preview. Therefore, formal interpolation operations are usually simplified, sometimes significantly. If the image quality requirements are sufficiently lax, one approach is to use a superpixel-inspired pixel subsampling, with or

without antialiasing preprocessing [29]. The main engineering task in these reduced computation environments is to minimize the creation of color aliasing artifacts resulting from CFA image data reduction.

If the size of the final image will be enlarged, then it would seem that the later on in the chain the resizing operation occurs, the better. The reasoning behind this decision is that the number of image pixels that must be processed is inflated only at the very end, thereby minimizing the amount of computation. This would lead to performing resizing either before or after edge enhancement. One confounding notion in this approach is the inherent redundancy between the resizing and CFA interpolation operations. Both perform a type of interpolation. With thought it is possible to conceive computational schemes that simultaneously perform CFA interpolation and resizing [30]. By directly computing a resized full-color image from original-resolution CFA data, a significant savings in execution time can be realized. Algorithmic complexity is the price that is paid for this improvement in efficiency. Chapter 17 discusses image resizing issues in detail.

3.3.4.2 Image Compression

Compression of an image is more an exercise in changing the representation of the data than in actually operating on the data itself. This is precisely true for lossless compression. On the other hand, while lossy compression can be viewed as having the added benefit of providing rudimentary noise reduction capabilities, the intent of most such compression operations is to leave the final image as visually similar to the original image as possible.

JPEG has long since become identified in the industry as the lossy image compression algorithm of choice [31]. (The lossless form of JPEG is largely ignored.) This algorithm, based on luminance-chrominance color spaces and discrete cosine transforms, performs a spatial frequency transform of the image, and then quantizes the frequency components in order to eliminate visually redundant information, which, as a result, produces a smaller image representation. With respect to lossless compression, there are a couple of choices. Especially for Web-based applications, use of Lempel-Ziv-Welch (LZW) compression as implemented in GIF and some TIFF file formats is a *de facto* standard [32]. More recently, the LZ77 variant called deflation and implemented in the PNG file format claims slightly better compression performance and, perhaps more importantly, freedom from any intellectual property (i.e., patent) entanglements [33]. It should be hastily added that this list is far from comprehensive, as compression continues to be an active area of research with applications in the areas of data storage and retrieval, transmission, and security being only some of the significant applications of this work.

Because the encoded state of the image data is generally an impractical one for any image processing operation other than decompression, it immediately becomes apparent that compression should occur on the final display RGB image prior to actual storage (Figure 3.15). This, of course, emphasizes one of the primary purposes of compression cited above: to reduce the storage size of the image. With the standardization of sRGB space, it has become commercially attractive to build JPEG hardware and firmware engines that perform an otherwise complex algorithm in a minimal amount of time. As a result, any digital camera that produces a fully processed image can largely be expected to compress that image in JPEG format prior to storage.

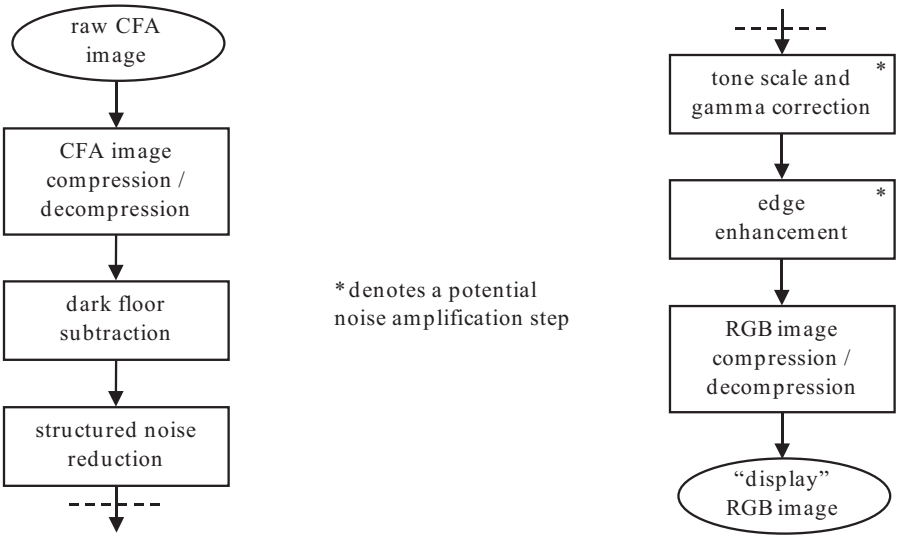


FIGURE 3.15

Image compression of the raw CFA image (left) and the display RGB image (right).

On a historical note, image compression was once performed much earlier in the image processing chain when digital cameras had only primitive compute capabilities on-board (see Figure 3.15) [25]. The role of the digital camera was once solely to capture a raw CFA image and then store that image in a compressed format for later image processing in a desktop environment. While it was clear that a luminance-chrominance space image can be compressed much more effectively than an RGB space image, CFA image data generally cannot be converted directly to, for example, YCrCb without an intermediate CFA interpolation step. Assuming a Bayer pattern CFA, the solution was to create a temporary interpolated green pixel value at each red and blue pixel location using a simple boxcar average of the four neighboring green pixel values. This in turn permitted the computation of R G and B G color differences at the red and blue pixel locations. The result was a luminance (green, full resolution)-chrominance (R G, B G, quarter resolution) representation of the CFA data that could be compressed in a lossy manner, resulting in a relatively small image file to be stored [34]. Upon decompression and transformation back to RGB, the temporary interpolated green pixel values were discarded in favor of results generated by better CFA interpolation methods. Detailed treatment of camera image compression can be found in Chapters 14 and 15.

3.3.5 Other Factors

Until recently, there has been a largely tacit assumption that the image data in the chain has been kept in the same photometric space as originally read from the sensor. There are many situations when this may not produce optimum results. These are discussed below.

TABLE 3.1
First values of the 8-bit linear to 8-bit video gamma transform.

Representation	*Values*										
8-bit linear	0	1	2	3	4	5	6	7	8	9	10
8-bit video	0	13	22	28	34	38	42	46	50	53	56

TABLE 3.2
First values of the 10-bit linear to 8-bit video gamma transform.

Representation	*Values*										
10-bit linear	0	1	2	3	4	5	6	7	8	9	10
8-bit video	0	3	6	10	13	15	18	20	22	23	25

3.3.5.1 Bit Depth

The number of code values used to span the range from full black to full white heavily influences the amount of *quantization error (contouring)* that will be seen in the final image. Unfortunately, this is the not the only factor as local scene content, image noise, and the Weber's Law sensitivity of the HVS also come into play. Still, bit depth can be considered of prime importance. Opposing the solution of simply carrying enough bits to eliminate quantization errors is the corresponding requirement of additional memory use to store and manipulate the extra data resolution.

Because sRGB video space consists of 8-bit data [0-255], it is natural to want to design an entirely 8-bit image processing chain. Supporting this decision is that to use larger bit depths immediately doubles (at least) the memory requirement from one-byte-per-pixel value to two-bytes-per-pixel value, unless data packing is used, which brings its own series of issues. Unfortunately, 8-bit linear data will not populate all 256 states after the video gamma transform. As shown in Table 3.1, the first 11 code values in 8-bit linear space map into only 11 of the first 57 code values in 8-bit video gamma space. The other 46 code values will never be populated. Because of the compressive nature of the video gamma curve, more of the larger code value states will be populated, but significant contouring should be expected in the shadow regions of the image.

Even-numbered bit depths appear to be preferred over odd-numbered bit depths, so the next data resolution level considered is usually 10 bits [0-1023]. The number of missing states after transformation to video gamma space is significantly reduced, but not eliminated (Table 3.2). Although there may be a small amount of contouring visible in shadow regions, this level is usually acceptable for consumer photography. For professional photography, however, it is best that all video gamma states are populated. This leads to the next bit depth to be considered (Table 3.3), which is 12 bits [0-4095].

While all of the output states are now populated in this example, quantization error can still occur as a result of some of the input 12-bit states being depopulated either because of poor exposure during capture (e.g., underexposure followed by digital exposure compensation) or loss of data resolution during the image processing chain computations. As a result, some professional digital cameras now employ 14 bits [0-16383]. From Table 3.4

TABLE 3.3
First values of the 12-bit linear to 8-bit video gamma transform.

Representation	*Values*										
12-bit linear	0	1	2	3	4	5	6	7	8	9	10
8-bit video	0	1	2	2	3	4	5	6	6	7	8

TABLE 3.4
First values of the 14-bit linear to 8-bit video gamma transform.

Representation	*Values*										
14-bit linear	0	1	2	3	4	5	6	7	8	9	10
8-bit video	0	0	0	1	1	1	1	1	2	2	2

it is clear that several of the 14-bit steps could be depopulated, and there would still be a great likelihood that all of the 8-bit video gamma states would be used.

The choice of bit depth is additionally complicated by matters beyond what have already been discussed. As the number of bits increases, the possibility of data overflow during the computation of intermediate numerical results becomes increasingly likely without using higher precision arithmetic. Because higher precision arithmetic requires more memory and takes longer to perform, this can become a significant liability. Additionally, the use of lookup tables with higher bit depths can become unwieldy as the sizes of the tables are forced to grow if numerical precision is to be preserved. In addition to having to maintain such large lookup tables, operations involving large tables can quickly produce an undesirable amount of memory cache hits.

3.3.5.2 Nonlinear Photometric Spaces

One compromise solution to the bit depth question is to store the image data in a nonlinear photometric space. One such space is the 8-bit video gamma space (see above). This is a solution frequency found in video imaging applications (see below). Another convenient family of spaces, to be discussed shortly, is based on logarithmic responses. The crux of these solutions is to constrain the numerical bit depth to eight bits by discarding code states that are least likely to produce visually objectionable image quantization artifacts. Referring to the Weber's Law response of the HVS, this suggests a compressive function response that preserves most of the states in the lower portion of the code range and discards states more frequently as the code values increase (i.e., a logarithmic-like response).

In the context of the image processing chain, the data would be produced from the image sensor and A/D converter at, for example, 10-bit resolution and then immediately transformed to an 8-bit space for storage in RAM. A pure logarithm function, $a\log(1+x)$, to perform this transformation will have problems near zero because of high slope of the curve and the resulting number of missing output states. Mimicking the solution used by the sRGB video gamma curve, a linear segment of unity slope is used to replace the logarithm in this region and the two functional pieces are matched in value and slope at the knot point to eliminate visible discontinuities (Figure 3.16) [35].

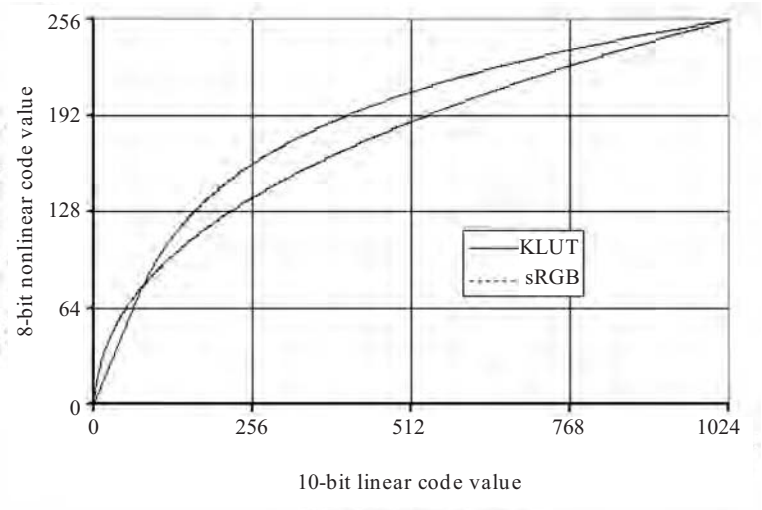


FIGURE 3.16

Comparison of linear-log transform (KLUT) and sRGB video gamma transform.

The general form of this transform then becomes

$$y = \begin{cases} x, & \text{for } x \le k \\ k + k \ln\left(\frac{x}{k}\right), & \text{for } x > k \end{cases} \tag{3.11}$$

$$k = -\frac{y_1}{W_{-1}\left(-\frac{y_1}{x_1} e^{-1}\right)} \tag{3.12}$$

where $W_{-1}(x)$ is the -1 branch of Lambert's W function [36]. The input code value range is from zero to x_1, and the output code value range is from zero to y_1. This approach can be augmented by adding a linear segment to the upper end of the logarithm to regulate the slope of the function in that region.

Once the functional form of the nonlinear transform is determined, the image data in the image processing chain can be freely transformed between nonlinear and linear spaces as needed. To speed the transform processes, lookup tables can be used. If this is impractical due to the size of the lookup table, the transform function can be approximated by a rational function of the form $(x + a_0)/(b_1 x + b_0)$, which may be simple enough to be recomputed as needed. Consideration must be given to keeping the number of transform-inverse transform cycles to a minimum as each operation may be lossy in terms of data resolution. To this end, while some image processing operations, such a color correction, in reality need to be performed on linear data (but see below), it is plausible to try to perform other operations, such CFA interpolation, in the nonlinear space.

3.3.5.3 Extended Dynamic Range

It is well known among photographers that silver halide films provide significantly more dynamic range capture capability ($\tilde{5}$ stops) than do silicon sensors ($\tilde{2}$ stops). Closing this

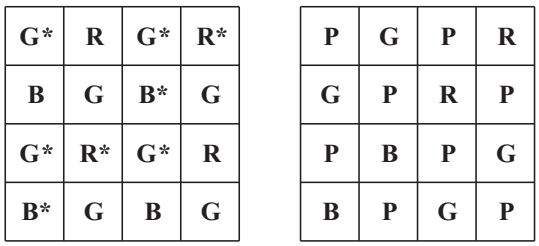


FIGURE 3.17

CFA MRU: (a) with different photometric gains and (b) with red (R), green (G), blue (B), and panchromatic (P) pixels.

performance gap is presently an active area of research in both academia and industry. The image processing chain ramifications of two prominent approaches are discussed below.

• *Multiple Color Channel Sensitivities*

One general approach is to expand the MRU of the CFA pattern, and for each color in the pattern provide two or more versions with different photometric gains [37].

Figure 3.17a illustrates one approach. Pixels marked with an asterisk have a higher photometric gain than those that are not. In processing, the point will inevitably be reached where the two inherent photometric ranges will need to be merged to produce an extended dynamic range image. This may take the form of a final image with the entire dynamic range preserved to provide the maximum flexibility in subsequent post-processing operations, or simply as an intermediate entity that is eventually reduced to an 8-bit sRGB image in a way that takes advantage of the extended dynamic range, for instance, via a custom tone scale transform. This will reintroduce the issue of bit depth and data resolution into the image processing chain design. Typically, one to two extra bits of data will be needed for each stop of exposure difference between the two photometric gains if no quantizing is performed. Where the merge of photometric ranges occurs in the image processing chain is a subject that must also be considered. One natural place for this is as early as possible in the image processing chain, usually CFA interpolation [38]. Once merged, the image can proceed through a normal image processing chain, albeit with a larger dynamic range. Alternately, the two dynamic ranges could be kept separated with custom image processing paths, in the manner of luminance-chrominance or high- and low-frequency splitting, until later on in the chain.

• *Separate Panchromatic Channel*

A different approach is to add a spectrally broad channel to the CFA MRU and use the natural boost in photometric sensitivity to create an extended dynamic range image [39], [40], [41].

Figure 3.17b illustrates one such arrangement. The pixels marked P in this figure are associated with the panchromatic (broad) spectral channel. The typical spectral response of such a panchromatic channel as compared to the more traditional RGB color channels is shown in Figure 3.18. The RGB color channels can be used to create a low-resolution

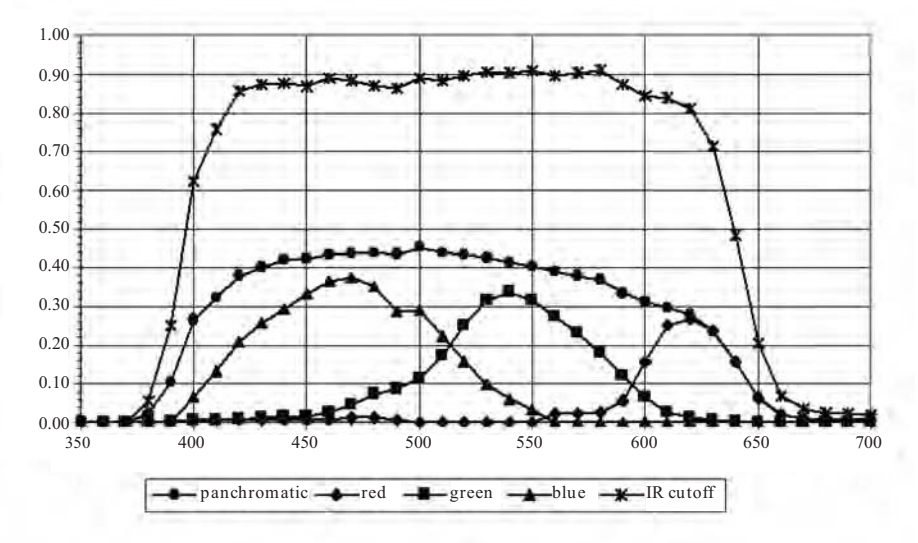


FIGURE 3.18

Typical spectral responses of red, green, blue, and panchromatic CFA filters. Also shown is a typical spectral response for an infrared (IR) cutoff filter.

full-color image with a typical photometric sensitivity. The panchromatic channel can be used to create a high-resolution grayscale image with high photometric sensitivity. These images can be merged by taking the low-frequency component of the full-color image and adding to it the high-frequency component of the panchromatic image [42], [43]. The result is a full-color image with extended dynamic range that is achieved by significantly lowering the noise floor of the imaging system. As discussed previously, the image processing chain can be bifurcated to process the low-resolution full-color image and the high-resolution panchromatic image differently before the point of merger.

3.4 How Video Differs from Still Photography

The foregoing discussion has been focused on *still* photography. When considering *video* photography, a number of assumptions must be significantly reassessed. Perhaps the most important issue is that real-time video image processing must be accomplished at video rates (e.g., 30 frames per second). This means the entire image processing chain must produce a fully finished video frame in around one-thirtieth of a second. This stands in stark contrast to a consumer digital still camera, which may take a second or two to produce a finished image. As a direct consequence, in order to lower the computation demands, pixel resolutions of video frames are much lower than for digital still images. Even with dramatically lowered pixel resolutions, it becomes almost immediately clear that the image processing chain must be significantly abbreviated for the video environment.

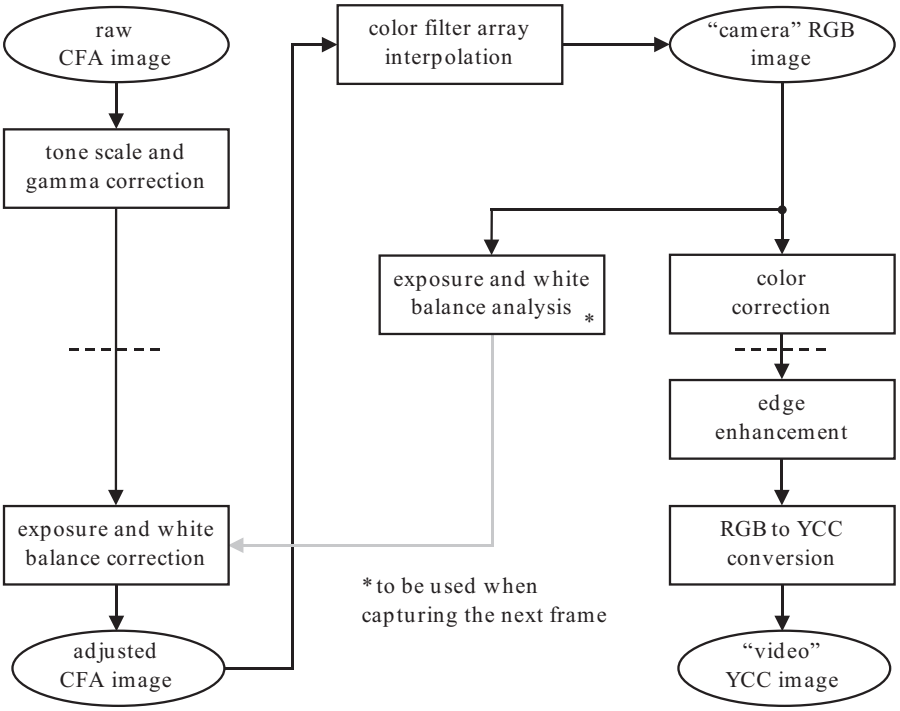


FIGURE 3.19

Idealized video image processing chain.

Somewhat offsetting the problem of limited execution times is that video image processing can take advantage of *temporal averaging* between consecutive frames. When the video sequence is viewed, each frame is only displayed for, continuing our example, one-thirtieth of a second. This is close to the refresh rate of the HVS, which is approximately between 20-60 Hz [44]. Each frame will "blur" into the next one and provide a perceived smoothing effect on the image. As a result, a certain amount of noise reduction comes "for free". This means that the image quality of the individual video frame does not need to be as high as a corresponding digital still image.

One engineering decision in video image processing chain design that seems to be made almost universally is to transform the raw data from the sensor directly into video gamma space and then retain it in that nonlinear space for all subsequent image processing operations (see Figure 3.19).

This has multiple benefits. First, the data resolution is fixed at 8 bits, permitting the use of a single byte of memory for each pixel value. Second, no time is spent transforming between photometric spaces in order to produce a final sRGB video frame. Third, quantization errors will tend to occur in the highlight of the image, where they are visually less objectionable, and will tend not to occur in the shadows of the image, where their visibility is more pronounced.

The liability with this "all-video gamma" approach is that certain image processing operations, most notably color correction, perform poorly in nonlinear spaces. Color correction is generally predicated on Grassmann's Laws [45], which describes color mixing as a lin-

ear phenomenon. For colors with low amounts of saturation (i.e., nearly neutral), video gamma computations will behave similarly to linear space computation. However, as the saturation increases (i.e., colors become more "colorful") computations in video gamma space can begin to produce results in significant variance from Grassmann's Laws. As a result, unexpected and unwanted color shifts may result from color correction. Therefore, performing color correction in video gamma space will reduce color fidelity. One way to address this problem is to tune the color correction matrix values to less aggressively correct the output colors in order to minimize the visibility of such errors. As a result, the loss of color accuracy due to reduced color correction has become part of the UE for consumer video photography.

In the idealized video image processing chain shown in the figure, no stochastic noise reduction operations are performed. Stochastic noise is considered to be addressed by either temporal averaging or any binning (summing) of adjacent pixel values performed by the hardware to produce the video frame raw CFA image data. (Temporal averaging will generally have no similar effect on structured noise.) After CFA interpolation, RGB data is transformed into YCC space, although these two operations could be consolidated into a single operation of CFA interpolating RGB data directly into YCC data if computational economies result. After color correction and edge enhancement, the video frame is complete and can be written to the output video stream, leaving the system ready to process the next frame. Exposure and white balance calculations are performed in a feedback loop between consecutive frames in the video sequence in order to prevent abrupt and undesirable changes in perceived image brightness or color.

The above describes video processing issues from the processing chain design point of view. Typical camera video processing tasks, such as video-demosaicking, resolution enhancement and video stabilization are discussed in Chapters 18 to 20.

3.5 Conclusion

The image processing chain that transformed digital camera raw sensor image data into a full-color fully processed image was the focus of this chapter. The possible orderings of individual operations and associated implementation details that constitute the image processing chain were discussed. Despite the seemingly immense number of available degrees of freedom, the problem of image processing chain design was seen to be overconstrained. The image processing task was to balance the opposing requirements of desirable image quality and modest compute resource use. It was shown that image processing operations that were highly effective might not be viable candidates for image processing chains in constrained compute environments. In the end, the process of designing an image processing chain became one of taking relatively simple, well-known image processing operations and staging them in a manner that produced the best synergistic effects.

References

[1] F. Sudo and T. Asaida, "Image defect correcting circuit for a solid state imager," U.S. Patent 5 144 446, September 1992.

[2] J. Heller and J. Breisch, "Electronic camera capable of detecting defective pixel," U.S. Patent 6 683 643, January 2004.

[3] J. Takayama and N. Takizawa, "Scaling algorithm for efficient color representation / recovery in video," U.S. Patent 6 236 433, May 2001.

[4] J. Hamilton, "Correcting for defects in a digital image taken by an image sensor caused by pre-existing defects in two pixels in adjacent columns of an image sensor," U.S. Patent 6 741 754, May 2004.

[5] J. Hamilton, "Correcting defects in a digital image caused by a pre-existing defect in a pixel of an image sensor," U.S. Patent 6 900 836, May 2005.

[6] J.S. Lee, "Digital image smoothing and the sigma filter," *Computer Vision, Graphics and Image Processing*, vol. 24, no. 2, pp. 255–269, November 1983.

[7] M. Gaboury, "Illuminant discriminator with improved boundary conditions," U.S. Patent 5 037 198, August 1991.

[8] Y. Takagi and T. Imaide, "White balance adjusting system including a color temperature variation detector for a color image pickup apparatus," U.S. Patent 5 170 247, December 1992.

[9] T. Miyano and E. Shimizu, "Automatic white balance adjusting device," U.S. Patent 5 644 358, July 1997.

[10] J. Adams, J. Hamilton, E. Gindele, and B. Pillman, "Method for automatic white balance of digital images," U.S. Patent 6 573 932, June 2003.

[11] R. Hunt, *The Reproduction of Colour*. Tolworth, UK: Fountain Press, 1987.

[12] B.E. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 1976.

[13] M. Noda and T. Imaide, "Solid-state imaging device with two-row mixing gates," U.S. Patent 4 768 084, August 1988.

[14] Y. Takizawa, "Solid-state color imaging apparatus for preventing color alias," U.S. Patent 4 794 448, December 1988.

[15] D.R. Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal," U.S. Patent 4 642 678, February 1987.

[16] J. Adams and J. Hamilton, "Adaptive color plan interpolation in single sensor color electronic camera," U.S. Patent 5 506 619, April 1996.

[17] J. Hamilton and J. Adams, "Adaptive color plan interpolation in single sensor color electronic camera," U.S. Patent 5 629 734, May 1997.

[18] P.S. Tsai, T. Acharya, and A. Ray, "Adaptive fuzzy color interpolation," *Journal of Electronic Imaging*, vol. 11, no. 3, pp. 293–305, July 2002.

[19] K. Hirakawa and T.W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," in *Proceedings of the IEEE International Conference on Image Processing*, Barcelona, Spain, September 2003, vol. III, pp. 669–672.

[20] M. Stokes, M. Anderson, S. Chandrasekar, and R. Motta, "A standard default color space for the Internet - sRGB," www.w3.org/Graphics/Color/sRGB.html, November 1996.

[21] R. Goodwin and A. Gallagher, "Method and apparatus for area selective exposure adjustment," U.S. Patent 5 818 975, October 1998.

[22] A. Gallagher and E. Gindele, "Method for adjusting the tone scale of a digital image," U.S. Patent 6 275 605, August 2001.

[23] J. Adams, J. Hamilton, and J.A. Hamilton, "Removing color aliasing artifacts from color digital images," U.S. Patent 6 804 392, October 2004.

[24] R. Hibbard, K. Parulski, and L. D'Luna, "Detail processing method and apparatus providing uniform processing of horizontal and vertical detail components," U.S. 4 962 419, October 1990.

[25] K. Parulski, D. Bellis, R. Hibbard, E. Giorgianni, and E. McInerney, "Method and apparatus for improving the color rendition of hardcopy images from electronic cameras," U.S. Patent 5 189 511, February 1993.

[26] J. Adams, J. Hamilton, and F. Williams, "Noise reduction in color digital images using pyramid decomposition," U.S. Patent 7 257 271, August 2007.

[27] D. Patterson and J. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*. San Francisco, CA: Morgan Kaufmann Publishers, 1998.

[28] W. Pratt, *Digital Image Processing: PIKS Scientific Inside*. Hoboken, NJ: John Wiley & Sons, Inc., 2007.

[29] C. Smith and J. Adams, "CFA correction for CFA images captured at partial resolution," U.S. Patent 6 366 318, April 2002.

[30] S. Yoshikawa, "Resizing images captured by an electronic still camera," U.S. Patent 7 092 020, August 2006.

[31] W. Pennebaker and J. Mitchell, *JPEG Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1993.

[32] T. Welch, "High speed data compression and decompression apparatus and method," U.S. Patent 4 558 302, December 1985.

[33] G. Roelofs, "Portable network graphics," www.libpng.org/pub/png/, 2007.

[34] D. Couwenhoven, B. Gandhi, and C. Smith, "Data compression rate control method and apparatus," U.S. Patent 5 596 602, January 1997.

[35] J. Adams, K. Spaulding, and K. Parulski, "Method and system for the reduction of memory capacity required for digital representation of an image," U.S. Patent 5 708 729, January 1998.

[36] E. Weisstein, "Lambert W-function," www.mathworld.wolfram.com/LambertW-Function.html, 2005.

[37] A. Gallagher and D. Nichols, "Method and apparatus to extend the effective dynamic range of an image sensing device," U.S. Patent 6 909 461, June 2005.

[38] J. Adams and J. Hamilton, "Extended dynamic range image sensor capture using an array of fast and slow pixels," U.S. Patent Application 2005/0140804, June 2005.

[39] J. Compton and J. Hamilton, "Image sensor with improved light sensitivity," U.S. Patent Application 2007/0024931, February 2007.

[40] J. Hamilton and J. Compton, "Capturing images under varying lighting conditions," U.S. Patent Application 2007/0046807, March 2007.

[41] T. Kijima, H. Nakamura, J. Compton, and J. Hamilton, "Image sensor with improved light sensitivity," U.S. Patent Application 2007/0177236, August 2007.

[42] J. Hamilton and J. Compton, "Processing color and panchromatic pixels," U.S. Patent Application 2007/0024879, February 2007.

[43] J. Adams, J. Hamilton, and M. OBrien, "Interpolation of panchromatic and color pixels," U.S. Patent Application 2007/0024934, February 2007.

[44] A. Giorgi, "Effect of wavelength on the relationship between critical flicker frequency and intensity in foveal vision," *Journal of the Optical Society of America*, vol. 53, no. 4, pp. 480–486, April 1963.

[45] R. Hunt, *Measuring Colour*. London, UK: Ellis Horwood, 1995.

4

Optical Antialiasing Filters

Russ Palum

4.1 Introduction

Aliasing is an artifact of digital images; it occurs when an image contains detail smaller than the pixel pitch. This chapter starts with an intuitive look at how aliasing can occur in sampled images. Then, the history and the description of antialiasing filters are presented in Section 4.2. Nyquist diagrams, which provide a way to analyze and visualize antialiasing filter requirements, are discussed in Section 4.3. Section 4.4 discusses the modulation transfer function, a measure of system resolution, because the occurrence of aliasing is dependent on system resolution. Lens modulation transfer function is the focus of Section 4.5. Sections 4.6 and 4.7 present a brief description of convolution and Fourier analysis. Many of the details of sampled imaging systems become apparent when analyzed in the frequency domain; therefore, Section 4.8 steps through the sampling process in the spatial and frequency domain. Subsequently, reconstruction issues are discussed in Section 4.9. Finally, antialiasing filter construction and testing are the central topics of Sections 4.10 and 4.11, and conclusions are presented in Section 4.12.

(a) (b)

FIGURE 4.1

Aliasing effects: (a) image with aliasing, and (b) antialiased image.

4.1.1 Aliasing

A digital camera is an example of an imaging system that samples with a regular array of points. Aliasing is an artifact of sampled imaging systems. Figure 4.1a shows an example of aliasing in a digital image. The swirling lines are the result of narrow stripes that have been imaged so that the stripes look like a few wide lines; this is an aliasing artifact. The image in Figure 4.1b has been properly prepared prior to sampling; the narrow stripes are not resolved but they do not produce swirling wide line artifacts. Low resolution is better than low frequency artifacts that do not represent the original object.

Once an image is sampled, the aliased low-frequency content is difficult to correct automatically because it is the same as actual low-frequency content. Software is available to correct aliasing artifacts but intervention is usually required to locate the artifact. Silver halide photographic systems sample on a random array of points so they do not produce aliasing artifacts. Interestingly (and surprisingly), for local areas, the human retina samples a regular array of points, a close-packed hexagonal grid. The eye lens limits the image spatial frequency content to prevent aliasing [1].

Figure 4.2 is the beginning of an intuitive look at the cause of aliasing. A section of imager 1 mm long is shown as a two-dimensional (2D) grid. The spatial image at the bottom varies sinusoidally in the horizontal direction. The top bars represent the sampled image, displayed so that each pixel occupies one square. This is a simple way to display a basic image from a sampled image. The example imager samples at a rate of 40 samples/mm; this sampling rate is called the Nyquist rate. The spatial frequency imaged on the array is one-half the Nyquist rate, 20 cycle/mm; this is called the Nyquist frequency. According to the Nyquist criteria [2], a spatial frequency greater than half the sampling rate cannot be reproduced. The Nyquist criterion is further extended by the Whittaker-Shannon theorem [3]: if a signal is band limited to within the Nyquist frequency of the sensor, the signal can be recovered, without error, from the sampled signal. Or, worded differently: any band-limited function can be specified exactly by its sampled values taken at regular intervals, provided that these intervals do not exceed some critical sampling interval.

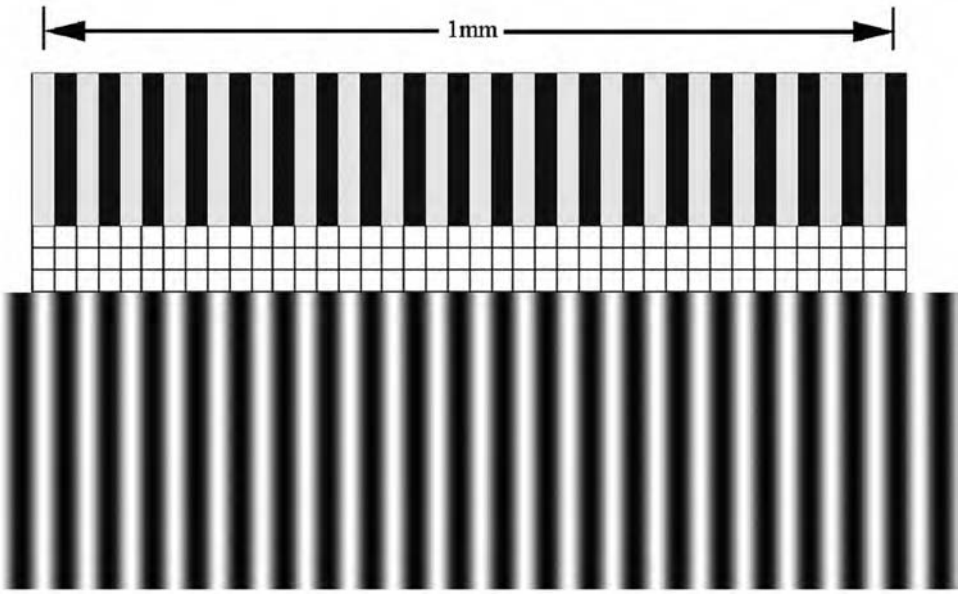


FIGURE 4.2

Nyquist frequency.

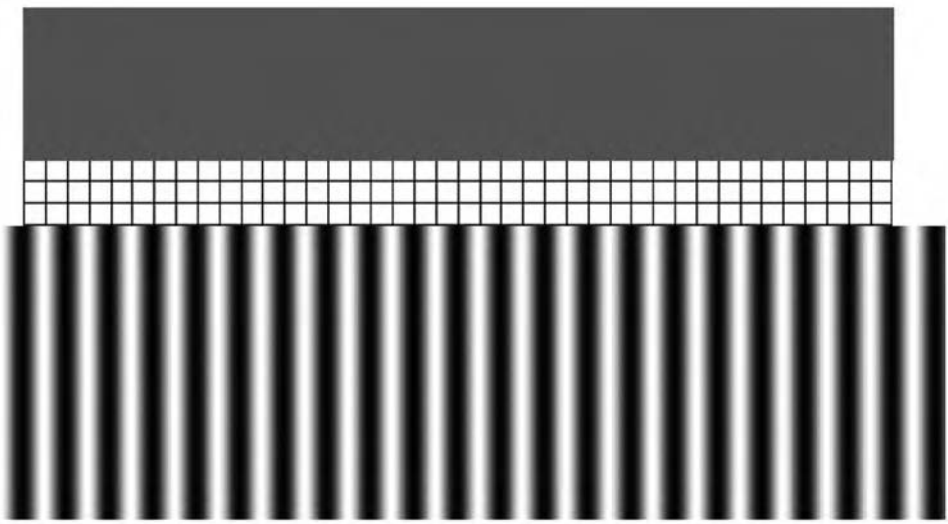


FIGURE 4.3

Shifted sine wave.

Figure 4.3 illustrates a pathological case at the edge of the Whittaker-Shannon theorem. A 20 cycle per mm image is shifted by one-half pixel. Each pixel is centered on the average value of the sine wave and each pixel sees half of the bright portion of the sine wave and half of the dark portion. There is no pixel-to-pixel modulation because all the pixels see the same amount of light. This sampled image cannot be recovered.

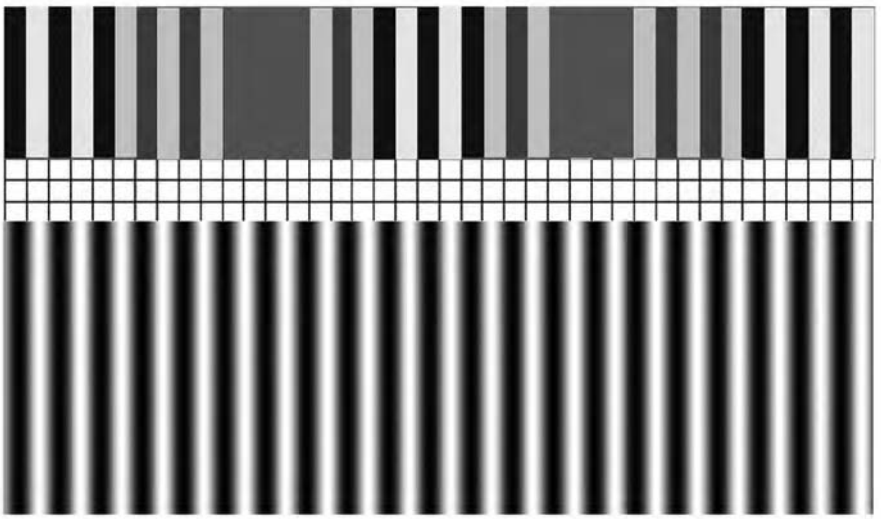


FIGURE 4.4

21 Cycle per millimeter sine wave.

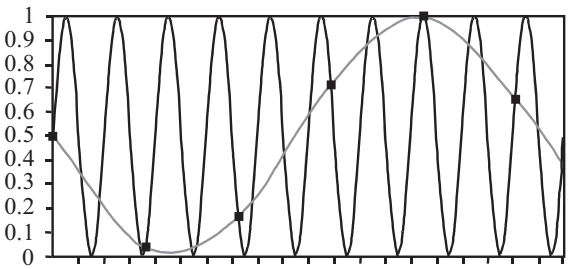


FIGURE 4.5

Aliasing in temporal signals.

The Whittaker-Shannon theorem is violated in Figure 4.4; a 21 cycle per mm image is falling on a 40 sample per mm imager. This spatial frequency exceeds the Nyquist criteria. Notice the sine wave image starts in phase with the pixels, so it is reproduced. It goes out of phase around the tenth pixel, so these pixels become a wide band. The image goes back in phase at the twentieth pixel, out of phase again at the thirtieth, and in phase again at the fortieth pixel. This leads to an image with wide bands separated by narrow bands. The sampled image cannot reproduce the original image because 40 pixels can only reproduce 20 cycles of light-to-dark transitions and 21 cycles requires 22 light-to-dark transitions. This is an aliasing artifact.

Figure 4.5 is a common representation of aliasing. The signal is usually temporal and it is sampled with an analog-to-digital converter or some other sampling device. The dark squares are evenly spaced samples on the high-frequency sine wave. The sampling in this case is well below Nyquist and the result is a very low-frequency aliased representation of the actual sine wave.

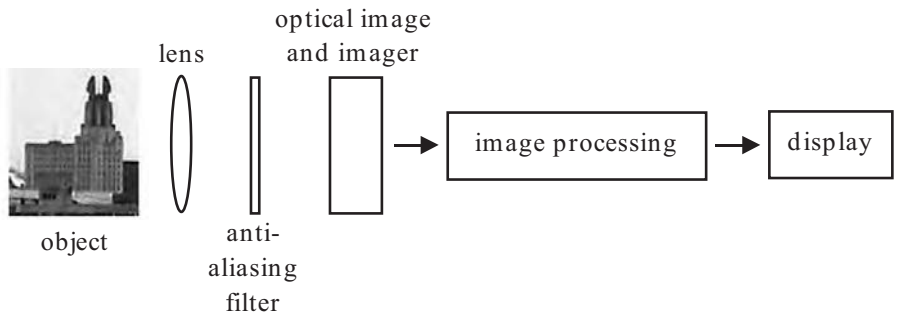


FIGURE 4.6

Digital imaging system.

4.1.2 Digital Photographic Systems

A typical digital photographic system contains a lens, an image capture device, an image display device, and some amount of image processing capability. The basic diagram in Figure 4.6 can be used for a film scanning system, microscopy, astronomy, or any other digital imaging system that starts with an optical image.

Aliasing is prevented by band-limiting the optical image spatial frequencies to the Nyquist frequency and below. Fourier analysis is used to determine if the frequency content of the image is band-limited to the Nyquist frequency. Most digital cameras use an antialiasing filter to band-limit the optical image spatial frequencies. Limiting the image spatial frequencies is equivalent to blurring the image, so these filters are sometimes called blurring filters, which really scares the marketing staff.

According to Greivenkamp (Reference [2], p. 676), the first antialiasing filter appears to have been invented by Pritchard for color stripe single-tube video cameras. Prior to single-tube color cameras, aliasing was not a substantial problem. Black and white cameras and three-tube color cameras have an analog horizontal signal. The vertical sampling has 100% fill so it does not alias substantially and there are no complications from color errors. Color stripe video cameras sample color horizontally and early cameras used analog electronic algorithms to produce a full-resolution color image. Aliasing in a single-tube color stripe camera can produce color artifacts that do not even match the color name of the original object.

4.2 Nyquist Domain Graph

The previous examples have been one-dimensional (1D). The Nyquist frequency of 2D imaging systems depends on direction. A Nyquist domain graph is used to display the locus of points at the Nyquist frequency in two dimensions [4]. Spatial frequency on the graph corresponds to the inverse of the spacing in the spatial domain. Radial distance from the center of the graph corresponds to spatial frequency.

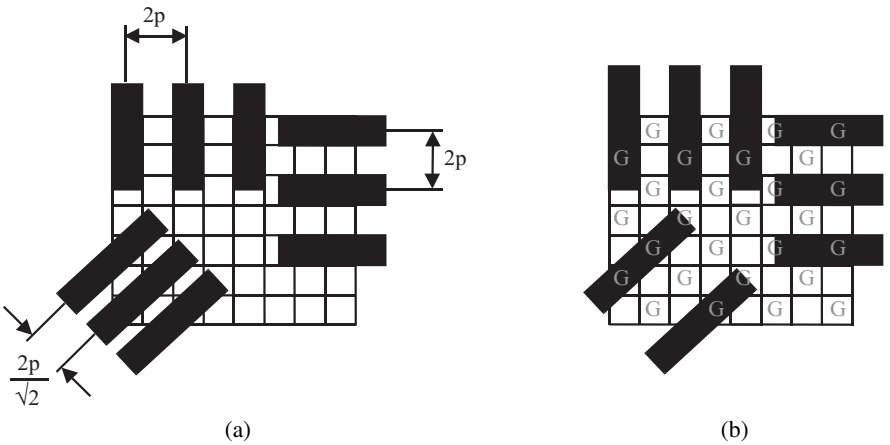


FIGURE 4.7

Bars at the Nyquist frequency: (a) imager, and (b) green channel Bayer pattern.

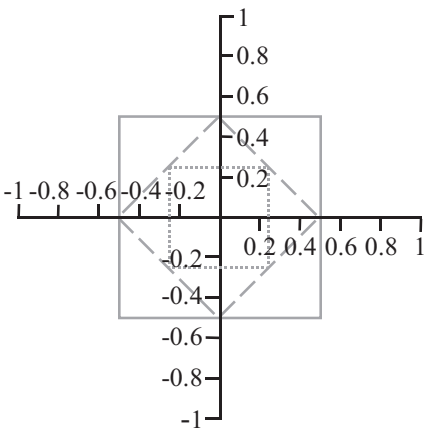


FIGURE 4.8

Nyquist domain graph with imager sampling frequency normalized to 1/p. Solid line corresponds to the monochrome imager, dash line corresponds to the green channel, and dot line corresponds to the red and green channels.

Figure 4.7a is a small section of a monochrome imager. The vertical and horizontal bars are at the vertical and horizontal Nyquist frequency. The period of the bars is twice the pixel pitch (p). The frequency of the bars is 1/(2p). Notice the spacing of the diagonal bars is smaller by 1/(square root of two). The corresponding Nyquist domain graph is shown in Figure 4.8 by the solid line, with the horizontal and vertical coordinates normalized to the sampling frequency, 1/p. Normalizing to the sampling frequency generalizes the discussion. The diagonal coordinates are at 1/(2p) or 1.414p. A monochrome imager has a higher Nyquist frequency diagonally than it has on the vertical and horizontal axes.

Single imager color systems with a color filter array (CFA) pattern add additional complexity to the Nyquist domain graph. A section of an imager with a Bayer CFA pattern is shown in Figure 4.9; the pattern has three color channels that are analyzed separately. The

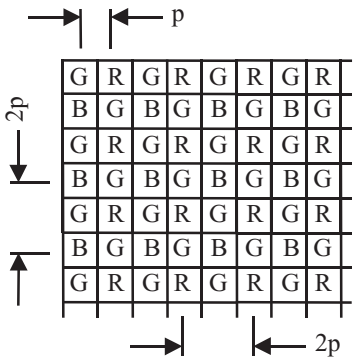


FIGURE 4.9

Bayer color filter array.

green channel is a square pattern with twice as many elements as the red and blue channels. However, the square pattern is rotated 45 degrees so that it looks like a checkerboard. The relationship between the diagonal and the vertical / horizontal Nyquist frequency is reversed compared to the monochrome imager because the green channel is a square pattern rotated 45 degrees.

The green pattern is shown with bars at the Nyquist frequency in Figure 4.7b. The pixels are not contiguous; this is called a sparse array. The Nyquist rate is still the inverse of the period for a sparse array. The horizontal and vertical Nyquist frequency, normalized to the monochrome pixel pitch, is a half cycle per sample. The period of the diagonal bars at the Nyquist frequency is larger so the Nyquist frequency is lower on the diagonal. The green channel Nyquist domain graph is the inner diamond shown in Figure 4.8 by the dash line.

As shown in Figure 4.9, the red and blue channels have the same pitch, 2p, where p is the monochrome pixel pitch. Both patterns are square sparse arrays with a spatial offset between the two channels. The offset does not affect the analysis. The red and blue channel Nyquist domain graphs are the same and they are similar to the monochrome Nyquist domain graph except the frequencies are half as high because the red and blue pitch is twice as large. The Nyquist domain graph for the red and blue channels is shown in Figure 4.8 by the dot line.

4.3 The Four-Spot Birefringent Antialiasing Filter

The four-spot birefringent antialiasing filter is the most common antialiasing filter. The construction of this type of filter is discussed in Section 4.10; however, a brief discussion of birefringence will be useful here.

Some optical crystalline materials are birefringent; calcite is a naturally occurring example of a material that is very birefringent. These materials can be cut so a ray of light that enters the crystal is split into two rays that take different paths through the crystal and emerge with a separation between the rays. An example is shown in Figure 4.10a and Fig-

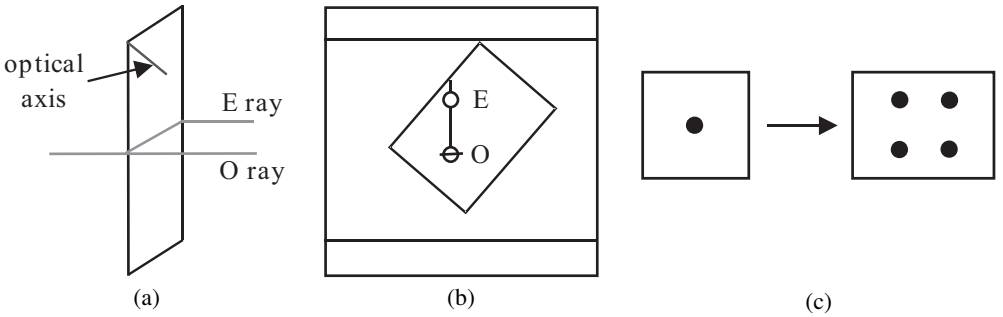


FIGURE 4.10

Birefringent antialiasing filter: (a) birefringence on the crystal slice, (b), the slice with the outline of a plate that will be included in an antialiasing filter, (c) four-spot blur filter.

ure 4.10b. The amount of separation is determined by the material and the thickness of the plate. When a plate of birefringent material is placed behind a lens it will spilt the image into two images with a separation between them. For a small separation the image appears to be slightly blurred. Multiple plates of a birefringent material can be made to produce four or more images that will blur the image in many directions.

The four-spot birefringent filter limits the scene spatial frequency content by spreading the light from every point in the image over four points as shown in Figure 4.10c. The thickness of the filter determines the spacing between the four spots. The separation is chosen to limit the image spatial frequency content to the Nyquist frequency and below. The separation for a monochrome imager is the pixel pitch. The details of the modulation calculation are presented in Section 4.8.

The modulation at the Nyquist frequency is reduced as shown in Figure 4.11a; light from a single point is focused on two pixels. This is true for every point on the object. If the object is a sinusoid that produces an image at the Nyquist frequency, the sinusoid will be dark on one pixel and light on the next. Blurring light from each object point over two pixels produces the same light level at every pixel when the image is a sinusoid at the Nyquist frequency so there is no modulation at the Nyquist frequency.

Alternatively, looking through the filter from the pixel (Figure 4.11b), the light that falls on the pixel will appear to come from two places in the object plane. If the object is a sinusoid that will produce an image at the Nyquist frequency, the two spots will appear to come from points 180 degrees of phase apart so these points will always add to the average value. This is the case for every pixel, so there is no modulation at the image plane for

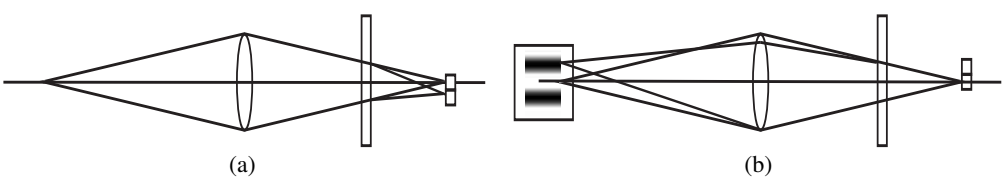


FIGURE 4.11

(a) Antialiasing filter effect on the image plane. (b) Antialiasing filter looking back at scene.

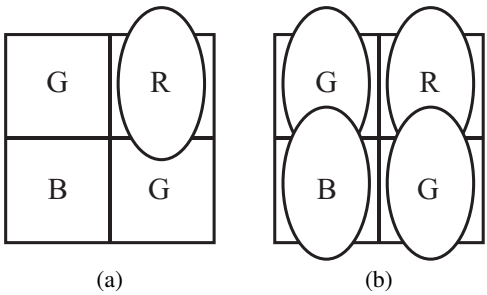


FIGURE 4.12

Preventing color interpolation error: (a) highlight falling on one color component, (b) spreading the light over four pixels to prevent the color error.

sinusoids at the Nyquist frequency.

The same four-spot filter is used for Bayer CFA imagers. It should be clear, based on the Nyquist domain discussion, that this blur filter will not prevent aliasing in a Bayer CFA imager. There are a few reasons why this apparently inadequate filter is adequate. First, there is resistance to applying enough blur to prevent aliasing because no one likes to pay for pixels and an expensive lens and then blur the image. Second, many interpolation algorithms take advantage of the correlation between channels so the effective color channel arrays may not be as sparse as they appear. Some examples of this are presented in Section 4.11. Next, the antialiasing filter prevents color interpolation errors. It is not unusual for a highlight, like a catch light in an eye, to fall on one pixel. The catch light in Figure 4.12a will be rendered bright red by the interpolation algorithm because it only illuminates a red pixel. The antialiasing filter spreads the light over four pixels as shown in Figure 4.12b to prevent the color error. Sharp edges may have the same type of problem depending on the interpolation algorithm. It is possible to make a birefringent antialiasing filter that has the appropriate spacing for each color but this filter is difficult to manufacture [2].

4.4 Modulation Transfer Function

An understanding of modulation transfer function (MTF), convolution, and a few Fourier transform pairs are required to understand the analysis of sampled imaging systems. MTF analysis is used to determine the limits of the optical image spatial frequency content at the imager. The MTF for each component can be measured or determined from theory and then the system MTF limits can be determined by cascading the component MTF's.

The MTF is a measure of system and component spatial resolution performance [5]. It is the ratio of the signal output modulation M_o to the sine wave input modulation M_i as a function of spatial frequency r. Modulation M is determined as follows:

$$M = \frac{M_{max} - M_{min}}{M_{max} + M_{min}} \tag{4.1}$$

where M_{max} and M_{min} are the sine wave peak and valley measurements. The input modula-

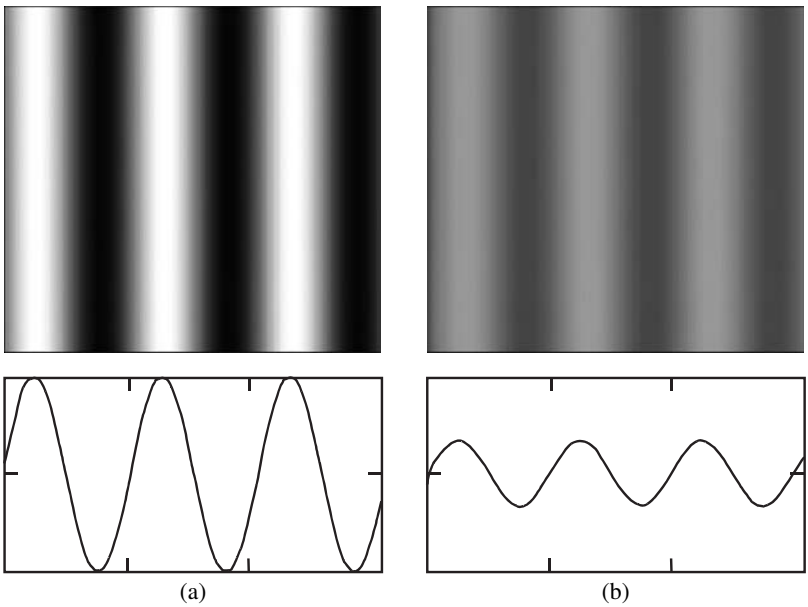


FIGURE 4.13

Input and output sine wave. Top row shows: (a) original sine wave with 100% modulation, and (b) image of sine wave with 33% modulation.

tion can be measured in luminance or any other linear metric. The output modulation also has to be measured in a linear metric like transmission or linear code value. The MTF ratio, the ratio of the output modulation to the input modulation, is calculated as follows:

$$\mathrm{MTF}(r) = \frac{M_o(r)}{M_i(r)} \tag{4.2}$$

The MTF curve is the result of a number of MTF measurements taken at different spatial frequencies.

To produce an MTF curve, an input sine wave as shown in Figure 4.13a is imaged to produce the output image shown in Figure 4.13. The input and output max and min are measured. The modulation is calculated using Equations 4.1 and 4.2 to determine the modulation at one spatial frequency. As previously mentioned, the complete MTF curve is the result of repeating this procedure for a range of spatial frequencies.

Figure 4.14 is a sine wave pattern that increases in frequency linearly with distance. This is commonly called a chirp because of its similarity to the sine wave pattern of a bird chirp. This pattern can be used to make an MTF measurement over a range of frequencies with just one image. The middle graph in Figure 4.14 is a trace of the input and output chirp. Notice the low frequency part of the pattern is reproduced with good modulation, but the higher frequencies drop off; therefore, the MTF drops as shown at the bottom of Figure 4.14.

MTF is nominally expressed as a number from zero to one, or it can be converted to a percentage. When a digital image is sharpened, the MTF at some frequencies can exceed one. Sharpening can also be applied to silver halide photographic systems. Silver halide film can be formulated to chemically enhance edges so the MTF may be larger than one at

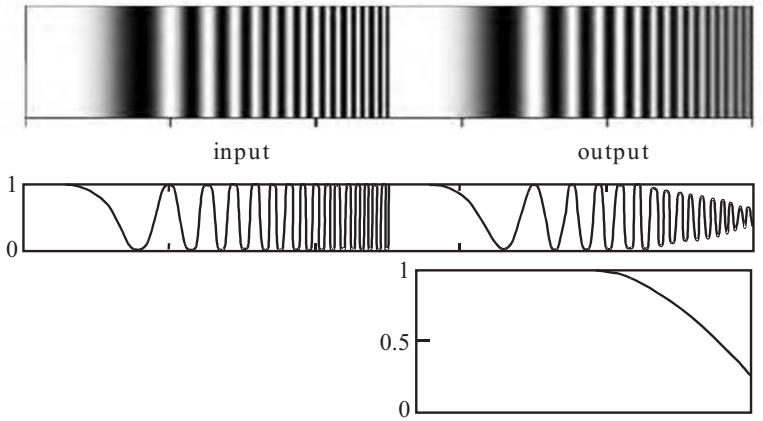


FIGURE 4.14

Chirp image.

some spatial frequencies. Negative MTF values are also possible; these are the result of a phase change in the image. For example, a white peak in an image may be at the location of a dark peak in the object.

The MTF measurement assumes the system is linear or can be made linear. Nonlinear systems can be made linear by measuring the low frequency transfer function to convert the output to the corresponding input luminance which is a linear measure. Silver halide photographic materials can be made linear with this technique even though they are inherently nonlinear.

4.5 Lens Modulation Transfer Function

A lens is usually the first component in a digital imaging system. It can reduce the system spatial frequency content, but lens effects change with f-number and focus so the lens alone is not an effective control for aliasing. In most systems, the lens controls the spatial frequency content beyond the Nyquist frequency, but a birefringent blur filter is usually used to control spatial frequency content near the Nyquist frequency.

Lenses are designed by adjusting glass types, surface shapes, and spacing to optimize the resolution in the image plane. This type of analysis is based on geometric optics. It is possible to design a lens that images perfectly based on geometric optics but lens performance is further limited by diffraction. Lenses that focus all the rays within the effects of diffraction are called diffraction-limited lenses. The point spread function for a diffraction-limited lens is an Airy disk. Figure 4.15a is an image of an Airy disk that has been adjusted to make the outer rings visible (Reference [5], p. 160). The outer rings are very faint in an accurate rendition. Eighty-four percent of the power is in the center core, 7% is in the first ring and 3% is in the next ring. The equation for the Airy disk is [6]:

$$E(q,\lambda,N) = \left(\frac{2J_1(\frac{\pi q}{\lambda N})}{(\frac{\pi q}{\lambda N})} \right)^2 \tag{4.3}$$

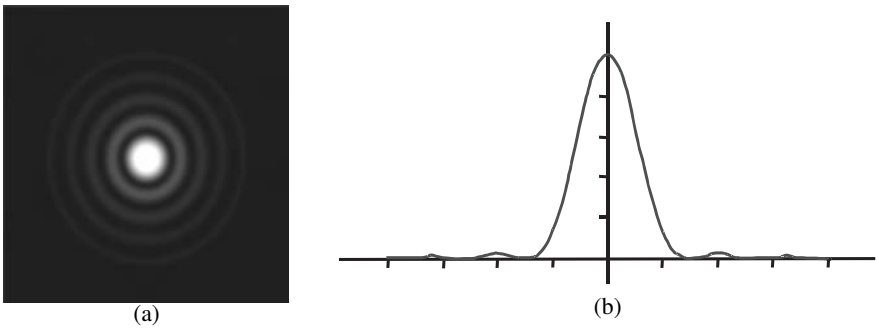


FIGURE 4.15

(a) Airy disk, (b) Airy disk graph.

where E is the relative illumination, q is radius, λ is wavelength in the same units as q, N is lens f/#, and $J_1(\cdot)$ is a Bessel function of the first type. The Airy disk graph is shown in Figure 4.15b.

For most photographic applications, the wavelength range of interest is 400 to 700nm. The center of the wavelength range, 550nm, is chosen to determine the size of the Airy disk. For a selected wavelength, the diameter of the center core from zero crossing to zero crossing is strictly a function of the lens f/#. The diameter can be determined as follows:

$$D \doteq 2.44\lambda N \tag{4.4}$$

As a rule of thumb, the diameter of the bright spot in micrometers is approximately equal to N, the f/#, because the product of the wavelength (approximately 1/2 mm) and the constant (value 2.44) is close to one.

Figure 4.16a shows a diffraction-limited lens MTF curve (see Reference [5], p. 377). All diffraction-limited lenses have the same MTF curve shape. The zero crossing, called the cutoff frequency, is a function of the lens f/#. The cutoff frequency increases with increasing aperture size, which corresponds to decreasing f/#. The cutoff spatial frequency (v_c) is:

$$v_c = \frac{1}{\lambda N} \tag{4.5}$$

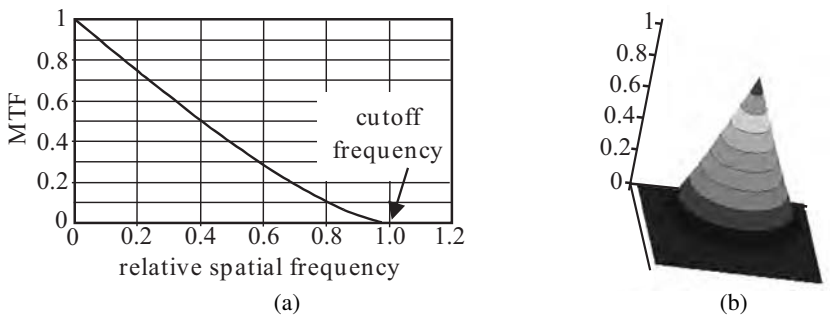


FIGURE 4.16

(a) Two-dimensional lens MTF, (b) three-dimensional lens MTF.

An equation for diffraction limited lens MTF as a function of spatial frequency is presented in Section 4.8. Lenses with a round aperture have a circularly symmetric diffraction-limited MTF; the MTF is the same for a given spatial frequency, regardless of direction. Figure 4.16a shows a 2D MTF whereas its three-dimensional version is depicted in Figure 4.16b.

4.6 Fourier Analysis and Convolution

Fourier analysis is used to analyze sampled systems because the MTF of each component or system is the modulus of the Fourier transform of the point spread function for that component or system. In addition, Fourier techniques can make sampled system analysis easier when convolution is required. The Fourier transform of the convolution of $f(x)$ and $g(x)$ equals the product of the Fourier transforms of $f(x)$ and $g(x)$. In addition, the Fourier transform of the product of $f(x)$ and $g(x)$ is the convolution of the Fourier transforms of $f(x)$ and $g(x)$. In many cases, it is easier to determine the result of a convolution using Fourier techniques than it is to directly compute the convolution.

Gaskill [7] presents an excellent explanation of convolution. A simplistic explanation of convolution in one dimension is offered in Figure 4.17. A rectangle is convolved with a rectangle. Convolution, by definition, includes flipping the first function left to right. Most sampled imaging system functions are symmetric, so neglecting the flip does not change the result. Starting at Figure 4.17a, the rectangle slides along the axis in infinitely small steps. The two functions are multiplied and the convolution is the area of the product at each point. In this case, the convolution is zero until the shaded rectangle touches the second rectangle shown in Figure 4.17b, then it ramps up, as shown in Figure 4.17d. The convolution is constant while the first rectangle is inside the second rectangle and then drops back to zero as the first rectangle leaves the second rectangle, as shown in Figure 4.17e.

A second case, depicted in Figure 4.18, shows the convolution between two equal rectangles. Notice that the rectangles are only exactly inside each other at one point, thus resulting in a triangle rather than a trapezoid. The image formed by a lens is the result of

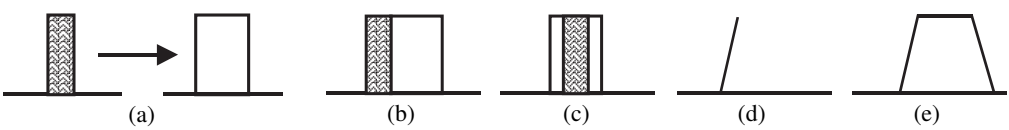


FIGURE 4.17

Convolution.

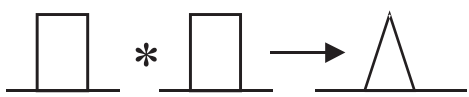


FIGURE 4.18

Convolution with equal rectangles.

FIGURE 4.19

Airy disk convolution.

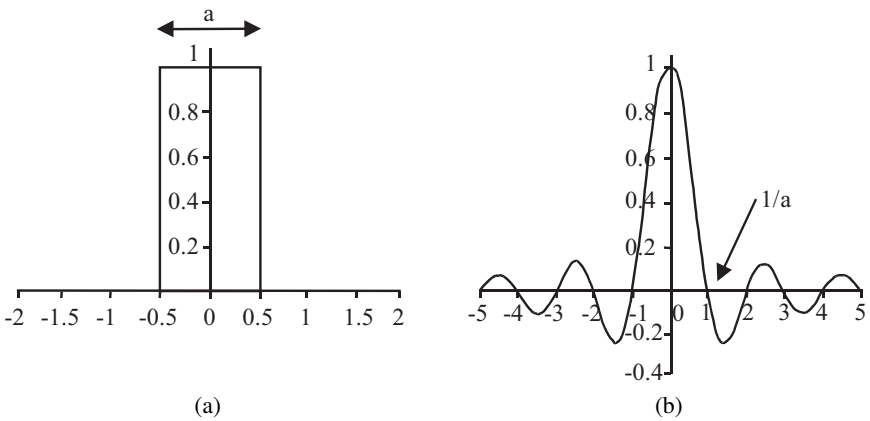


FIGURE 4.20

(a) Rect function. (b) Sinc function.

a 2D convolution between the object and the lens point spread function. The point spread function for a lens is an Airy disk. To simplify the explanation a magnification of one and a bar target object are used. The Airy disk slides over the bar target in Figure 4.19, and the convolution is calculated at each point, resulting in the blurred bar target.

4.7 Fourier Transform Pairs

There are a few Fourier transform pairs needed for sampled image system analysis. For a more detailed discussion of Fourier analysis refer to References [3] and [7]. There are also two properties of the Fourier transform that are very useful for imaging system analysis.

First, the Fourier transform of the convolution of two functions equals the product of the Fourier transforms of the individual functions. Second, except for scaling, the Fourier transform of the Fourier transform of a function is the original function. The rect function described below is a good example of this. The Fourier transform of a rect is a sinc and the Fourier transform of a sinc is a rect.

The rect function is a rectangle in the spatial domain. Figure 4.20a is a 1D rect of width a. The Fourier transform of a rect as shown in Figure 4.20b is a sinc. The following describes the sinc function:

$$F\left(\text{rect}\left(\frac{x}{a}\right)\right) = |a|\,\text{sinc}(\pi ar) = \frac{\sin(\pi ar)}{\pi r} \tag{4.6}$$

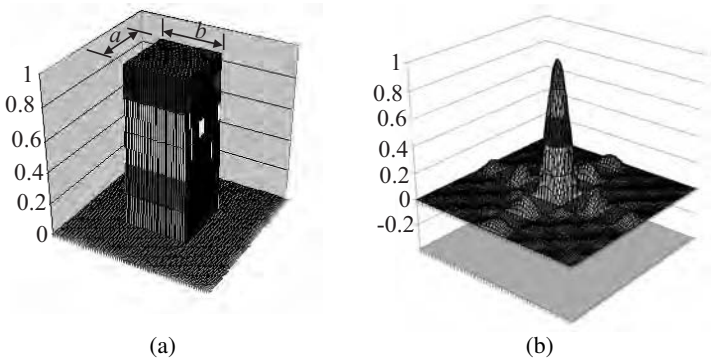


FIGURE 4.21

(a) Two-dimensional rect, (b) two-dimensional sinc.

where a is the width of the rect on the x axis in the spatial domain and the frequency variable is r. The F indicates a Fourier transform should be taken. In general, the width of a Fourier transform is inversely related to the size of the original function. In Figure 4.20b, the first zero of the sinc in the frequency domain is at $1/a$ where a is the width of the rect function in the spatial domain.

The 2D rect is a rectangular solid (Figure 4.21a). Its Fourier transform:

$$F\left(\text{rect}\left(\frac{x}{a},\frac{y}{b}\right)\right) = |a|\,\text{sinc}(\pi ar)\cdot|b|\,\text{sinc}(\pi bs) = \frac{\sin(\pi ar)}{\pi r}\cdot\frac{\sin(\pi bs)}{\pi s} \tag{4.7}$$

is a 2D sinc (Figure 4.21b). The 2D sinc is the product of two 1D sinc functions (this is called a separable function). Where, in the spatial domain, a is the x-axis width of the rect and b is the y-axis width. The corresponding frequency variables are r and s.

The delta function (also known as the Dirac delta function or the impulse function), is represented graphically in Figure 4.22a. The arrow is optional; it indicates that the height is not as shown. The function has an area of one and a width of zero. The function is zero except at $x = 0$. The function notation and definition, including a shift in two dimensions, is:

$$\begin{aligned}\delta(x-a,y-b) &\neq 0 \;\text{ at } x=a,\; y=b\\ \delta(x-a,y-b) &= 0 \;\text{ at } x\neq a,\; y\neq b\end{aligned} \tag{4.8}$$

The Fourier transform of the delta function at x and y coordinates a and b is an exponential:

$$F\left(\delta(x-a,y-b)\right) = e^{-j2\pi(ar+bs)} \tag{4.9}$$

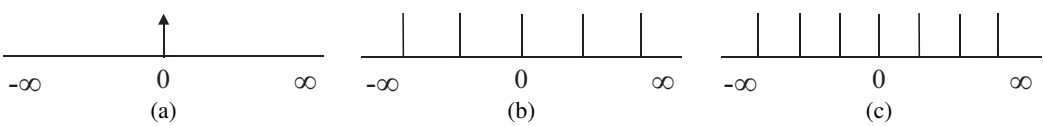


FIGURE 4.22

(a) Delta function. (b) Comb function. (c) Fourier transform of comb function.

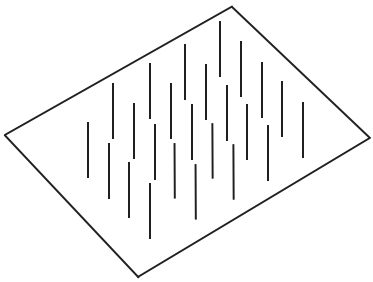


FIGURE 4.23

Bed of nails.

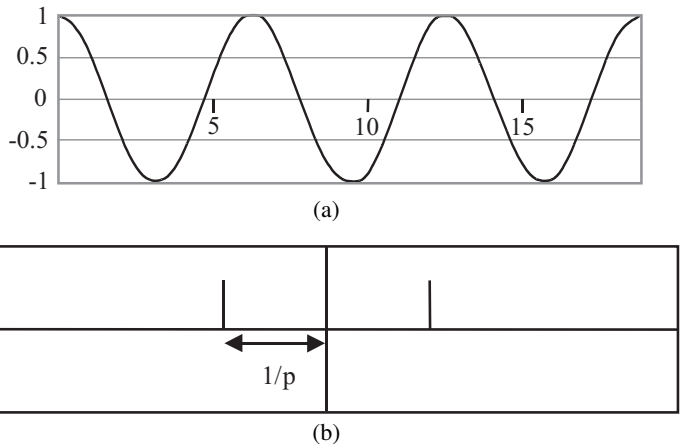


FIGURE 4.24

(a) Cosine with pitch p. (b) Fourier transform of cosine, two delta functions at, with frequency 1/p.

The comb is a 1D array of delta functions. A representation is shown in Figure 4.22b. The Fourier transform is another comb with spacing equal to the inverse of the original function (Figure 4.22c).

The 2D comb function is sometimes called a bed of nails; a representation is shown in Figure 4.23. The Fourier transform of the bed of nails is another bed of nails with spacing equal to the inverse of the original bed of nails spacing.

The Fourier transform of the cosine in Figure 4.24a is a pair of delta functions at $\pm 1/p$ from the origin, as shown in Figure 4.24b.

4.8 Image System Response

To prevent aliasing in an image the spatial frequency content has to be limited prior to sampling. This means the optical system has to control the spatial frequency content. The analysis of the optical system is based on Figure 4.25. The top half of Figure 4.25 represents the capture process in the spatial domain. The bottom half represents the frequency

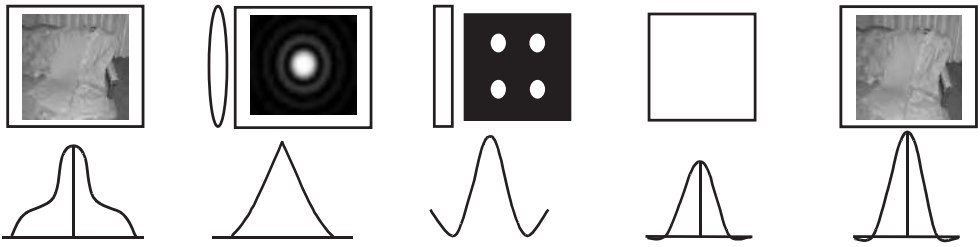


FIGURE 4.25

Capture in (top) the spatial domain and (bottom) the frequency domain.

domain in one dimension. The picture in the top of Figure 4.25a is the original object. The frequency spectrum of the original object is shown in the bottom of the figure. The top portion of Figure 4.25b shows a representation of the lens and its point spread function, an Airy disk. The bottom portion shows the lens MTF — the modulus of the Fourier transform of the lens point spread function. The rectangle shown in Figure 4.25c represents a four-spot antialiasing filter. In one dimension the MTF of the antialiasing filter is a cosine with negative values reflected across the horizontal axis. The first zero is at the Nyquist frequency, 1/(2p). The square shown in Figure 4.25d is a representation of the pixel active area. The MTF, shown in the bottom (the modulus of the Fourier transform), is a sinc function. If the pixel active area is one pixel pitch wide the MTF goes to zero at the sampling frequency. The image shown in Figure 4.25e is the result of convolving the image with the component point spread functions. The graph in the bottom of the figure is the result of cascading the MTFs for all of the point spread functions (multiplicative combination). This procedure describes the optical processing applied before the image is sampled.

Figure 4.26 represents sampling. It is also divided in half — the left represents the spatial domain and the right represents the frequency domain. Only one scan line will be included in this discussion. The graph shown in Figure 4.26a is the profile of a line of pixels; the frequency spectrum is shown in the bottom of the figure. The graph shown in Figure 4.26b is the same line of pixels that has been convolved with the combined point spread functions as described by the capture process in Figure 4.25. The frequency spectrum of the line is shown in the bottom of Figure 4.26b. The function in the top of Figure 4.26c is a comb, and in the bottom is the corresponding Fourier transform, another comb with the inverse pitch. Figure 4.26d shows a representation of how the line is sampled; the profile on the top of the figure is multiplied by the comb from Figure 4.26c to produce the sampled

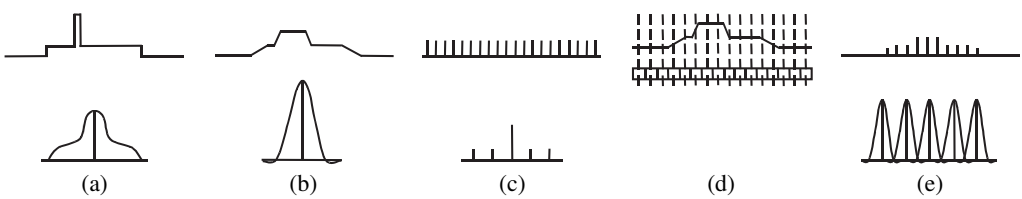


FIGURE 4.26

Sampling in (top) the spatial domain and (bottom) the frequency domain.

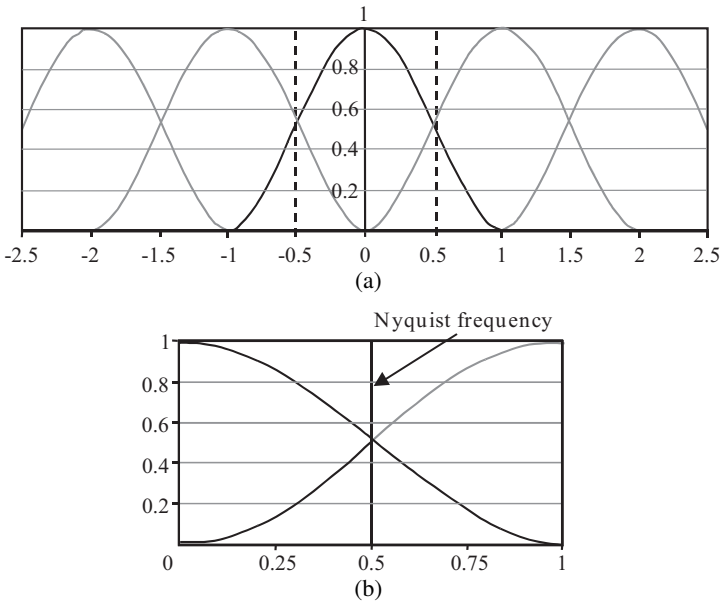


FIGURE 4.27

(a) Replicated spectrum, (b) mirrored spectrum.

image on the top of Figure 4.26e. In the frequency domain, the frequency spectrum in Figure 4.26b is convolved with the comb to produce the repeating frequency spectrum shown in the bottom of Figure 4.26e. If the image spatial frequencies are above the Nyquist frequency, the repeating spectrums overlap the baseband and the high spatial frequency image is aliased to a low-frequency image. The dark vertical bars in Figure 4.27a are at the Nyquist frequency. The Nyquist frequency is also called the fold frequency because the overlap of the replicated spectrum is identical to the baseband spectrum mirrored across the Nyquist frequency (Figure 4.27b). The grey section of the curve in Figure 4.27a is the replicated spectrum that actually produced the aliased content.

Converting the sampled image back into an analog representation is called reconstruction or desampling. This is the last, and usually overlooked, step in the display of a sampled image. An artifact that looks similar to aliasing results without reconstruction. Reconstruction is treated in more detail in Section 4.9.

4.8.1 System Modulation Transfer Function

Most of the analysis of aliasing involves capture. Scene frequency content above the Nyquist frequency has to be suppressed prior to sampling in order to prevent aliasing. Figure 4.25 shows that the lens MTF, the antialiasing filter MTF (AAfilterMTF) and the pixel MTF have to be cascaded to analyze the MTF prior to sampling. According to

$$\mathrm{MTF}_{capture} = \mathrm{MTF}_{lens} \cdot \mathrm{MTF}_{AAfilter} \cdot \mathrm{MTF}_{pixel} \tag{4.10}$$

the lens MTF, antialiasing filter MTF, and pixel MTF are multiplied together at each spatial frequency to produce the capture MTF.

The lens MTF can be evaluated using

$$\mathrm{MTF}_{lens}(\phi) = \frac{2}{\pi}\left(\phi - \cos(\phi)\sin(\phi)\right) \tag{4.11}$$

$$\phi(r,s) \cong \arccos\left(\lambda N\sqrt{r^2+s^2}\right) \tag{4.12}$$

where r is spatial frequency on the horizontal axis, s is spatial frequency on the vertical axis, N denotes lens f/#, λ is wavelength in the same units as r and s, and ϕ is an intermediate variable. This is the MTF for a perfectly designed and manufactured lens. The measured lens MTF can be substituted for this MTF if it is available, or additional terms can be added to this expression to account for field angle and defocus (Reference [5], p. 378).

The MTF of a standard four-spot antialiasing filter is next. The first zero of the filter MTF is at the Nyquist frequency if the spot pitch is equal to the pixel pitch. The MTF of the four-spot filter is a 2D cosine that goes to zero at 1/(spot pitch). This is not difficult to evaluate but a more general solution for any number of spots uses the properties of the delta function. Complex variable math is required, but spreadsheet programs and math programs that handle complex variables are common and make a more general solution fairly easy.

As shown in Equation 4.9, the Fourier transform of a delta function is an exponential. Thus, the Fourier transform of the four-spot filter is the sum (average) of four exponentials:

$$\begin{aligned} F\left(\mathrm{fourspot}(x,y)\right) = \frac{1}{4}\Big(&e^{-j2\pi(a_0 r+b_0 s)} + e^{-j2\pi(a_1 r+b_1 s)} \\ &+e^{-j2\pi(a_2 r+b_2 s)} + e^{-j2\pi(a_3 r+b_3 s)}\Big) \end{aligned} \tag{4.13}$$

The MTF is the modulus of the Fourier transform:

$$\mathrm{MTF}_{AAfilter} = \sqrt{\mathrm{Real}^2 + \mathrm{Imaginary}^2} \tag{4.14}$$

The MTF of a filter with any number of spots uses the same technique, except exponentials corresponding to each of the spots are summed (averaged). If possible, the spots should be placed symmetrically around the origin to avoid phase terms.

The pixel aperture also affects the system MTF. The MTF of the pixel aperture is a 2D sinc function (Equation 4.7) with the first zero at a frequency of $(1/a)$ on the horizontal axis where a is the width of the pixel on the x axis. Similarly, the first zero on the vertical axis is at $1/b$. The area of the standard form of the rect function becomes the peak value of the Fourier transform. The point spread function has to be scaled by $1/a$ so it has an area of one. The MTF will then peak at 1.0. Alternatively, the sinc function can be scaled by $1/a$ so the peak is 1.0:

$$\mathrm{MTF}_{pixel} = \frac{1}{ab}\cdot\frac{\sin(\pi a r)}{\pi r}\cdot\frac{\sin(\pi b s)}{\pi s} \tag{4.15}$$

Figure 4.28 is an example of an MTF cascade. In this case, the MTF is only analyzed on one axis. The actual system spatial frequencies are used because the lens MTF is based on lens parameters; these cannot be scaled to the sampling frequency unless a particular sampling frequency is chosen. The imager in the example has a two mm pixel pitch, the lens is set at f/4 and the antialiasing filter is a four spot filter with a spot separation equal to

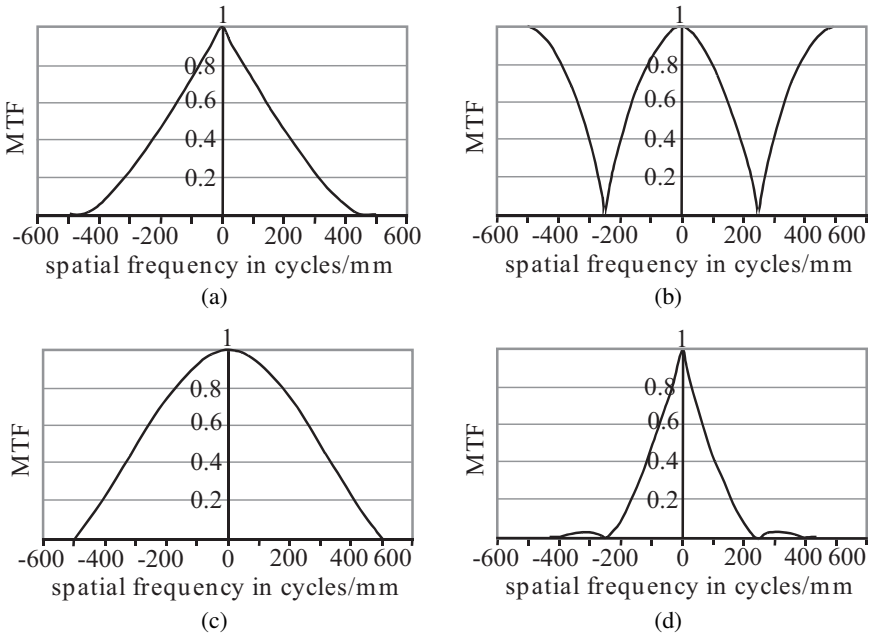


FIGURE 4.28

Cascade system response: (a) lens MTF, (b) filter MTF, (c) pixel MTF, (d) system MTF.

the pixel pitch. The lens MTF shown in Figure 4.28a is calculated using Equation 4.11. The antialiasing filter MTF response shown in Figure 4.28b is calculated using Equation 4.13. The pixel MTF shown in Figure 4.28c is calculated using Equation 4.15. Finally, the system MTF shown in Figure 4.28d is the result of multiplying the three component MTFs, spatial frequency by spatial frequency. Notice the blur filter MTF goes to zero at the Nyquist frequency for the sensor, 250 cycles per mm. The system MTF past the Nyquist frequency is suppressed by the pixel MTF and the lens MTF.

The area under the curve beyond the Nyquist frequency can be used as a merit function to optimize system performance; the minimum area provides the best performance. If the analysis is done in 2 dimensions the weighted volume beyond Nyquist can be used to optimize system performance. Weighting is required because the volume is not a linear function of radius. A weighting function can also be used to give more influence to spatial frequencies that are more visible to the eye.

4.9 Reconstruction

In general, a sampled image has to be resampled for display. If the display has enough samples it can emulate a nonsampled display. The process of converting a sampled image to a continuous image is called reconstruction or desampling [8]. Without reconstruction, an artifact, sometimes called interpolation error, may appear. The artifact is a variation in modulation that occurs at spatial frequencies below the Nyquist frequency.

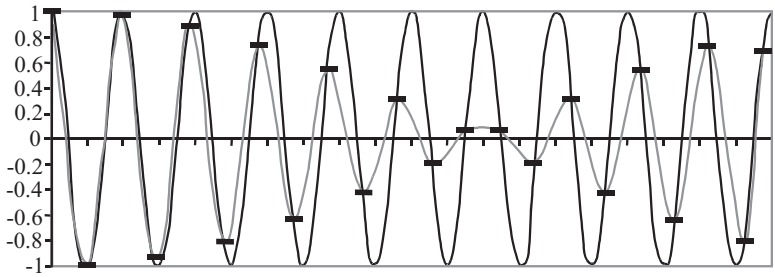


FIGURE 4.29

Sampled sine wave, ten cycles and twenty-two samples.

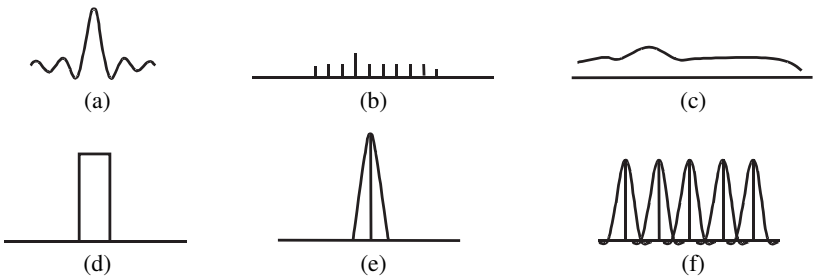


FIGURE 4.30

Reconstruction: (a-c) spatial domain, (e-f) frequency domain.

To illustrate the problem, a ten-cycle sine wave is shown in Figure 4.29. Samples are taken at the evenly spaced dark squares. The output image is displayed by drawing a smooth curve through the samples. Notice the amplitude is high when the peak and valley samples are in phase with the sampled points but the amplitude goes down when the peaks and valleys are out of phase with the sampled points. The minimum amplitude occurs when two samples are split across a peak or valley.

The modulation envelope is caused by the replicated versions of the image spatial frequency content. Multiplying the image frequency spectrum by a rect (Figure 4.30d) as wide as the first order spectrum will eliminate the replicas (Figure 4.30f). This can be accomplished in the spatial domain by convolving the image with a 2D sinc as shown in one dimension in Figure 4.30a. The first zero of the sinc function is at 1/p where p is the capture sampling pitch. The convolution is a continuous function so it reconstructs an analog image. The low modulation sections of the sampled image are boosted in the convolution by modulation at the tails of the sinc function. The reconstructed sine wave at Figure 4.31c is an analog reproduction of the original image. In practice, the convolution is only evaluated at the points required for the displayed image.

Computational speed is an issue with this technique because the sinc is infinitely wide. This can be handled, with some image quality penalty, by windowing with suitable functions (e.g., Hamming, Hann, etc.). The operation is still computationally expensive although the 2D sinc function is separable, so a 1D sinc can be applied to the rows and then to the columns. This reduces the amount of computer arithmetic required to reconstruct a

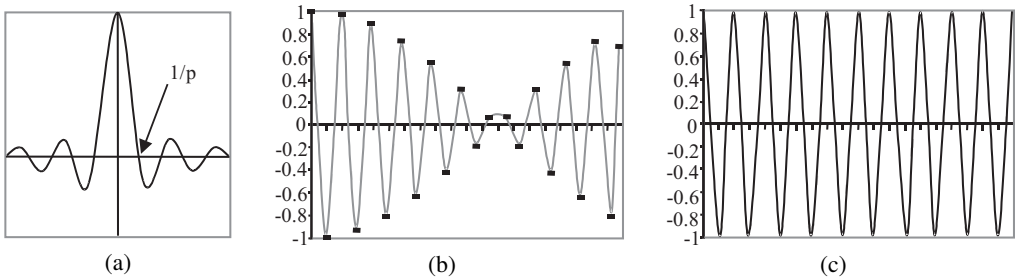


FIGURE 4.31

Reconstruction.

sampled image. Finally, the image has to be rendered for display. After reconstruction the modulation envelope is suppressed. Resampling for the display will create a new modulation envelope if the image is not low pass filtered based on the display resolution.

The number of display pixels required depends on how much interpolation error is acceptable. The display pixel count required can be determined from Figure 4.32, a graph of modulation due to interpolation error. It is assumed the sine wavelength is not an even multiple of the number of pixels so the phase of the samples will precess from a sample at a peak or valley to samples split evenly across a peak or valley. Figure 4.33 shows the origin of this graph; at some point the sine wave will be sampled and displayed at a value of 1 or -1 (peak or valley) and at some other point the samples will be split across the peak or valley. At 1/2 cycle per display pixel, the sine wave envelope can go from ± 1 at the peak and valley samples to zero when the samples are split across the peak or valley. At 1/4 cycle per display pixel the sine only drops to 70% when the samples are split across a sine wave peak so the modulation envelope is reduced to 30%. At 1/6 cycle per display pixel, the modulation is reduced to approximately 10%. To reduce the modulation envelope to a particular value the reconstructed image can be low pass filtered to limit spatial frequencies to a particular number of cycles per display pixel or a higher resolution display can be used if reducing the displayed resolution is not an option.

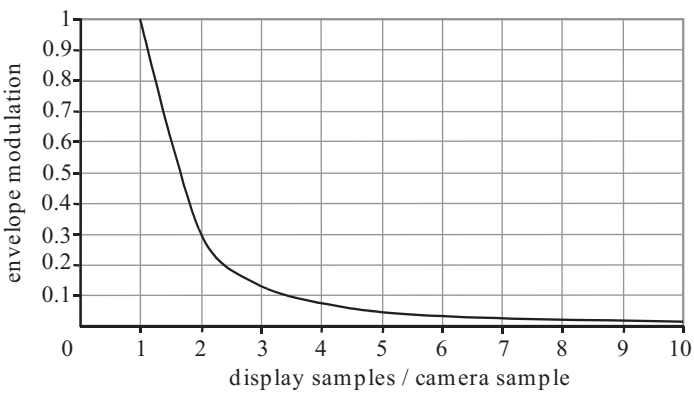


FIGURE 4.32

Modulation based on interpolation error.

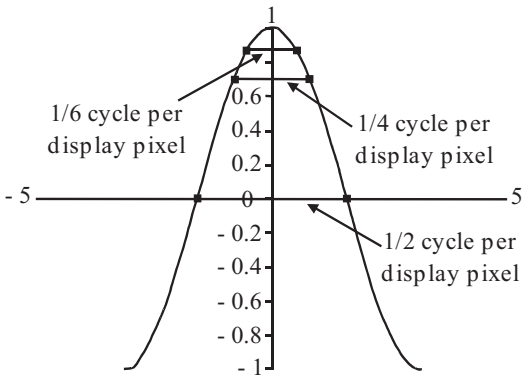


FIGURE 4.33

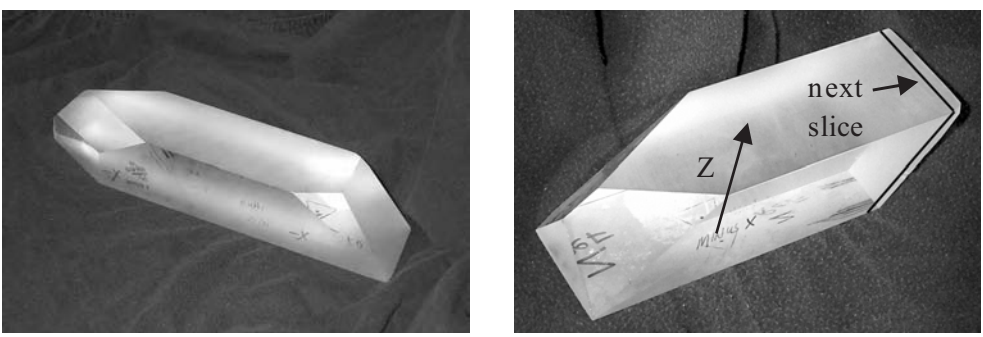
Determine envelope modulation.

4.10 Construction

The birefringent antialiasing filter is the most common antialiasing filter and the four spot square pattern is the most common pattern. An example of the four spot square pattern is shown on the right side of Figure 4.10c. The single spot shown the left side of Figure 4.10c is a representation of the point spread function produced by a lens from an object point. The conventional antialiasing filter design makes this into four spots separated by the pixel pitch. To create this pattern requires three birefringent plates cemented together. The usual material is quartz but lithium niobate and calcite have also been used.

Figure 4.34a is an example of the raw birefringent material, cultured crystalline quartz. This particular piece is an optical half section. There are less expensive forms used in the electronics industry that can be used for small filters.

The index of refraction of this material is dependent on the polarization and direction of travel through the crystal. If a spark could be set off inside the crystal, the light would radiate from the spark as shown in Figure 4.35a. Some of the rays, depending on the po-



(a) (b)

FIGURE 4.34

(a) Cultured quartz bar. (b) Cut crystal.

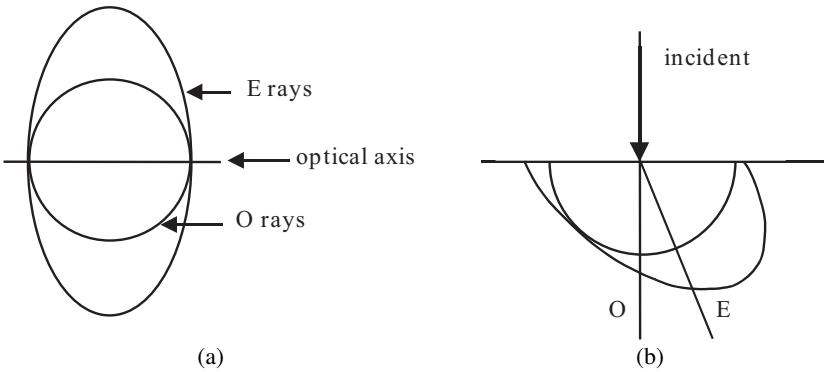


FIGURE 4.35

The *E* and *O* rays in: (a) birefringent crystal, (b) crystal cut at 45 degrees to the optical axis.

larization axis, travel at the same speed in all directions. These rays are called the ordinary or *O rays* [9], [10]. The speed of the rest of the rays depends upon direction in the crystal. These rays are called the extraordinary or *E rays*. The *E* and *O* rays are orthogonally polarized. Both rays travel through the quartz at the same speed on one axis called the optical axis. The maximum speed difference is on the orthogonal axis. If the crystal is cut at 45 degrees to the optical axis (Figure 4.35b) an incident beam will separate into *E* and *O* rays. The rays separate at 5.8 micrometers per millimeter of material. A crystal that has been cut at 45 degrees to the optical axis is shown in Figure 4.34b. The optical axis, *Z*, is perpendicular to the base. A ray incident on a slice of this crystal is shown in Figure 4.10a, illustrating how the ray splits into an ordinary ray and an extraordinary ray. The two rays are separated but parallel to each other when they leave the crystal. Figure 4.10b represents the slice of Figure 4.10a with the outline of a plate that will be included in an antialiasing filter. The line with two circles represents the projection of the optical axis in the plane of the plate. In this case the optical axis is at 45 degrees to the edge of the plate in addition to a tilt of 45 degrees in and out of the page.

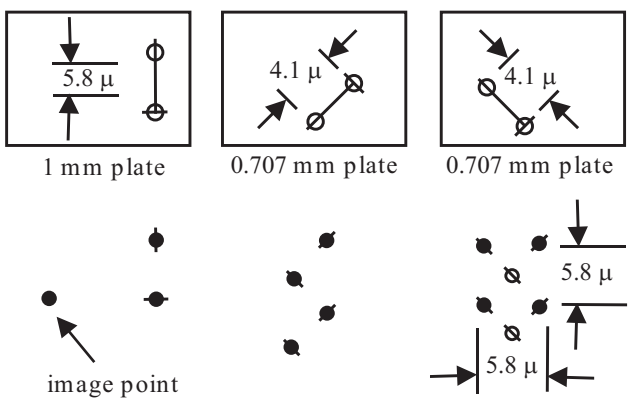
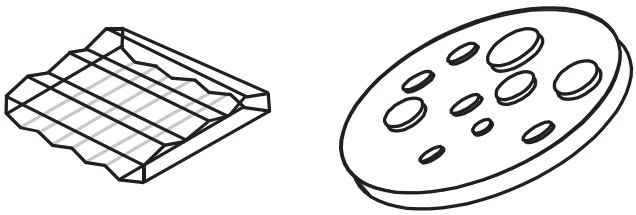


FIGURE 4.36

Four-spot filter construction.



(a) (b)

FIGURE 4.37

(a) Pleat filter. (b) Phase-noise filter.

To build a square pattern filter, three plates are cut with different angles of rotation between the edge of the plate and the projection of the optical axis. The plates are cemented together to produce a filter. A representation of the three plates in Figure 4.36 shows the angle of the edge of the plate to the projection of the optical axis and the resulting spot patterns. The line through the spots indicates the polarization of the spots. In practice the plate thickness is chosen on the basis of the pixel pitch but for the purpose of illustration the first plate is chosen to be one millimeter thick and the remaining plates are chosen to produce a square pattern. Notice the first plate produces two spots. It takes two steps to copy these spots and make a square pattern. The next plate is cut 45 degrees to the optical axis. At 45 degrees, the spots from the first plate have equal 45 degree components as seen from the second plate, so both spots are doubled by the second plate. Notice the third plate does not produce additional spots because its optical axis is 90 degrees to the second plate optical axis. The spots on the right do not move because they pass through as the ordinary ray, but the spots on the left are shifted [11].

There is an alternative design for this filter that uses a retarder for the middle plate. The retarder is still made out of quartz but it is cut differently. It effectively depolarizes the light that reaches the third plate, so the third plate replicates the first two spots to create a square pattern. There are other antialiasing filter types that use refraction or diffraction to reduce

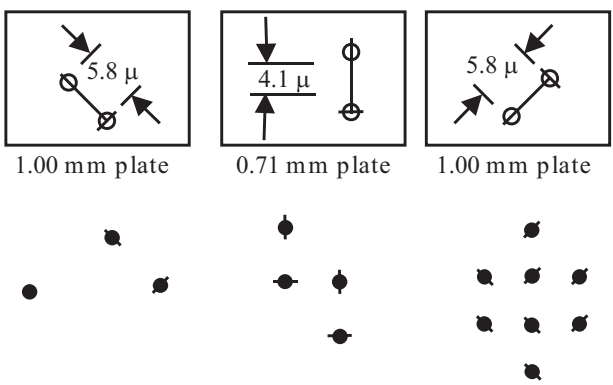


FIGURE 4.38

Eight-spot filter.

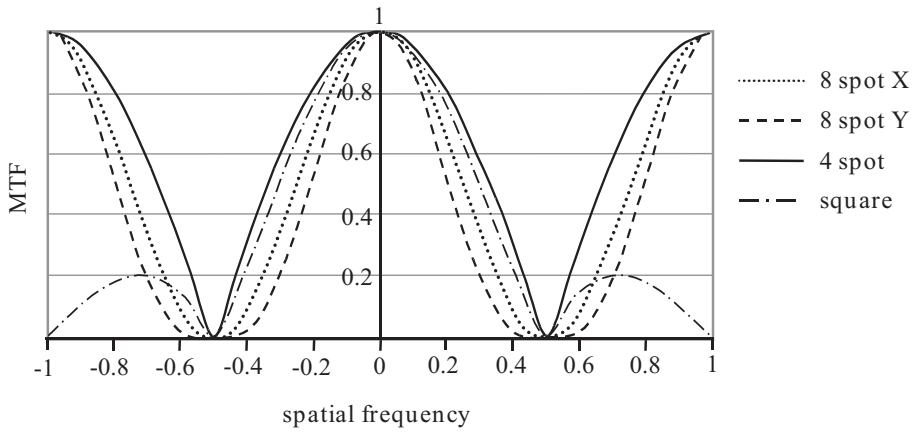


FIGURE 4.39

Comparison of different filters.

the high frequency content of an image. Figure 4.37a is a pleated filter, which produces a four-spot pattern [12]. This filter should be co-optimized with the lens. Figure 4.37b is a phase-noise filter. It has different size spots with a thickness on the order of the wavelength of light. The spots change the wave front entering the lens to increase the size of the point spread function and reduce the image high-frequency content [13], [14]. It is possible to make this filter wavelength dependent so the point spread function can be adjusted to match the color channel Nyquist domain.

It is possible to build an eight-spot and a seven-spot birefringent filter with three plates. The details of an eight-spot filter are shown in Figure 4.38. Figure 4.39 shows a comparison of the MTF of a square pattern filter, four-, and eight-spot filter. The filter point spread functions are chosen to have a zero at a half cycle per sample. Notice that the four-spot filter maintains MTF below the Nyquist frequency better than the other filters, but the MTF suppression past Nyquist is not as good as the other filters. The eight-spot filter starts to fill in the four-spot filter, so the eight-spot performance has some of the attributes of the four-spot filter and some of the attributes of a filter with a square pattern point spread function. In the spatial domain, the eight-spot filter pattern is taller than it is wide, so the X-axis MTF is different than the Y-axis MTF. The four-spot filter works well when the lens and pixel aperture control the MTF past the Nyquist frequency. If the antialiasing filter has to control the MTF past Nyquist, one of the other patterns may work better.

4.11 Testing

Antialiasing filters can be tested with a point source, a good quality photographic objective, and a microscope. The photographic objective is used to image a point source. The image-forming beam passes through the antialiasing filter and the resulting image is viewed with a microscope (Figure 4.40). If the antialiasing filter is a four-spot birefringent filter,

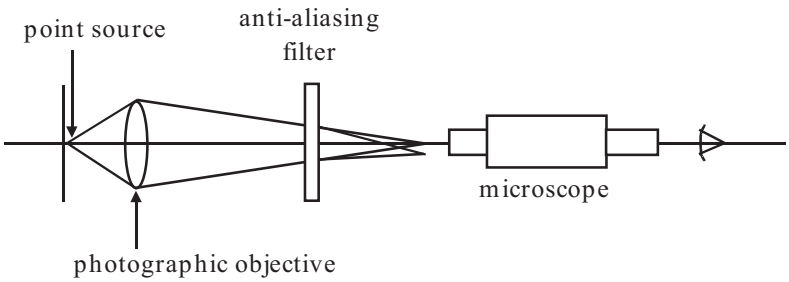


FIGURE 4.40

Viewing antialiasing point spread function.

the image will be four copies of the lens point spread function in a square pattern. The lens point spread function has to be small relative to the size of the filter pattern or the image will just be a square patch of light.

There are a number of issues with this technique. When cameras had six or twelve micrometer pixels, this technique was easy to set up without much attention to detail but current consumer cameras are rapidly moving toward two micrometer pixels or less. This pixel size is pushing the limits of optical microscopes and photographic objectives. To image two micrometer pixels, the objective has to be diffraction limited at f/2 or less. The point source has to fill the entrance pupil of the lens or the lens point spread function will be larger because the lens is effectively operating at a smaller aperture. Finally, the microscope objective has to be aligned carefully and the NA has to be large enough to accept the f/2 beam.

If the antialiasing filter is not available to test separately, or if a system test is required, a circular chirp pattern or a zone plate can be used to analyze the aliasing potential of a camera or other digital imaging system. The spatial frequency of a circular chirp increases linearly with radius from the center of the pattern. The equation is:

$$u = \left(\cos(2\pi q^2) + 1\right)/2 \tag{4.16}$$

where u is the reflectance or transmission of the chirp and q is the radius from the center of the chirp.

A chirp pattern is illustrated in Figure 4.41a. All of the following chirp images are simulated captures that are processed as if they were camera captures. Figure 4.41b is a red or blue channel chirp pattern from a Bayer CFA that was interpolated using bilinear interpolation. Notice the aliasing at the Nyquist frequency and the pronounced aliasing at the sampling frequency. Figure 4.41c shows an interpolated green channel without an antialiasing filter. Figure 4.41d is an interpolated red or blue channel with an antialiasing filter. The improvement with the filter is clear. Figure 4.41e and Figure 4.41f show the results of adaptive interpolation. In this case, the red or blue channel does not look as good as the bilinear interpolation version because the color errors are not apparent in the black and white image. The green channel is very clearly better using adaptive interpolation. To show the image quality improvement with adaptive interpolation compared to bilinear interpolation, the images are compared in Lab color space (Figure 4.42). Lab is an opponent

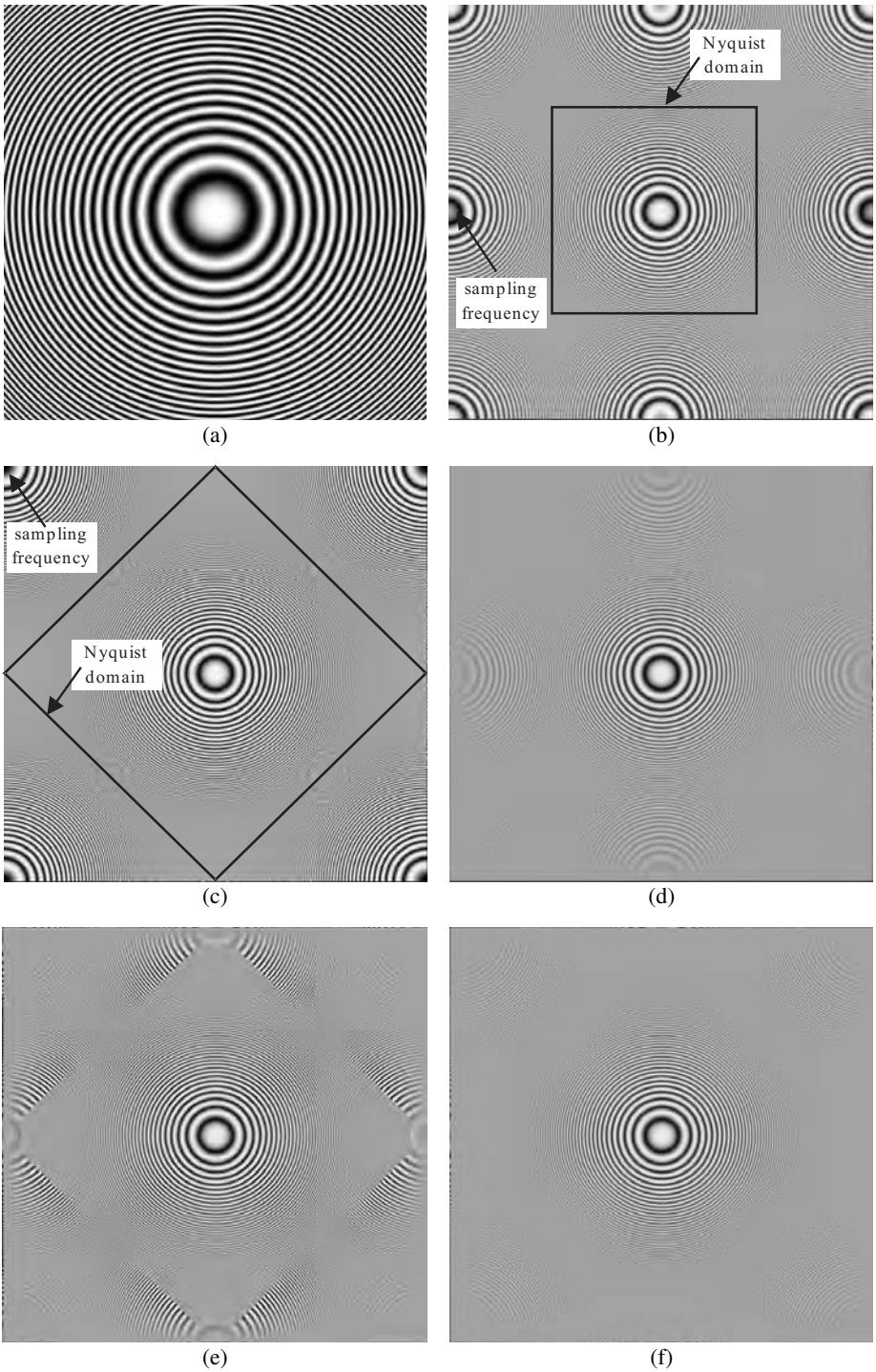


FIGURE 4.41

Experimentation using a chirp pattern: (a) original image, (b) red or blue channel interpolated from Bayer pattern without antialiasing filter, (c) green channel interpolated from Bayer pattern without antialiasing filter, (d) red or blue channel interpolated from Bayer pattern with antialiasing filter, (e) red or blue channel interpolated from Bayer pattern with antialiasing filter and adaptive interpolation, and (f) green channel interpolated from Bayer pattern with antialiasing filter and adaptive interpolation.

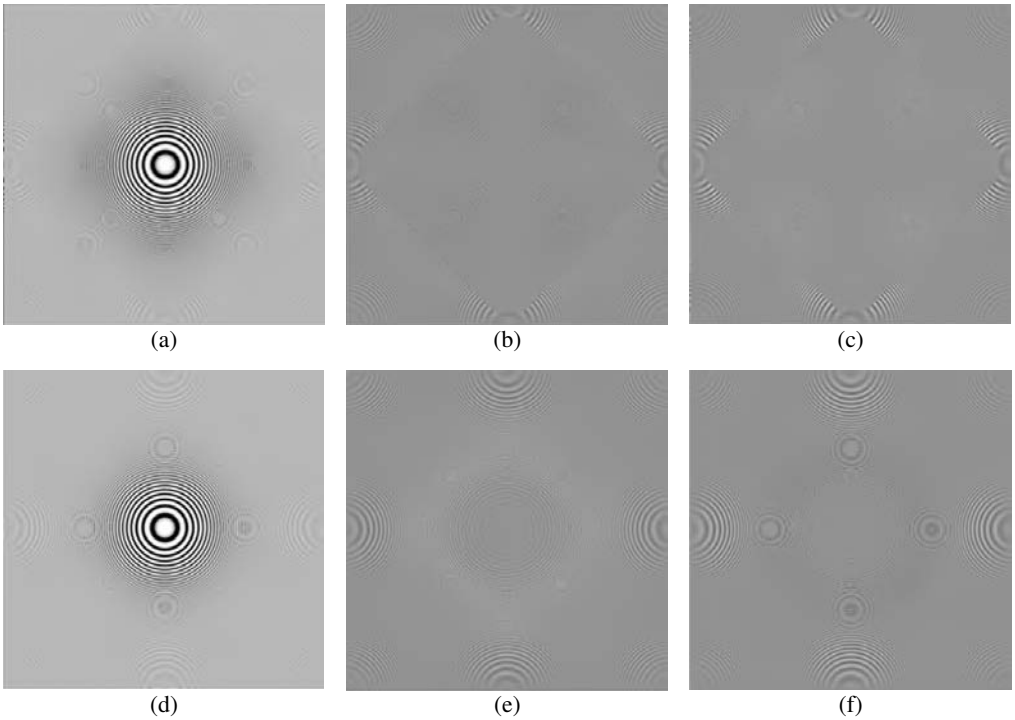


FIGURE 4.42

Interpolation of a chirp pattern image in Lab space: (a-c) adaptive interpolation, (d-f) bilinear interpolation; (a,d) *L* channel, (b,e) *a* channel, (c,f) *b* channel.

color space; negative values of *a* represent green and positive values represent magenta whereas negative values of *b* represent blue and positive values represent yellow. The *a* and *b* channels are both zero for gray images. As illustrated, the *a* and *b* images are scaled to all positive values between zero and 255. The image value 128 corresponds to *a* or *b* equal to zero. Areas darker or lighter than 128 indicate more color content. Notice the images with adaptive interpolation have much less color detail. This indicates a much lower aliased color level in the adaptively interpolated image.

4.12 Conclusions

Analysis of antialiasing filter performance must include all capture system parameters, particularly the pixel aperture size, lens performance, and interpolation technique. Ideally the antialiasing filter and the interpolation technique should be co-optimized to maximize system MTF below the Nyquist frequency and minimize system MTF above the Nyquist frequency.

Antialiasing filters have been in electronic and digital cameras for over forty years. A good compromise has been reached between reduced MTF below the Nyquist frequency

and reduced aliasing because of scene content above the Nyquist frequency. Reducing the lens MTF has always been an attractive alternative to the expense of an antialiasing filter, but variation in lens MTF with f/number and focus makes it difficult to achieve consistent antialiasing with this alternative. Some consumer cameras with pixel sizes of two micrometers or less are not including an antialiasing filter because the lens MTF is low enough beyond the Nyquist frequency to suppress aliasing. Pixel size is still being driven lower although the inherent noise due to small pixels may stop this trend. If pixel size drops to one micrometer and lens f/numbers remain at about f/3 camera systems certainly will not need an antialiasing filter. At this pixel pitch even a diffraction limited lens does not have a high enough MTF at the Nyquist frequency to produce aliasing.

Acknowledgments

The author is especially grateful to Bruce Pillman and John Hamilton, both of Eastman Kodak Company. Bruce insightfully suggested some of the included topics and was always available to discuss content. John helped formulate the math; any errors that may have gotten by John's watchful eye are the author's.

References

[1] D.R. Williams, "Aliasing in human foveal vision," *Vision Research*, vol. 25, no. 2, pp. 195–205, 1985.

[2] J.E. Greivenkamp, "Color dependent optical prefilter for the suppression of aliasing artifacts," *Applied Optics*, vol. 29, no. 5, pp. 676–684, 1990.

[3] J.W. Goodman, *Introduction to Fourier Optics*. New York: McGraw-Hill, 1968, reissued 1988.

[4] P. Dillon, D.M. Lewis, and F.G. Kaspar, "Color imaging system using a single CCD area array," *IEEE Journal of Solid-State Circuits*, vol. 13, no. 1, pp. 28–33, February 1978.

[5] W.J. Smith, *Modern Optical Engineering*. New York: McGraw-Hill, 3rd edition, 2000.

[6] E. Hecht and A. Zajac, *Optics*. Reading, MA: Addison-Wesley Longman, 3rd edition, 1997.

[7] J.D. Gaskill, *Linear Systems, Fourier Transforms and Optics*. New York: John Wiley & Sons, 1978.

[8] R.E. Volmerhausen and R.G. Driggers, *Analysis of Sampled Imaging Systems*. Bellingham, WA: SPIE Press, 2000.

[9] D. Clarke and J.F. Grainger, *Polarized Light and Optical Measurement*. Oxford: Pergamon Press Ltd., 1971.

[10] A.F. Jenkins and H.E. White, *Fundamentals of Optics*. New York: McGraw-Hill, 2001.

[11] D. Kessler, A. Nutt, and R. Palum, "Anti-aliasing low-pass blur filter for reducing artifacts in imaging apparatus," U.S. Patent 6 937 283, August 2005.

[12] R. Palum, "Optical blur filter having a four-feature pattern," U.S. Patent 6 326 998, December 2001.

[13] K. Sayanagi, "Phase noise filter and its application to photography and photolithography," U.S. Patent 2 959 105, August 1960.

[14] J. Hirsh, J.F. Revelli, and A. Nutt, "Phase-noise type broad spectral bandwidth optical low-pass anti-aliasing filter," U.S. Patent 6 040 857, March 2000.

5

Spatio-Spectral Sampling and Color Filter Array Design

Keigo Hirakawa and Patrick J. Wolfe

5.1 Introduction

Owing to the growing ubiquity of digital image acquisition and display, several factors must be considered when developing systems to meet future color image processing needs, including improved quality, increased throughput, and greater cost-effectiveness [1], [2], [3]. In consumer still-camera and video applications, color images are typically obtained via a spatial subsampling procedure implemented as a color filter array (CFA), a physical construction whereby only a single component of the color space is measured at each pixel location [4], [5], [6], [7]. Substantial work in both industry as well as academia has been dedicated to postprocessing this acquired raw image data as part of the so-called *image processing pipeline*, including in particular the canonical *demosaicking* task of reconstructing a full color image from the spatially subsampled and incomplete data acquired using a CFA [8], [9], [10], [11], [12], [13]. However, as we detail in this chapter, the inherent shortcomings of contemporary CFA designs mean that subsequent processing steps often yield diminishing returns in terms of image quality. For example, though distortion may be masked to some extent by motion blur and compression, the loss of image quality resulting from all but the most computationally expensive state-of-the-art methods is unambiguously apparent to the practiced eye. Refer to Chapters 1 and 3 for additional information on single-sensor imaging fundamentals.

As the CFA represents one of the first steps in the image acquisition pipeline, it largely determines the maximal resolution and computational efficiencies achievable by subsequent processing schemes. Here we show that the attainable spatial resolution yielded by a particular choice of CFA is quantifiable, and propose new CFA designs to maximize it [14], [15]. In contrast to the majority of the demosaicking literature, we explicitly consider the interplay between CFA design and properties of typical image data, and its implications for spatial reconstruction quality. Formally, we pose the CFA design problem as simultaneously maximizing the allowable spatio-spectral support of luminance and chrominance channels, subject to a partitioning requirement in the Fourier representation of the sensor data. This classical *aliasing-free* condition preserves the integrity of the color image data and thereby guarantees exact reconstruction when demosaicking is implemented as demodulation (demultiplexing in frequency).

Surprisingly, from this perspective we can show the suboptimality of CFA designs based on pure tristimulus values [15]—a standard design approach long taken by industry, particularly as manifested by the popular Bayer pattern [4]. Such designs are less resilient to spatial aliasing as image resolution increases, requiring both stronger assumptions about the image data as well as more computationally demanding nonlinear demosaicking methods to avoid reconstruction artifacts. Here our interest lies in quantifying the trade-offs between performance and complexity for different classes of CFA design; we consider the purely linear reconstruction of typical images as an indication of baseline performance, and interpret the resultant degree of aliasing as providing a measure of the maximally attainable spatio-spectral resolution.

As an alternative to existing CFA patterns, we provide a constructive method to generate feasible CFA designs that exhibit robustness to prior assumptions on color channel bandlimitedness and yield high performance while implying only low complexity for subsequent processing steps in the imaging pipeline. Because our emphasis is on the efficiencies of the overall color image acquisition pipeline, we omit an explicit comparison of demosaicking strategies. However, our analysis yields a general class of linear demosaicking methods that provide state-of-the-art performance and enjoy complexity comparable to simple bilinear interpolation. In addition, our proposed CFA designs are also designed for increased noise robustness: the color filters themselves are panchromatic, alleviating difficulties in low-light conditions, and the linear reconstruction methods we propose can also be expected to enable more tractable noise modelling [15].

The remainder of this chapter is organized as follows. We begin in Section 5.2 by examining the spatio-spectral properties of typical CFA designs in the Fourier domain, and discuss their susceptibility to aliasing. We propose in Section 5.3 a constructive method to specify a physically realizable CFA pattern in terms of its spatio-spectral properties. The resultant CFA designs admit fast, optimal linear reconstruction schemes, which we outline in Section 5.4. In Section 5.5 we give several explicit examples of these new patterns, and provide empirical evaluations on standard color image test sets. We summarize and conclude with a discussion in Section 5.6.

5.2 Spatio-Spectral Analysis of Existing Patterns

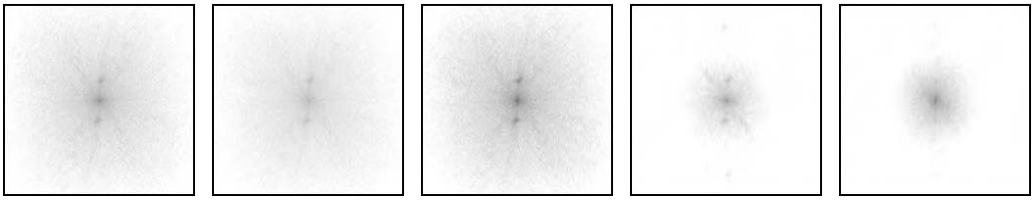
In this section, the spatio-spectral properties of the sampling induced by existing CFA patterns are analyzed. In single-sensor cameras, the pixel sensor at each spatial location is equipped with a color filter, a physical device whose pigments absorb a portion of the electro-magnetic wave in the visible spectrum while passing the rest to the photosensitive element beneath this filter. The measured value at each location is therefore an inner product resulting from a spatio-temporal integration of the incident light over each pixel's physical area and exposure time, taken with respect to the corresponding color filter's spectral response. This is similar to the acquisition process in the retina, where each cone measures the intensity of the light with respect to its spectrally-shifted response [4], [5], [6], [7]. Because the spectral response functions of the cones can be taken to span a three-dimensional space, and cone and sensor measurements are largely proportional to the intensity of the light (i.e., linear), the observed light can be uniquely represented (up to linear transformation) by a color triple. We therefore adopt the standard convention and identify these filters by their color names such as red, green, and blue—though these may not be synonymous with perceived color, which is a function of the environmental illuminant [1]. As the goal of this chapter is the identification and optimization of relevant objective metrics, rather than subjective metrics related to perception, we make no further attempt to elaborate on the issues of color science.

5.2.1 Color Filter Arrays

Here we begin with the Fourier analysis of the spatio-spectral properties of the CFA patterns [10], [13]. This spatially global perspective is a logical starting point for a number of reasons (a spatially local perspective is provided in the next section). First, color filter arrays are physical constructions that are fixed prior to image acquisition, and therefore not adapted to local image properties. Second, color filter arrays typically comprise a repetitive tiling of the image plane formed by the union of alternating color samples.[1] As we describe below, the global spatial periodicity of CFA sampling patterns may be understood in terms of lattices, with a so-called dual or reciprocal lattice determining the resultant spectral periodicity under Fourier transform. Finally, the linear reconstruction methods we consider in the interest of evaluating computation-quality trade-offs preclude adaptation to local statistics of the image under consideration.

To motivate our analysis, let us first consider the interplay between color channels of the acquired image. Let $\boldsymbol{x}(\boldsymbol{n}) = [x_r(\boldsymbol{n}), x_g(\boldsymbol{n}), x_b(\boldsymbol{n})]^T$ denote the RGB tristimulus value of the desired color image at pixel location $\boldsymbol{n} \in \mathbb{Z}^2$. Define $\boldsymbol{c}(\boldsymbol{n}) = [c_r(\boldsymbol{n}), c_g(\boldsymbol{n}), c_b(\boldsymbol{n})]^T$ as the corresponding CFA color combination, so that the measured sensor value $y(\boldsymbol{n})$ at location $\boldsymbol{n}$ can be expressed as the inner product $y(\boldsymbol{n}) = \boldsymbol{c}(\boldsymbol{n})^T \boldsymbol{x}(\boldsymbol{n})$. For the moment, we restrict our attention to $\boldsymbol{c}(\boldsymbol{n}) \in \{[1,0,0]^T, [0,1,0]^T, [0,0,1]^T\}$ as a model for CFA schemes

[1] Pseudo-random CFA patterns have also been considered in the past [7]. Despite their potential theoretical advantages, we omit them from our discussion, as the corresponding reconstruction schemes incur much greater computational expense.



(a) (b) (c) (d) (e)

FIGURE 5.1

Log-magnitude spectra of a typical color image (i.e., image *flower* here) illustrating the lowpass nature of difference channels x_α and x_β relative to x_r, x_g, and x_b. Individual spectra correspond to: (a) red channel, (b) green channel, (c) blue channel, (d) difference x_α, and (e) difference x_β.

that multiplex color samples; note that $c_r + c_g + c_b = 1$. Each pixel sensor thus measures

$$y(\boldsymbol{n}) = \boldsymbol{c}(\boldsymbol{n})^T \boldsymbol{x}(\boldsymbol{n}) = \begin{bmatrix} c_r(\boldsymbol{n}) & c_g(\boldsymbol{n}) & c_b(\boldsymbol{n}) \end{bmatrix} \begin{bmatrix} x_r(\boldsymbol{n}) \\ x_g(\boldsymbol{n}) \\ x_b(\boldsymbol{n}) \end{bmatrix} = \begin{bmatrix} c_r(\boldsymbol{n}) & 1 & c_b(\boldsymbol{n}) \end{bmatrix} \begin{bmatrix} x_\alpha(\boldsymbol{n}) \\ x_g(\boldsymbol{n}) \\ x_\beta(\boldsymbol{n}) \end{bmatrix}, \quad (5.1)$$

where $x_\alpha = x_r - x_g$ and $x_\beta = x_b - x_g$ are difference channels. As noted in References [10] and [13], this $\{x_\alpha, x_g, x_\beta\}$ representation offers an advantage over the original $\{x_r, x_g, x_b\}$ formulation; the difference channels x_α and x_β serve as a proxy for chrominance components, which enjoy rapid decay in the spatial frequency domain, whereas x_g can be taken to represent the image luminance component, which embodies edge and texture information. In fact, the Pearson correlation coefficient measured between the high-frequency components of the color channels $\{x_r, x_g, x_b\}$ is typically larger than 0.9 [8]—and because of this high degree of redundancy, it is often assumed that x_α and x_β are lowpass relative to $\{x_r, x_g, x_b\}$; see Figure 5.1.

The key observation to be gleaned from Equation 5.1 is that y constitutes a sum of the green channel x_g and the subsampled difference images $c_r \cdot x_\alpha$ and $c_b \cdot x_\beta$. In order to understand the limitations of existing color filter array designs, it is helpful to consider the geometric and algebraic structure of subsampling patterns c_r and c_b through the notion of point lattices [15]. To this end, we say a (nonsingular) sampling matrix $\boldsymbol{M} \in \mathbb{R}^{2\times 2}$ generates a lattice $\boldsymbol{M}\mathbb{Z}^2$. Certain sampling patterns c_r and c_b can in turn be rewritten as two-dimensional pulse trains using lattice notation:

$$c_r(\boldsymbol{n}) = \sum_{\boldsymbol{n}_0 \in \{\boldsymbol{m}_r + \boldsymbol{M}_r\mathbb{Z}^2\}} \delta(\boldsymbol{n} - \boldsymbol{n}_0); \qquad c_b(\boldsymbol{n}) = \sum_{\boldsymbol{n}_0 \in \{\boldsymbol{m}_b + \boldsymbol{M}_b\mathbb{Z}^2\}} \delta(\boldsymbol{n} - \boldsymbol{n}_0), \quad (5.2)$$

where $\boldsymbol{M}_r, \boldsymbol{M}_b$ are 2×2 sampling matrices; $\boldsymbol{m}_r, \boldsymbol{m}_b \in \mathbb{Z}^2$ are termed coset vectors; and $\delta(\boldsymbol{n})$ is the Kronecker delta function.[2] Lattices themselves admit the notion of a Fourier transform as specified by a dual lattice $2\pi\boldsymbol{M}^{-T}\mathbb{Z}^2$; if we define $Y(\boldsymbol{\omega})$ as the Fourier transform (in angular frequency $\boldsymbol{\omega}$) of sensor data $y(\boldsymbol{n})$, it follows from Equations 5.1 and 5.2 that

[2] In fact, Equation 5.2 represents a special case in which sampling patterns c_r and c_b are each themselves lattices. More generally, they are defined in terms of unions of lattice cosets [15]; however, this does not change the fundamentals of our present discussion.

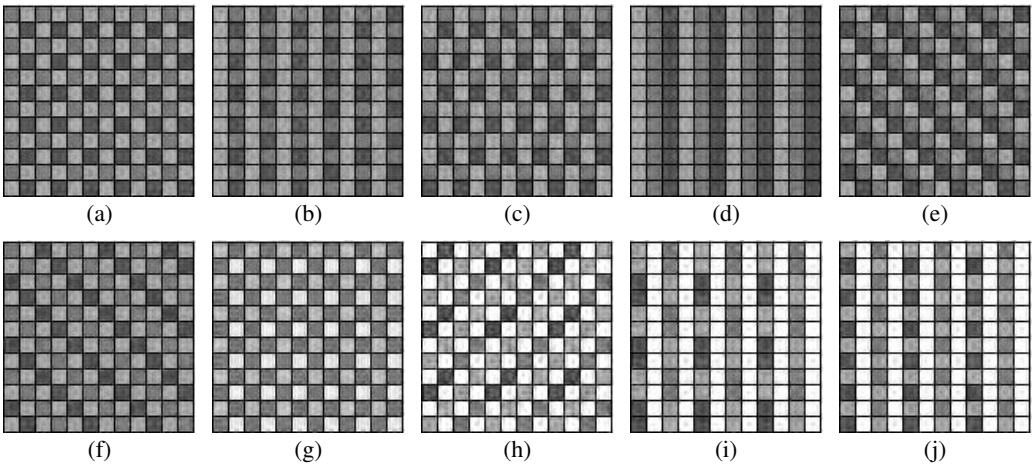


FIGURE 5.2 (See color insert.)

Examples of existing CFAs: (a) Bayer [4], (b) Yamanaka [5], (c) Lukac [7], (d) vertical stripes [7], (e) diagonal stripes [7], (f) modified Bayer [7], (g) cyan-magenta-yellow, (h) Kodak I [16], (i) Kodak II [16], (j) Kodak III [16].

$Y(\boldsymbol{\omega})$ over the region $[-\pi,\pi)\times[-\pi,\pi)$ is given by

$$Y(\boldsymbol{\omega}) = X_g(\boldsymbol{\omega}) + |\det(\boldsymbol{M}_r)|^{-1} \sum_{\boldsymbol{\lambda}_r\in\{2\pi\boldsymbol{M}^{-T}\mathbb{Z}^2\cap[-\pi,\pi)^2\}} e^{-j\boldsymbol{m}_r^T\boldsymbol{\omega}} X_\alpha(\boldsymbol{\omega}-\boldsymbol{\lambda}_r)$$
$$+|\det(\boldsymbol{M}_b)|^{-1} \sum_{\boldsymbol{\lambda}_b\in\{2\pi\boldsymbol{M}^{-T}\mathbb{Z}^2\cap[-\pi,\pi)^2\}} e^{-j\boldsymbol{m}_b^T\boldsymbol{\omega}} X_\beta(\boldsymbol{\omega}-\boldsymbol{\lambda}_b). \tag{5.3}$$

The key point of Equation 5.3 is that these dual lattices specify the *carrier* frequencies $\{\boldsymbol{\lambda}_r,\boldsymbol{\lambda}_b\}$ about which spectral copies of the difference channels x_α and x_β are replicated in the Fourier domain. The popular Bayer CFA [4], for instance, can be specified as $\boldsymbol{M}_r = \boldsymbol{M}_b = 2\boldsymbol{I}$, $\boldsymbol{m}_r = [0,0]^T$, and $\boldsymbol{m}_b = [1,1]^T$—implying dual lattices equal to $\pi\mathbb{Z}^2$, with nonzero $\{\boldsymbol{\lambda}_r,\boldsymbol{\lambda}_b\}$ given by $[-\pi,0]^T$, $[0,-\pi]^T$, and $[-\pi,-\pi]^T$.

Examples of several existing CFAs $\boldsymbol{c}(\boldsymbol{n})$ and the corresponding spectra $Y(\boldsymbol{\omega})$ of typical sensor data are illustrated in Figure 5.2 and Figure 5.3, respectively; note that aliasing occurs when, for nonzero $\boldsymbol{\lambda}_r$ or $\boldsymbol{\lambda}_b$, the spectral supports of $X_g(\boldsymbol{\omega})$ and $X_\alpha(\boldsymbol{\omega}-\boldsymbol{\lambda}_r)$ or $X_\beta(\boldsymbol{\omega}-\boldsymbol{\lambda}_b)$ overlap.

Despite its widespread use, the spectral periodization about $[-\pi,0]^T$ and $[0,-\pi]^T$ induced by the Bayer CFA severely limits allowable spectral bandwidth for X_g. In fact, all CFAs depicted in Figure 5.2 are suboptimal in at least one of two ways: First, as shown in Figure 5.3a to Figure 5.3d and Figure 5.3g to Figure 5.3j, spectral copies of the difference channels appear along the horizontal and/or vertical axes of the Fourier representation, leaving the baseband channel X_g vulnerable to the horizontal and vertical features that frequently dominate natural images [17]. Second, as shown in Figure 5.3d to Figure 5.3f and Figure 5.3h to Figure 5.3j, maximal separation between $X_g(\boldsymbol{\omega})$ and $X_\alpha(\boldsymbol{\omega}-\boldsymbol{\lambda}_r), X_\beta(\boldsymbol{\omega}-\boldsymbol{\lambda}_b)$ is precluded unless all nonzero carrier frequencies $\{\boldsymbol{\lambda}_r,\boldsymbol{\lambda}_b\}$ lie elsewhere along the perimeter of $[-\pi,\pi)\times[-\pi,\pi)$.

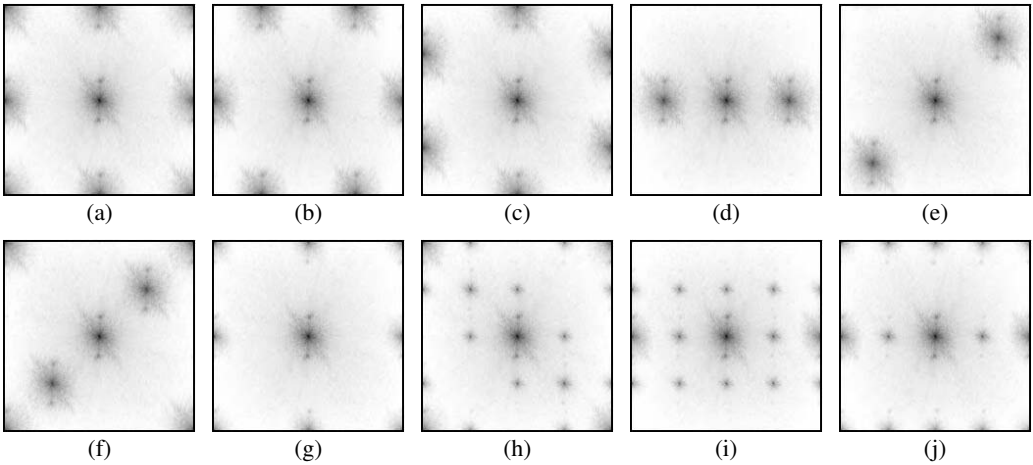


FIGURE 5.3 (See color insert.)

Log-magnitude spectra of a typical color image (i.e., image *flower* here) sampled with CFAs corresponding to Figure 5.2. Color coding is used to distinguish different components, with the $x_g(\boldsymbol{n})$ component shown in green, $x_\alpha(\boldsymbol{n}) = x_r(\boldsymbol{n}) - x_g(\boldsymbol{n})$ in red, and $x_\beta(\boldsymbol{n}) = x_b(\boldsymbol{n}) - x_g(\boldsymbol{n})$ in blue. Individual spectra correspond to: (a) Bayer [4], (b) Yamanaka [5], (c) Lukac [7], (d) vertical stripes [7], (e) diagonal stripes [7], (f) modified Bayer [7], (g) cyan-magenta-yellow, (h) Kodak I [16], (i) Kodak II [16], (j) Kodak III [16].

In fact, these two conditions can be used to formulate a precise statement of CFA suboptimality [15]: any CFA design of the form $\boldsymbol{c}(\boldsymbol{n}) \in \{[1,0,0]^T, [0,1,0]^T, [0,0,1]^T\}$ that places all spectral replicates on the perimeter of $[-\pi,\pi) \times [-\pi,\pi)$, while avoiding $[-\pi,0]^T$ and $[0,-\pi]^T$, can only support *two* distinct colors. While we show in Section 5.3 how panchromatic designs can overcome this restriction, those that have emerged to date (including four-color CFAs) fail to satisfy the above two conditions.

5.2.2 Aliased Sensor Data and Demosaicking

Because the suboptimal CFA designs detailed above are prone to aliasing, linear reconstruction methods no longer suffice as the spectral support of $X_g(\boldsymbol{\omega})$ increases. Reconstruction is then an ill-posed problem, meaning that stronger assumptions about the signal are needed to recover the full-color image from aliased sensor data. To this end, the most common approach is to invoke the principle that *local* image features are sparse in some canonical representation. One explicit form of this principle is *directionality*—the notion that image features are assumed to be oriented in one direction, and thus that the energy of the corresponding local Fourier coefficients is concentrated accordingly. If X_g is sparse in the direction parallel to an image feature orientation, then aliasing can in turn be avoided; this principle is exploited either explicitly or implicitly by many state-of-the-art demosaicking methods [9], [10], [11], [12], [13]. In a similar manner, under a transformation that is local in both space and frequency, the signal energy may be assumed to be compressed into a few transform coefficients; regularization in the transform domain then helps to recover the full color image [8], [12]. However, demosaicking methods that exploit these assumptions are usually highly nonlinear and computationally demanding. Indeed, effec-

tive detection of image feature orientation (especially under the influence of noise) is an active area of research, and the determination of local image statistics requires additional computation. Moreover, subsequent interpolation steps are tightly coupled to estimates of feature directionality; this type of nonlinearity is effectively a data-driven switching mechanism that is expensive to implement in ASIC or DSP hardware. On the other hand, wavelet- and filterbank-based methods often employ iterative reconstruction schemes that may not easily be implemented in portable imaging devices.

The difficulties posed by nonlinear reconstruction methods are especially evident in today's digital video camera architectures. In order to meet the required frame rate with limited computational complexity, for example, it is common to implement demosaicking using methods such as bilinear interpolation that fail to yield satisfactory results. Other processing schemes may introduce pixel flickering artifacts, for instance, interframe oscillation or toggling of pixel colors caused by the susceptibility of edge-detection techniques to noise. Finally, nonlinear demosaicking methods are themselves subject to perturbations due to noise. Although simultaneous image denoising and interpolation methods have emerged in recent years (see, for example, Reference [12]), the difficulties of characterizing noise statistics after nonlinear demosaicking often render stand-alone image denoising methods ineffective. In contrast, the statistics of noise that undergoes only linear processing remain highly tractable, suggesting that a combination of denoising and demosaicking may indeed be possible.

5.3 Spatio-Spectral Color Filter Array Design

By simultaneously considering both the spectral support of luminance and chrominance components, and the spatial sampling requirements of the image acquisition process, we may conceive of a new paradigm for designing CFAs. With robustness to aliasing achieved via ensuring that spectral replicates lie along the perimeter of the Fourier-domain region $[-\pi,\pi)\times[-\pi,\pi)$ while avoiding the values $[-\pi,0]^T$ and $[0,-\pi]^T$ along the horizontal and vertical axes, our CFA design methodology aims to preserve the integrity of color images by way of subsampled sensor data. Images acquired in this manner are easily manipulated, enjoy simple reconstruction schemes, and admit favorable computation-quality trade-offs with the potential to ease subsequent processing in the imaging pipeline [14], [15].

5.3.1 Frequency-Domain Specification of Color Filter Array Designs

Let $0 \leq c_r(\boldsymbol{n}), c_g(\boldsymbol{n}), c_b(\boldsymbol{n}) \leq 1$ indicate the CFA projection values at a particular spatial location, where $c_r(\boldsymbol{n}), c_g(\boldsymbol{n}), c_b(\boldsymbol{n})$ now assume continuous values and hence represent a *mixture* of prototype channels. With the additional constraint that $c_r + c_g + c_b = \gamma$, it follows in analogy to Equation 5.1 that

$$y(\boldsymbol{n}) = \boldsymbol{c}(\boldsymbol{n})^T \boldsymbol{x}(\boldsymbol{n}) = \begin{bmatrix} c_r(\boldsymbol{n}) & \gamma & c_b(\boldsymbol{n}) \end{bmatrix} \begin{bmatrix} x_\alpha(\boldsymbol{n}) \\ x_g(\boldsymbol{n}) \\ x_\beta(\boldsymbol{n}) \end{bmatrix},$$

and we may determine the modulation frequencies of difference channels $x_\alpha(\boldsymbol{n})$ and $x_\beta(\boldsymbol{n})$ by our choice of $c_r(\boldsymbol{n})$ and $c_b(\boldsymbol{n})$. Recalling Equation 5.3, we seek choices such that Fourier transforms of the frequency-modulated difference images $X_\alpha(\boldsymbol{\omega}-\boldsymbol{\lambda}_r), X_\beta(\boldsymbol{\omega}-\boldsymbol{\lambda}_b)$ are maximally separated from the baseband spectrum $X_g(\boldsymbol{\omega})$.

In the steps outlined below, we first specify candidate carrier frequencies $\{\boldsymbol{\tau}_i\}$ and corresponding weights $s_i, t_i \in \mathbb{C}$ for color filters $c_r(\boldsymbol{n})$ and $c_b(\boldsymbol{n})$. Recalling that for constants ν, κ we have that $\mathscr{F}\{\kappa c_r + \nu\}(\boldsymbol{\omega}) = \kappa \mathscr{F} c_r(\boldsymbol{\omega}) + \nu\delta(\boldsymbol{\omega})$, we see that it is possible to manipulate our candidate color filter values until the realizability condition $0 \le c_r(\boldsymbol{n}), c_g(\boldsymbol{n}), c_b(\boldsymbol{n}) \le 1$ is met. This notion leads to the following algorithm for frequency-domain specification of color filter array designs (with $\bar{\cdot}$ denoting complex conjugation, and Figure 5.4 illustrating the algorithmic steps):

ALGORITHM 5.1 Frequency-domain color filter array design.

1. Specify initial values $\{\boldsymbol{\tau}_i, s_i, t_i\}$. Set modulation frequencies:
$$c_r^{(0)} = \mathscr{F}^{-1} \sum_i s_i \delta(\boldsymbol{\omega}+\boldsymbol{\tau}_i) + \bar{s}_i \delta(\boldsymbol{\omega}-\boldsymbol{\tau}_i)$$
$$c_b^{(0)} = \mathscr{F}^{-1} \sum_i t_i \delta(\boldsymbol{\omega}+\boldsymbol{\tau}_i) + \bar{t}_i \delta(\boldsymbol{\omega}-\boldsymbol{\tau}_i).$$
2. Subtract a constant $\nu_r = \min c_r^{(0)}(\boldsymbol{n}), \nu_b = \min c_r^{(0)}(\boldsymbol{n})$ (non-negativity):
$$c_r^{(1)} = c_r^{(0)} - \nu_r, \qquad c_b^{(1)} = c_b^{(0)} - \nu_b.$$
3. Scale by $\kappa = (\max_{\boldsymbol{n}} c_r^{(1)}(\boldsymbol{n}) + c_b^{(1)}(\boldsymbol{n}))^{-1}$ (convex combination):
$$c_r^{(2)} = \kappa c_r^{(1)}, \qquad c_b^{(2)} = \kappa c_b^{(1)}.$$
4. Find green: $c_g^{(2)} = 1 - c_r^{(2)} - c_b^{(2)}$.
5. Scale by $\gamma = (\max\{c_r^{(2)}(\boldsymbol{n}), c_g^{(2)}(\boldsymbol{n}), c_b^{(2)}(\boldsymbol{n})\})^{-1}$:
$$c_r = \gamma c_r^{(2)}, \qquad c_g = \gamma c_g^{(2)}, \qquad c_b = \gamma c_b^{(2)}.$$

In the first step, candidate carrier frequencies are determined by taking the inverse Fourier transform of $\delta(\boldsymbol{\omega} \pm \boldsymbol{\tau}_i)$. The conjugate symmetry in this step guarantees a real-valued color filter array; in general, however, the resultant design is not physically realizable (points in Figure 5.4a fall outside of the first quadrant, for example). Constants ν_r, ν_b are then subtracted to ensure non-negativity of color filters (Figure 5.4b). A scaling by κ and computation of the green component in the next two steps projects candidate values onto the unit simplex, ensuring convexity and a maximum component value of unity (Figure 5.4c and Figure 5.4d). Finally, multiplication by γ maximizes the quantum efficiency of the color filters (Figure 5.4e). The resultant CFA is physically realizable, with observed spectral data Y given by the sum of baseband components and modulated versions of X_α and X_β:

$$\begin{aligned} Y(\boldsymbol{\omega}) = {} & \gamma X_g(\boldsymbol{\omega}) - \gamma\kappa\nu_r X_\alpha(\boldsymbol{\omega}) - \gamma\kappa\nu_b X_\beta(\boldsymbol{\omega}) \\ & + \gamma\kappa \sum_i \{s_i X_\alpha + t_i X_\beta\}(\boldsymbol{\omega}+\boldsymbol{\tau}_i) + \{\bar{s}_i X_\alpha + \bar{t}_i X_\beta\}(\boldsymbol{\omega}-\boldsymbol{\tau}_i). \end{aligned}$$

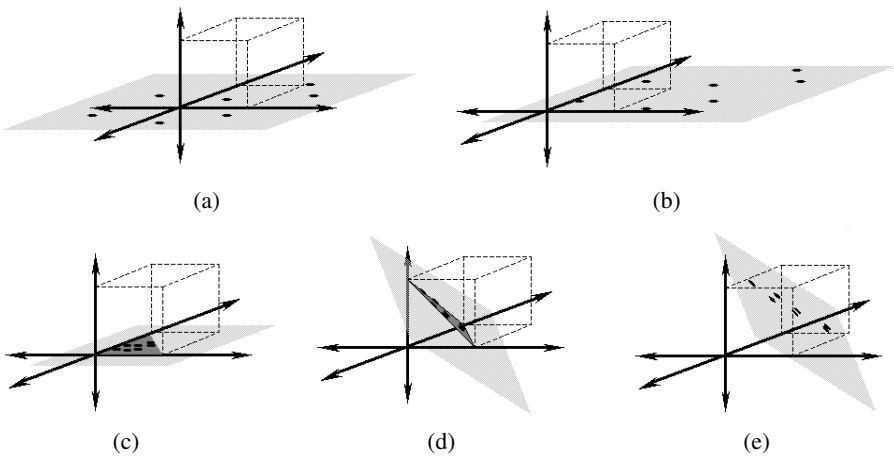


FIGURE 5.4

Color filter array design visualized in Cartesian coordinates (c_r, c_b, c_g), with the dotted cube representing the space of physically realizable color filters ($0 \leq c_r(\boldsymbol{n}), c_g(\boldsymbol{n}), c_b(\boldsymbol{n}) \leq 1$). Steps 1 to 5 in Algorithm 5.1 are shown as (a) to (e), respectively.

This approach enables the specification of CFA design parameters directly in the Fourier domain, by way of carrier frequencies $\{\boldsymbol{\tau}_i\}$ and weights $\{s_i, t_i\}$. In doing so, we ensure that nonzero carrier frequencies lie along the perimeter of $[-\pi, \pi) \times [-\pi, \pi)$, while avoiding the values $[-\pi, 0]^T$ and $[0, -\pi]^T$ as desired.

5.3.2 Analysis and Design Trade-Offs

In this section, some notable features of the above CFA design strategy are considered; readers are referred to Reference [15] for a thorough analysis of design trade-offs. We first note that CFA designs resulting from Algorithm 5.1 are panchromatic, with the resultant filters comprising a mixture of red, green, and blue colors at each spatial location. As color filters are commonly realized by pigment layers of cyan, magenta, and yellow dyes over an array of pixel sensors (i.e., subtractive colors) [18], designs for which $\gamma > 1$ suggest improved quantum efficiency. Furthermore, it becomes easier to control for sensor saturation, as the relative quantum efficiency at each pixel location is approximately uniform ($c_r + c_g + c_b = \gamma$). We also note that the space of feasible initialization parameters $\{\boldsymbol{\tau}_i, s_i, t_i\}$ corresponding to Algorithm 5.1 is underconstrained, offering flexibility in optimizing the CFA design according to other desirable characteristics such as demosaicking complexity, pattern periodicity, resilience to illuminant spectrum, and numerical stability [15].

Our design strategy assumes bandlimitedness of the difference images x_α and x_β, and therefore its robustness hinges on how well this claim holds in various practical situations (e.g., under changes in illuminant). Even as the bandwidths of the modulated difference spectra grow, the increased distance between these channels and the baseband component serves to reduce the risk of aliasing, effectively increasing the spatial resolution of the imaging sensor. Consequently, local interpolation methods are less sensitive to the directionality of image features, and a linear demosaicking method then suffices for many applications.

As described earlier, linearization of the demosaicking step is attractive for several reasons: it can be coded more efficiently in DSP chips, it eliminates the temporal toggling pixel problems in video sequences, it provides a more favorable setup for deblurring, and it yields more tractable noise and distortion characterizations.

5.4 Linear Demosaicking via Demodulation

In this section, we show that the processing pipeline of a typical digital camera can be exploited to greatly reduce the complexity of reconstruction methods [14]. Suppose the conjugate modulation sequences $c_\alpha(\boldsymbol{n}) = c_r^{(0)}(\boldsymbol{n})^{-1}$ and $c_\beta(\boldsymbol{n}) = c_b^{(0)}(\boldsymbol{n})^{-1}$ exist;[3] when these sequences are orthogonal, the modulated signal can be recovered via a multiplication by the conjugate carrier frequency followed by a lowpass filter. Assuming mutual exclusivity of the supports of X_g, X_α, and X_β in the frequency domain, we expect an exact reconstruction according to

$$\hat{\boldsymbol{x}}(\boldsymbol{n}) = \begin{bmatrix} \hat{x}_r(\boldsymbol{n}) \\ \hat{x}_g(\boldsymbol{n}) \\ \hat{x}_b(\boldsymbol{n}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1/(\gamma\kappa) & 0 & 0 \\ \nu_r/\gamma & 1/\gamma & \nu_b/\gamma \\ 0 & 0 & 1/(\gamma\kappa) \end{bmatrix} \begin{bmatrix} h_\alpha * \{c_\alpha y\} \\ h_g * y \\ h_\beta * \{c_\beta y\} \end{bmatrix}, \tag{5.4}$$

where $*$ denotes the discrete convolution operator, and the passbands of lowpass filters h_α, h_g, h_β match the respective bandwidths of the signals x_α, x_g, x_β.

Given the mutual exclusivity of the signals x_α, x_g, x_β in the Fourier domain, we assume $c_r^{(0)} h_\alpha + h_g + c_b^{(0)} h_\beta = \delta$, where $\delta(\boldsymbol{n})$ is again a Kronecker delta function. Using the linearity and modulation properties of convolution, we obtain:

$$\begin{aligned} h_g * y &= (\delta - c_r^{(0)} h_\alpha - c_b^{(0)} h_\beta) * y \\ &= y - \{c_r^{(0)} h_\alpha\} * y - \{c_b^{(0)} h_\beta\} * y \\ &= y - c_r^{(0)}\{h_\alpha * \{c_\alpha y\}\} - c_b^{(0)}\{h_\beta * \{c_\beta y\}\}. \end{aligned}$$

The demodulation in Equation 5.4 in turn takes the following simplified form:

$$\begin{aligned} \hat{\boldsymbol{x}}(\boldsymbol{n}) &= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1/(\gamma\kappa) & 0 & 0 \\ \nu_r/\gamma & 1/\gamma & \nu_b/\gamma \\ 0 & 0 & 1/(\gamma\kappa) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -c_r^{(0)}(\boldsymbol{n}) & 1 & -c_b^{(0)}(\boldsymbol{n}) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h_\alpha * \{c_\alpha y\} \\ y \\ h_\beta * \{c_\beta y\} \end{bmatrix} \\ &= \begin{bmatrix} 1/(\gamma\kappa) + \nu_r/\gamma - c_r^{(0)}(\boldsymbol{n})/\gamma & 1/\gamma & \nu_b/\gamma - c_b^{(0)}(\boldsymbol{n})/\gamma \\ \nu_r/\gamma - c_r^{(0)}(\boldsymbol{n})/\gamma & 1/\gamma & \nu_b/\gamma - c_b^{(0)}(\boldsymbol{n})/\gamma \\ \nu_r/\gamma - c_r^{(0)}(\boldsymbol{n})/\gamma & 1/\gamma & 1/\gamma + \nu_b/\gamma - c_b^{(0)}(\boldsymbol{n})/\gamma \end{bmatrix} \begin{bmatrix} h_\alpha * \{c_\alpha y\} \\ y \\ h_\beta * \{c_\beta y\} \end{bmatrix}. \end{aligned} \tag{5.5}$$

The first term in Equation 5.5 is a 3×3 matrix multiplication (a completely pixelwise operation), whereas the spatial processing component is contained in its second term. In

[3] In this chapter, we do not discuss cases in which there are zeros; however, the results presented here generalize easily to such cases via an appropriate multiplicative constant.

the usual layout of a digital camera architecture, a color conversion module follows immediately, converting the tristimulus output from demosaicking to a standard color space representation through another 3×3 matrix multiplication on a per-pixel basis. The two cascading matrix multiplication steps can therefore be performed together in tandem, with the combined matrix computed offline and preloaded into the camera system.

Given sufficient separation of the modulated signals in the frequency domain, crudely designed low-pass filters suffice for the reconstruction task. Suppose we choose to implement Equation 5.5 using a separable two-dimensional odd-length triangle filter — a linear-phase filter with a modest cutoff in the frequency domain. Four cascading boxcar filters can be used to implement a filter of length $2q-1$ having the following Z transform, with Z_1 and Z_2 corresponding to delay lines in horizontal and vertical directions, respectively:

$$H_\alpha(\mathbf{Z}) = H_\beta(\mathbf{Z}) = \left(\frac{1-Z_1^{-q}}{1-Z_1^{-1}}\right)\left(\frac{1-Z_1^{-q}}{1-Z_1^{-1}}\right)\left(\frac{1-Z_2^{-q}}{1-Z_2^{-1}}\right)\left(\frac{1-Z_2^{-q}}{1-Z_2^{-1}}\right). \tag{5.6}$$

The computational complexity of the above system is eight adders for h_α and h_β each. Moreover, in 4×4 repeating CFAs, the carrier frequencies $c_r^{(0)}$ and $c_b^{(0)}$ are often proportional to sequences of ± 1's (and by extension, c_α and c_β also). In this case, the multiplication by -1 before addition in Equation 5.6 simply replaces adders with subtracters, which is trivial to implement. The overall per-pixel complexity of the demodulation demosaicking in Equation 5.5 is therefore comparable to that of bilinear interpolation (16 add/subtract operations per full pixel), despite its state-of-the-art image quality performance.

5.5 Examples and Analysis

In this section we provide several examples of CFA designs and analyze their performance. These designs, shown in Figure 5.5 and detailed in Table 5.1, were generated in the spirit of Algorithm 5.1 by employing an exhaustive search over a restricted parameter space $\{\tau_i, s_i, t_i\}$ [15]. Though some CFAs in Figure 5.5 have rectangular geometries, we see that nevertheless every pixel sensor has an equal number of neighboring colors, a condition that helps mitigate cross-talk noise due to leakages of photons and electrons. Their

TABLE 5.1
Example CFA patterns specified in terms of parameter values $\{\tau_i, s_i, t_i\}$.

pattern		$i=0$	$i=1$	pattern		$i=0$	$i=1$	pattern		$i=0$	$i=1$
	τ_i	$(\pi, \frac{\pi}{2})$	(π,π)		τ_i	$(\pi, \frac{2\pi}{3})$	$(\frac{2\pi}{3}, \pi)$		τ_i	$(\pi, \frac{\pi}{2})$	(π,π)
A	red s_i	$1+1j$	1	C	red s_i	$1j$	$1j$	E	red s_i	$1+1j$	1
	blue t_i	$1+1j$	-1		blue t_i	$1j$	$-1j$		blue t_i	$1+1j$	-1
	τ_i	$(\pi, \frac{\pi}{2})$	(π,π)		τ_i	$(\pi, \frac{\pi}{3})$	(π,π)		τ_i	$(\pi, \frac{\pi}{2})$	(π,π)
B	red s_i	$1+1j$	0	D	red s_i	$3+4j$	1	F	red s_i	$1+1j$	0
	blue t_i	0	1		blue t_i	$3-4j$	1		blue t_i	0	1

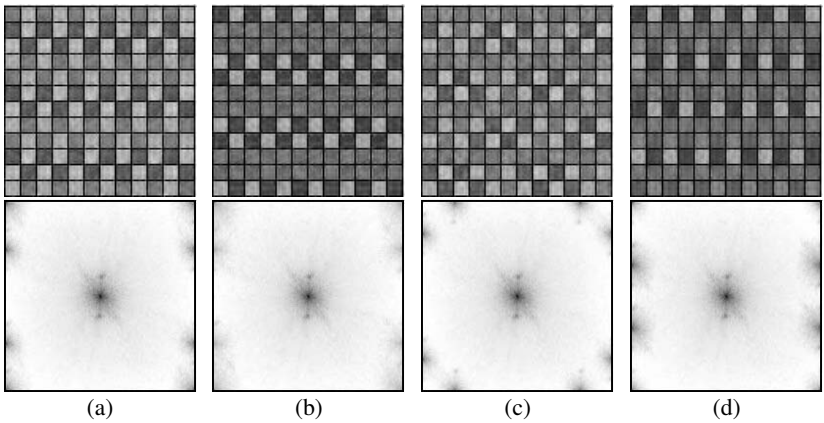


FIGURE 5.5 (See color insert.)

Proposed CFAs (top) and resultant log-magnitude spectra (bottom) of a typical color image (i.e., image *flower* here). Color coding is used as in Figure 5.3 to distinguish components X_α, X_g, and X_β. Subfigures correspond to: (a) *pattern A*, (b) *pattern B*, (c) *pattern C*, and (d) *pattern D*.

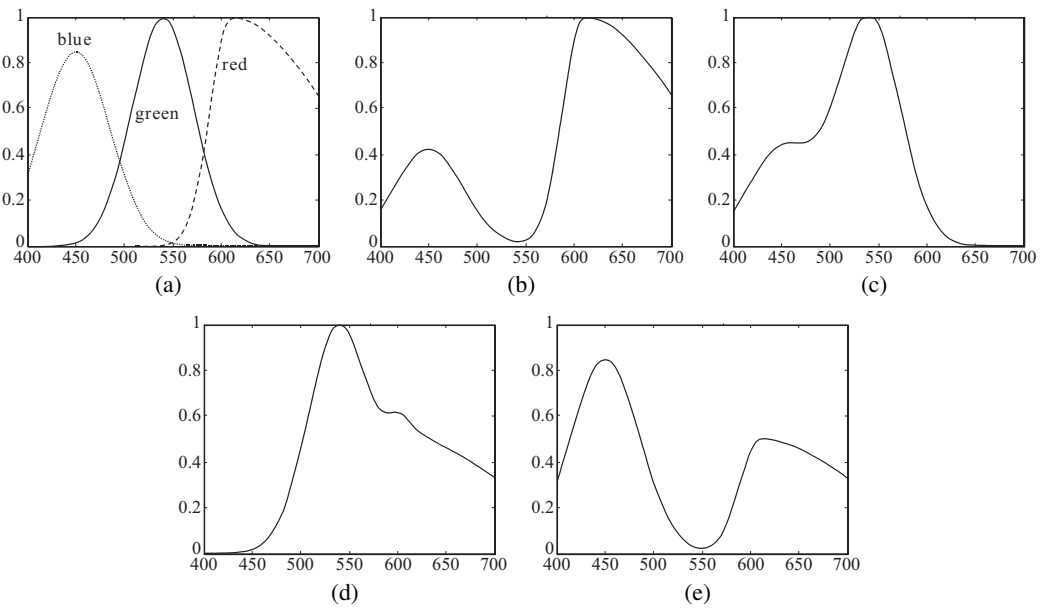


FIGURE 5.6

Spectral sensitivity characteristics (a) of a typical Sony CCD sensor [19], and (b-e) the corresponding *pattern A* color filters derived from these characteristics.

designs are given in Table 5.1 as combinations of prototype red, green, and blue filters; the precise color specifications used in subsequent demosaicking experiments were derived from a popular Sony CCD quantum efficiency function [19] shown in Figure 5.6a. The resultant spectral responses, shown in Figure 5.6b, may be implemented using subtractive color pigments such as cyan, magenta, and yellow.

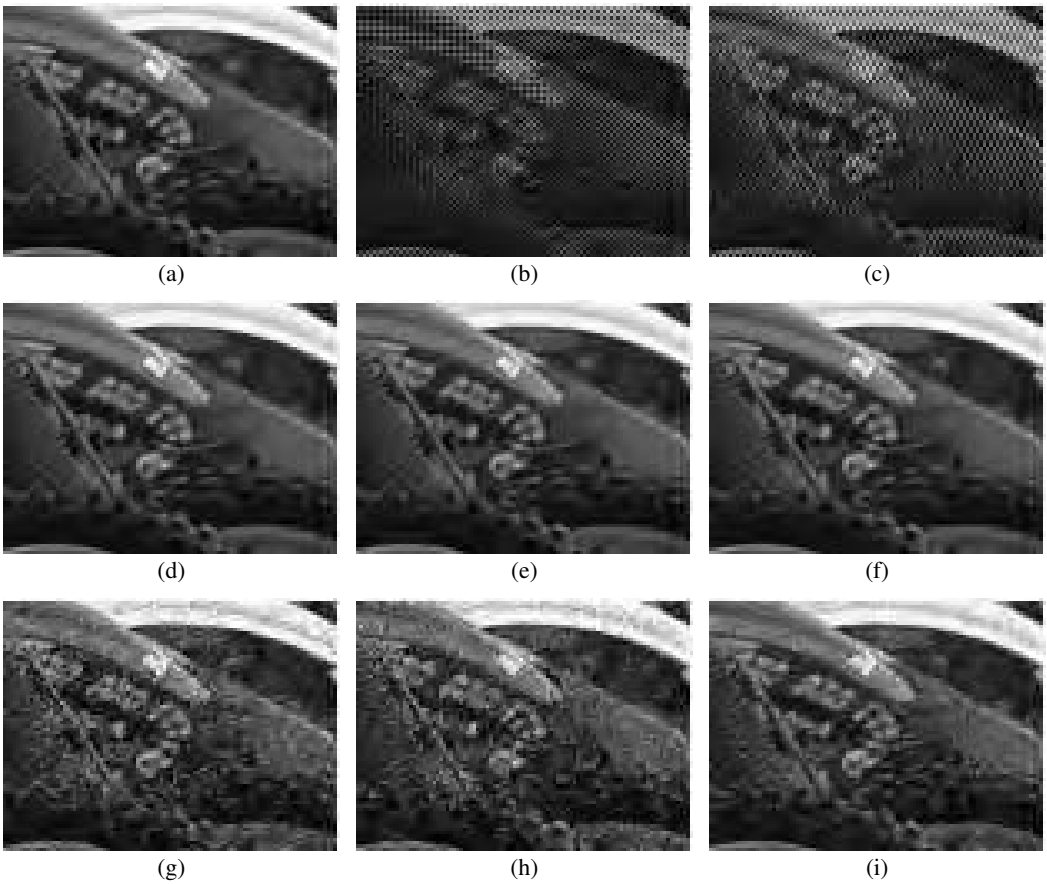


FIGURE 5.7 (See color insert.)

Bike image sensor data (top row), with nonlinear and linear reconstruction methods shown for the case of clean (middle row) and noisy (bottom row) sensor data. Individual images correspond to: (a) original image, (b) Bayer CFA sampling, (c) *pattern A* sampling, (d) nonlinear Bayer reconstruction [8], (e) linear Bayer reconstruction, (f) linear *pattern A* reconstruction, (g) noisy nonlinear Bayer reconstruction, (h) noisy linear Bayer reconstruction, and (i) noisy *pattern A* linear reconstruction.

In comparing Figure 5.3 and Figure 5.5, we see that in the latter case spectral copies of x_α and x_β are placed farther from the Cartesian axes and the origin, thus achieving a better separation of channels in the Fourier domain. The implications of this design improvement may be seen in the demosaicking examples of Figure 5.7; while demosaicking performance is both algorithm- and CFA-dependent, we may consider state-of-the-art methods for demosaicking Bayer CFA data along with the linear reconstruction methodology outlined in Section 5.4, using the well-known *bike* test image shown in Figure 5.7a.

To this end, Figure 5.7b and Figure 5.7c show simulated sensor data $y(\boldsymbol{n}) = \boldsymbol{c}(\boldsymbol{n})^T \boldsymbol{x}(\boldsymbol{n})$ for the *bike* image $\boldsymbol{x}(\boldsymbol{n})$, acquired under $\boldsymbol{c}(\boldsymbol{n})$ representing the Bayer CFA and *pattern A* of Figure 5.5, respectively. Figure 5.7d to Figure 5.7f show demosaicked images corresponding respectively to a reconstruction of a color image from Bayer CFA data using the iterative, nonlinear method of Reference [8], the linear demosaicking algorithm of Sec-

tion 5.4, and the same linear method applied to the *pattern A* sampled data. This latter reconstruction is competitive with the nonlinear Bayer reconstruction of Figure 5.7d, and exhibits significantly reduced zipper artifacts. On the other hand, compared to the purely linear Bayer demosaicking shown in Figure 5.7e, the linear *pattern A* reconstruction shows a significant gain in fidelity for equal hardware resolution and computational cost. Finally, Figure 5.7g to Figure 5.7i demonstrate its improved resilience to noise, by way of showing the same three reconstructions applied to sensor data corrupted by simulated Poisson noise. Compared to the reconstructions using Bayer CFA data depicted in Figure 5.7g and Figure 5.7h, the *pattern A* linear reconstruction of Figure 5.7i renders contributions from signal-dependent noise far less noticeable.

5.6 Conclusion

By considering the interplay between color filter arrays and typical images, we have posed here the CFA design problem as one of simultaneously maximizing the spectral support of luminance and chrominance channels subject to their mutual exclusivity in the Fourier domain. From this perspective, current design practices were seen to be suboptimal: as image resolution increases, existing CFAs are prone to aliasing, linear reconstruction methods no longer suffice, stronger assumptions must be made about the underlying signal, and additional computational resources are needed to reconstruct the full-color image.

Key to our design paradigm was the notion that the measurement process, an inner product between the color filter array and the image data, induces a modulation in the frequency domain. To this end, we chose to modulate the chrominance spectra away from the baseband luminance channel, and in doing so we proposed a constructive method to design a physically realizable CFA by specifying these modulation frequencies directly. This method generates panchromatic CFA designs that mitigate aliasing and admit favorable computation-quality trade-offs. As we have shown, our corresponding linear demosaicking method yields state-of-the-art performance with an order of complexity comparable to that of bilinear interpolation.

References

[1] K. Parulski and K.E. Spaulding, *Digital Color Imaging Handbook*, ch. Color image processing for digital cameras, G. Sharma (ed.), Boca Raton, FL: CRC Press, 2002, pp. 728–757.

[2] R. Ramanath, W.E. Snyder, Y. Yoo, and M.S. Drew, "Color image processing pipeline," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 34–43, January 2005.

[3] R. Lukac and K.N. Plataniotis, *Color Image Processing: Methods and Applications*, ch. Single-sensor camera image processing, R. Lukac and K.N. Plataniotis (eds.), Boca Raton, FL: CRC Press / Taylor & Francis, October 2006, pp. 363–392.

[4] B.E. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 1976.

[5] S. Yamanaka, "Solid state color camera," U.S. Patent 4 054 906, August 1977.

[6] M. Parmar and S.J. Reeves, "A perceptually based design methodology for color filter arrays," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Canada, May 2004, vol. III, pp. 473–476.

[7] R. Lukac and K.N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1260–1267, November 2005.

[8] B.K. Gunturk, J. Glotzbach, Y. Altunbasak, R.W. Schafer, and R.M. Mersereau, "Demosaicking: Color filter array interpolation in single chip digital cameras," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 44–54, January 2005.

[9] K. Hirakawa and T.W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 360–369, March 2005.

[10] D. Alleysson, S. Süsstrunk, and J. Hérault, "Linear demosaicing inspired by the human visual system," *IEEE Transactions on Image Processing*, vol. 14, no. 4, pp. 439–449, April 2005.

[11] R. Lukac and K.N. Plataniotis, "Universal demosaicking for imaging pipelines with an RGB color filter array," *Pattern Recognition*, vol. 38, no. 11, pp. 2208–2212, November 2005.

[12] K. Hirakawa and T.W. Parks, "Joint demosaicking and denoising," *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2146–2157, August 2006.

[13] E. Dubois, "Filter design for adaptive frequency-domain Bayer demosaicking," in *Proceedings of the IEEE International Conference on Image Processing*, Atlanta, GA, USA, October 2006, pp. 2705–2708.

[14] K. Hirakawa and P.J. Wolfe, "Second-generation CFA and demosaicking design," in *Proceedings of the IS&T/SPIE 19th Annual Symposium on Electronic Imaging*, San Jose, CA, USA, January 2008.

[15] K. Hirakawa and P.J. Wolfe, "Spatio-spectral color filter array design for enhanced image fidelity," in *Proceedings of the IEEE International Conference on Image Processing*, San Antonio, TX, USA, September 2007, vol. 2, pp. 81–84. Extended version submitted to *IEEE Transactions on Image Processing*, October 2007.

[16] T. Kijima, H. Nakamura, J. Compton, and J. Hamilton, "Image sensor with improved light sensitivity," U.S. Patent Application 2007 0 177 236, August 2007.

[17] D.M. Coppola, H.R. Purves, A.N. McCoy, and D. Purves, "The distribution of oriented contours in the real world," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 95, no. 7, pp. 4002–4006, March 1998.

[18] R. Ramanath and W.E. Snyder, "Adaptive demosaicking," *Journal of Electronic Imaging*, vol. 12, no. 4, pp. 633–642, October 2003.

[19] Sony Corporation, "Diagonal 6 mm (type 1/3) progressive scan CCD image sensor with square pixel for color cameras," http://products.sel.sony.com/semi/PDF/ICX204AK.pdf, 2004.

6

Mosaicking and Demosaicking in the Design of Multispectral Digital Cameras

Lidan Miao, Hairong Qi, and Wesley E. Snyder

6.1 Introduction

In recent years, considerable work has been conducted in multispectral imaging, which expands color cameras' capability to capture spectral information at multiple wavelengths other than visible light. Multispectral images have been widely investigated for their applications in remote sensing [1], [2] to analyze the landscapes and structures from aircrafts or satellites. Particularly, differences in spectral signatures among various land covers enable the detection and classification of different crops or minerals [3], [4]. Multispectral imaging has also been widely used in the field of biological microscopy in an effort to discriminate multiple co-localized fluorescent molecules [5], [6], [7]. Using common mi-

croscopy methods, the number of molecules that can be detected simultaneously is limited by both spectral and spatial overlap. These issues can be tackled using spectral information which extends the possibilities to distinguish multiple proteins, organelles or functions within a single cell [8]. In biomedical imaging, one of the potential applications of multispectral imagery is the detection of breast cancer at its early stage when cancer cells are still very small in size but show aggressive growth activities which can be picked up by infrared imaging [9], [10]. Moreover, multispectral imaging is a significant technology for the acquisition, analysis, and display of accurate color information [11], [12], [13].

Many methods have been used to obtain multispectral imagery [14]. To achieve high spectral resolution, the following techniques are popularly adopted in existing imaging systems: 1) imaging spectrometer, which uses an optical dispersing element such as a grating or prism to split the light into many narrow and adjacent wavelength bands, and the energy in each band will be measured by a separate detector. Such imaging systems are complex and expensive. The manufacture of multiple sensor array is also complicated and delicate. In addition, the data size is limited by the requirements of data storage, transmission, and processing [15]; 2) filter-based spectral imaging, in which images are taken with a static camera mounted with a set of discrete filters. The filters are switched by revolving a filter wheel, then a discrete set of multispectral images can be obtained. However, due to the changing environment, the acquired images need to be registered before other processing algorithms can be applied [14]; and 3) the technique of quantum well imaging arrays, which still needs years of research before maturity [16].

To achieve an efficient solution for multispectral imaging, we study the application of multispectral filter array (MSFA), a mosaic array of multiple wavelength-specific filters, which is stimulated by the color filter array (CFA) technique in commercial digital color cameras [17]. Although considerable work has been conducted in the color domain, to the best of our knowledge, no attempt has been given to multispectral imaging. To acquire multispectral information, instead of using multiple detectors at each pixel location to obtain measurements for different spectral bands, single photo detector covered by an MSFA is adopted. In this way, a multispectral camera captures a scene such that each photo detector only captures spectral information at a single band, resulting in a mosaic-like monochrome image called the *mosaicked* image. To obtain the full multispectral data, a reconstruction operation, referred to as *MSFA demosaicking*, is required to estimate the missing spectral components at each pixel location. We call the resulted multispectral image the *reconstructed* image or the *demosaicked* image. The system diagram of an MSFA digital camera

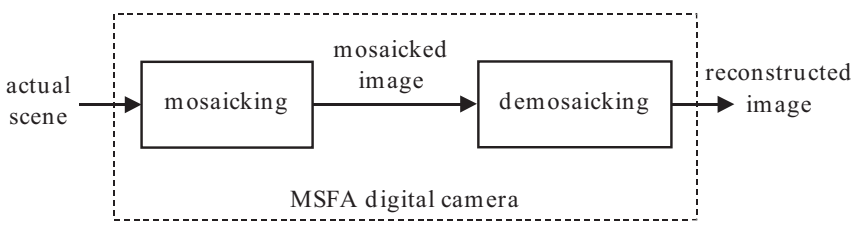


FIGURE 6.1

System diagram of an MSFA camera.

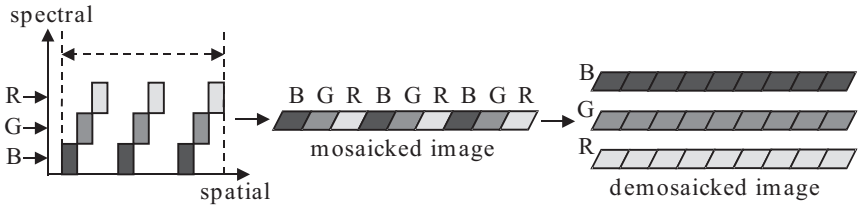


FIGURE 6.2

Illustration of mosaicking and demosaicking process. The light red, green, and blue pixels in the right figure represent the estimated values.

is shown in Figure 6.1. Figure 6.2 illustrates the mosaicking and demosaicking process for a row of a color image. It is clear that at each spatial location, only the information at a single band is measured, and for individual spectral bands, there exist a large number of missing components across the image plane. Compared to a full multispectral imaging system, an MSFA camera trades spatial resolution for spectral resolution. The MSFA technique provides several advantages like low cost, exact registration, compact physical setup and strong robustness, which have made it very attractive to the industry.

This chapter focuses on the MSFA design methodologies (*MSFA mosaicking*) and the development of effective reconstruction algorithms (*MSFA demosaicking*). In Section 6.2, we discuss the underlying design principles starting from a brief review of color filter array and its biological relevances. Three design criteria for MSFA are then identified and summarized which form the essential building block of this chapter. The gist of this chapter is the MSFA design approach and the corresponding demosaicking algorithm which are detailed in Section 6.3 and Section 6.4, respectively. In Section 6.3, we present a binary tree based method to generate MSFAs for both rectangular and hexagonal tessellations. Given the number of spectral bands and the probability of appearance (POA) of each band, the algorithm starts from a checkerboard pattern and generates various MSFAs following a binary tree separation procedure. The demosaicking algorithm addressed in Section 6.4 follows the same tree in a reverse direction and progressively estimates the missing pixel components. Three interrelated processes are involved in the reconstruction algorithm, namely band selection, pixel selection, and interpolation, which facilitate the exploration of spectral correlation to achieve better reconstructions than individual demosaicking of each image plane. In Section 6.5, the performance of a multispectral mosaicking and demosaicking system is evaluated from two perspectives. We first evaluate the intrinsic properties of MSFA patterns to see how well the created patterns satisfy the design criteria. We then assess the entire system by evaluating the performance of reconstructed images in terms of classification accuracy and root mean square error compared with the full multispectral data. Finally, Section 6.6 concludes this chapter.

6.2 Mosaicked Filter Array Patterns and Their Design Philosophy

In essence, a multispectral camera is simply a *visual* system, which can sense spectral information outside the visual wavelength range as many animal visual systems do. One

important feature of the human and animal visual systems is that they possess the capability of instant processing and high resolution of discrimination. Many recent efforts have been devoted to the emulation of human and animal visual systems to achieve cost-effective, high-resolution and real-time imaging systems. The technique of CFA is one of the major achievements following this principle. Due to the unique advantages of the mosaic technique, the potential application of MSFAs has been studied in References [18] and [19].

Several problems have to be addressed before the MSFA technique becomes a reality. First, there is a tradeoff between the spatial and spectral resolution as shown in Figure 6.2. To achieve high spectral resolution, less number of samples within each band can be acquired, resulting in low spatial resolution images; on the other hand, for a given imaging field, many more detectors have to be integrated on chip to obtain higher spatial resolution, which would increase the camera cost and complexity. There is yet another concern we need to address besides the resolution issue. Since the design of MSFA is associated with a set of selected spectral bands and different targets would possess different sets of *signature* spectral bands, the bands that are most effective in discriminating the target from its background, it appears that a multispectral camera can only be specialized in imaging a certain type of target which is apparently not cost-effective. Fortunately, ongoing research in the area of adaptive imaging [20], [21], [22] has proposed potential solutions to this problem. The Defense Advanced Research Projects Agency (DARPA) has recently developed an Adaptive Focal Plane Array (AFPA) program [23], in which a high-performance focal plane array (FPA) is to be developed that is widely tunable on a pixel-by-pixel basis across the relevant wavebands in the infrared spectrum. With this technology, real-time reconfiguration of the array can be realized to meet different application requirements in an ever changing environment.

Two fundamental procedures involved in the MSFA technique are the design of MSFAs and the demosaicking algorithm. In the color domain, considerable work has been reported to optimally reconstruct the full color image for the popular Bayer array [24]. However, studies on the intrinsic properties of the filter array as well as the underlying design principles have been very limited [25], [26]. Moreover, the design philosophies of CFAs are only applicable in the color domain. Its generalization to MSFA in the multispectral domain is not straightforward and needs further research. Due to the increased number of spectral bands, the design of MSFAs as well as the reconstruction of multispectral data is more complicated. Therefore, the development of a generic algorithm with capability of generalizing the mosaicking and demosaicking processes of different multispectral applications is of great importance. In this section, we summarize our study findings in the CFA technique and its biological relevance, from which we identify three design requirements for MSFA that are the guidelines for the generic MSFA generation algorithm discussed in the next section.

6.2.1 Color Filter Arrays

Most digital cameras use a rectangular array of light-sensing elements covered by wavelength-specific filters to capture spectral information at different bands. By doing so, only one color component is sensed at each pixel location. The resulting mosaic-like image is then processed using spectral interpolation algorithms to estimate the missing color

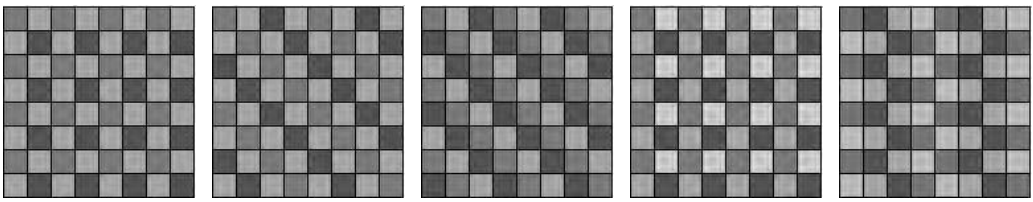


FIGURE 6.3

Examples of CFAs: (a) Bayer array [27], (b) diagonal Bayer array [28], (c) diagonal stripe [28], (d) Sony RGBE array [29], and (e) spatio-spectral array [26]. Figure 5.2 shows presented CFAs in color.

components. The key idea of using the filter array technique has been demonstrated in Figure 6.2. Although some expensive cameras use a set of three light-sensing elements or place three layers of sensors one over the other to produce a high quality image, a single array of sensors with a color filter array is the simplest solution, and its disadvantages, i.e., reduced spatial resolution, will be overcome as it becomes possible to make larger arrays at higher densities. For a color filter array, the type of filters and the spatial arrangement of different filters constitute its two basic features.

Regarding the type of filters used, different color bases have been considered [25], [28], including the tristimulus color basis (RGB, YMC), the mixed primary/complementary color (MGCY), and various four-color schemes [29]. Due to the complexity issue in the demosaicking process and the widely used RGB image format for storage, most existing color systems utilize the RGB CFAs [25]. In terms of the spatial arrangement of color filters, the earliest and most popularly used CFA is the Bayer array [27] (shown in Figure 6.3a). However, the design philosophy was not followed up and extended until several improvements proposed in References [30], [31], and [32]. Recently, there is a growing research interest in investigating the interplay between the CFA design and the subsequent demosaicking process. The studies in this area have shown that the CFA has a great impact on the quality of reconstructed images besides the demosaicking algorithms [25], [26]. A spatio-spectral method is proposed in Reference [26] aiming at improving color filter arrays to achieve enhanced image quality. In Figure 6.3, we illustrate several examples of CFA patterns; for more discussions on the design of color filter arrays, see Reference [25].

6.2.2 Biological Relevance

The idea of using mosaic filter array instead of multiple photo detectors is stimulated from the study findings of the human visual system (HVS) as well as many other animal visual systems. In the human retina, three types of cones with absorbance maxima in the long-, middle-, and short-wavelength (L, M, and S cones) region are organized into mosaics that tile the retina [33] as illustrated in Figure 6.4a. Only a single type of photoreceptor samples the image scene at any given location. Several studies [33], [34] have attempted to analyze the spatial arrangement of the L, M, and S cones. It was suggested in Reference [35] that the three types of cones are arranged randomly in the human eyes. The random arrangement gives rise to clumps containing only single type of cones, where the eyes cannot distinguish colors. In addition, the random sampling causes a deterioration

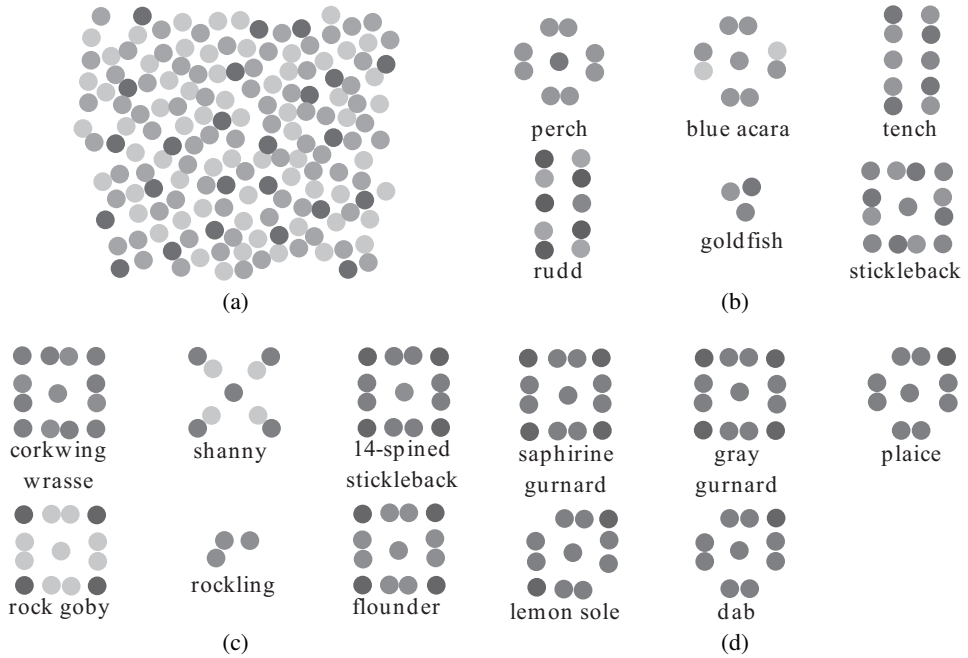


FIGURE 6.4 (See color insert.)

Cone mosaic of human and fish retina (pictures taken from Reference [37]): (a) human cone mosaic, (b) freshwater fish, (c) litoral coastal fish, and (d) deep coastal fish.

of image quality [36], in which the authors showed that an irregular retinal mosaic array causes a frequency-dependent reduction in signal amplitude and introduces random noise. They further pointed out the retina irregularity reduces visual acuity especially for high frequency signals.

The cone mosaic has also been examined in a variety of species in the animal society. Figure 6.4b to Figure 6.4d illustrate the cone mosaic of fish in various aquatic environments. In the fresh water, where many wavelengths of light still penetrate the water, fish have more photoreceptors and a more heterogeneous arrangement across species. As one moves to deeper water, where the available wavelengths of light are more limited, there is less variation in both the types of photopigments and the mosaic arrangements across species. Those fish living in deep coastal waters where the available spectra are very limited show the least variation and the fewest photopigments [37]. In addition, researchers have discovered UV (300-360nm)-absorbing cones in the Japanese dace fish, which enable the fish to see wavelengths down to 360nm [38]. It has also been found that the mosaic array of most vertebrates is regular. Those animals who need high acuity and rely heavily on vision possess a very regular mosaic array, such as fish [39], [40] and mouse [41], [42].

6.2.3 Design Requirements for Multispectral Filter Arrays

Several critical issues in the design of CFA related to the effort of camera manufacture have been summarized in References [25] and [28]: matching the sensitivity of HVS, enabling cost-effective reconstruction algorithms, immunity to color artifacts, tolerance to

sensor imperfections, and immunity to optical crosstalk among neighboring pixels. These criteria are associated with the two intrinsic features of CFAs, i.e., the selection of filter type and the spatial arrangement of different filters. Inspired by the CFA design characteristics and its biological relevance, we identify three important design requirements for MSFAs: probability of appearance, spectral consistency, and spatial uniformity.

Probability of appearance guides the selection of filters. One criterion used in the design of CFA (e.g., Bayer array) is that the pattern has to match the sensitivity response of the HVS. Since the HVS is more sensitive to changes in the green spectral band, most CFAs have more pixels sensitive to the green than to the red or the blue. In the design of MSFA, the objective is mostly to achieve better target recognition and separation of the object from clutter. We thus relate the POA, which is the ratio of the number of samples of a certain type of filter to the total number of samples, to the effectiveness of the spectral band in recognizing the target. The spectral band that affects the classification result the most will be assigned more pixels in the filter array. Generally, for a specific application scenario, the target(s) of interest is/are known *a priori*. An efficient multispectral camera would select a proper subset of spectral bands that maximizes the class separability [43]. A number of band selection algorithms [44], [45] have been proposed. In addition to selecting an optimal subset of bands out of the original set according to a class-separability criterion, some algorithms [46], [47] rank the selected spectral bands on the basis of eigenvectors and eigenvalues. Knowing the importance of each spectral band, its probability of appearance can then be derived accordingly.

Spectral consistency concerns the spatial locations of different filters to reduce the optical crosstalk, a very common phenomenon existed in optical imaging systems [48]. As illustrated in Figure 6.5, an incoming photon intersects with the blue filter at a certain angle and enters the adjacent photodetector under the green filter instead of the blue one. This results in a contamination of the adjacent pixel's charge packet and generates artifacts in the output image. Since the effect of optical crosstalk cannot be corrected using any image processing methods, a sub-optimal design consideration is to arrange the filter array mosaic pattern in a way such that the crosstalk would be uniformly distributed across the entire imaging plane, since a consistent effect of contamination would cause less damage than an

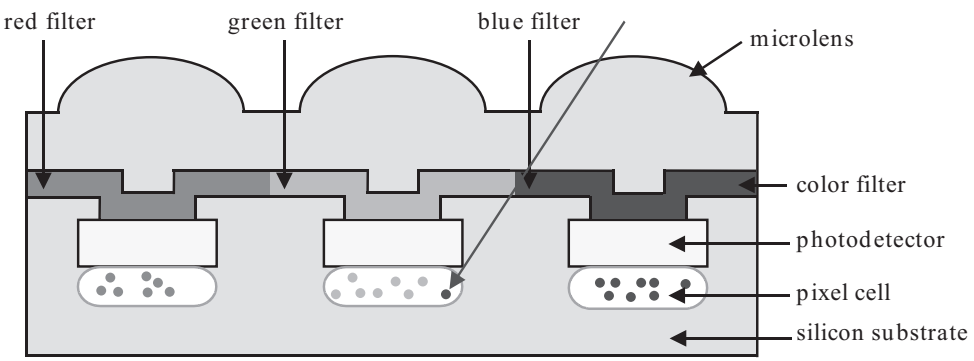


FIGURE 6.5

Illustration of optical crosstalk. Redrawn from Reference [48]. © 2006 IEEE

inconsistent artifact which interferes with the object recognition. In order to achieve this design requirement, pixels of a certain spectral band should always have the same pattern of neighbors, a property which we refer to as the *spectral consistency*.

Spatial uniformity also concerns the spatial arrangement of different filters. In a mosaicked pattern, since each pixel only has one direct measurement from a certain spectral band, the unmeasured spectral components of the pixel must be estimated from its neighbors. This requires that the filter array for each spectral band samples the entire image as evenly as possible. If the pixels distribute densely in some regions while sparsely in other regions, serious information loss might occur. The research in biological studies also supports that the uniform distribution outweighs the random arrangement.

We consider the above three criteria as the most important issues in the design of MSFAs. Although there might exist some other concerns, the extra constraints introduced could result in empty set of solutions. As in the color domain, no single CFA can satisfy all the design criteria listed previously [25].

6.3 A Generic Filter Array Design Method

Since normally, two-dimensional signals are digitized and stored as rectangularly sampled arrays, in this chapter, we only discuss the MSFA generation using rectangular arrays. For in-depth discussions using the hexagonal tessellation, readers are referred to Reference [18].

Suppose we have selected a set of representative spectral bands and derived their probabilities of appearance, this section reviews the generic MSFA design method [18] with a focus on the spatial arrangement of various filters. The design algorithm starts from a checkerboard pattern and generates different MSFAs following a binary tree separation procedure. The binary tree-driven MSFA design process guarantees that the pixel distributions of different spectral bands are uniform and highly correlated. We will show, through case studies, that most of the CFAs currently used by the industry can be derived as special cases of MSFAs generated using the generic algorithm.

We adopt the checkerboard pattern as the starting point to generate different filter arrays. The selection of the checkerboard pattern is based on a number of properties that this pattern possesses: first, the checkerboard pattern is symmetric horizontally, vertically, and diagonally; second, the black and white blocks are uniformly distributed across the whole board; and third, this pattern has the same sampling frequency in both the horizontal and vertical directions. These properties facilitate the generation of MSFA patterns that satisfy all the design requirements.

Suppose we need to generate a K-band filter array and each spectral band has its specific POA $r_1, \cdots, r_K$, where $r_i = \frac{1}{2^n}$ with n being an integer, and $\sum_{i=1}^{K} r_i = 1$. First, we generate a binary tree such that it has K leaves and the leaf i represents a spectral band with a POA of r_i. Following this binary tree, we treat the original checkerboard as the root and use a combination of *decomposition* and *subsampling* operations to generate various patterns. Each resulting pattern should correspond to one node in the binary tree. Finally, all the leaf

patterns are combined to form a mosaic pattern, which is the desired MSFA satisfying the three design requirements.

Figure 6.6 illustrates the creation of a five-band MSFA using a binary tree with five leaves (Figure 6.6a), which is generated based on the specified probabilities $r = \{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\}$. Following this tree, various patterns are generated through the operation of decomposition and subsampling as shown in Figure 6.6b. The decomposition is applied to the nodes at the even levels of the binary tree (including level zero, i.e., the root). The function of decomposition is to treat the pattern as a checkerboard and then divide the black and white blocks into two patterns. For example, the label **1** and label **2** patterns are generated by decomposing the original checkerboard, and the label **7** and label **8** patterns are the decomposed results of the pattern **3**. The subsampling is to downsample the pattern by $2^{\frac{level+1}{2}}$ along the horizontal and vertical directions, where *level* refers to the level of the pattern being processed. It can be seen that the label **3** and label **4** patterns are obtained by subsampling pattern **1**, and the label **5** and label **6** patterns are the results of subsampling pattern **2** by 2.

Process the checkerboard until it has the same structure as the binary tree. The next step is to combine all the leaves to generate a mosaic pattern, as shown in Figure 6.6c, in which the left figure is obtained by combining all the leaf patterns in Figure 6.6b, and the right figure is the color representation. It can be shown that two of the popularly used CFAs illustrated in Figure 6.3 are actually special cases generated from the generic algorithm. For example, if we combine patterns **2**, **3**, **4** and assign different colors, the resulting mosaic pattern is the same as the Bayer array, and the Sony RGBE array can be obtained by combining patterns **3**, **4**, **5**, and **6**.

One unavoidable constraint associated with the binary tree-based method is that the POA is limited to power of two. In the case that the probabilities do not fit the tree, we choose the closest approximation to substitute the original POAs. This approximation is necessary to satisfy the *uniform distribution* design requirement which dictates that each pixel always has 2^n amount of neighbors. Note that the rectangular domain is similarly constrained; each pixel is either in a four-neighborhood or an eight-neighborhood.

From the above case studies, it is easy to see that the filter arrays generated from the generic method have the following characteristics: first, each spectral band is arranged symmetrically and uniformly; second, each band has the same number of neighbors of a certain spectral band but the relative positions of different bands are not always the same; and third, the probability of appearance of each spectral band is determined by the two separation steps. The MSFA generation process described above can be mathematically formulated as a sampling problem of multispectral images [18].

6.4 A Generic Binary Tree-based Demosaicking Method

Although there has been considerable research in the field of demosaicking algorithm [49], [50], [51], [52], [53], [54], [55], they are confined to the three-band Bayer array and cannot be directly extended to multispectral demosaicking. Due to the increased

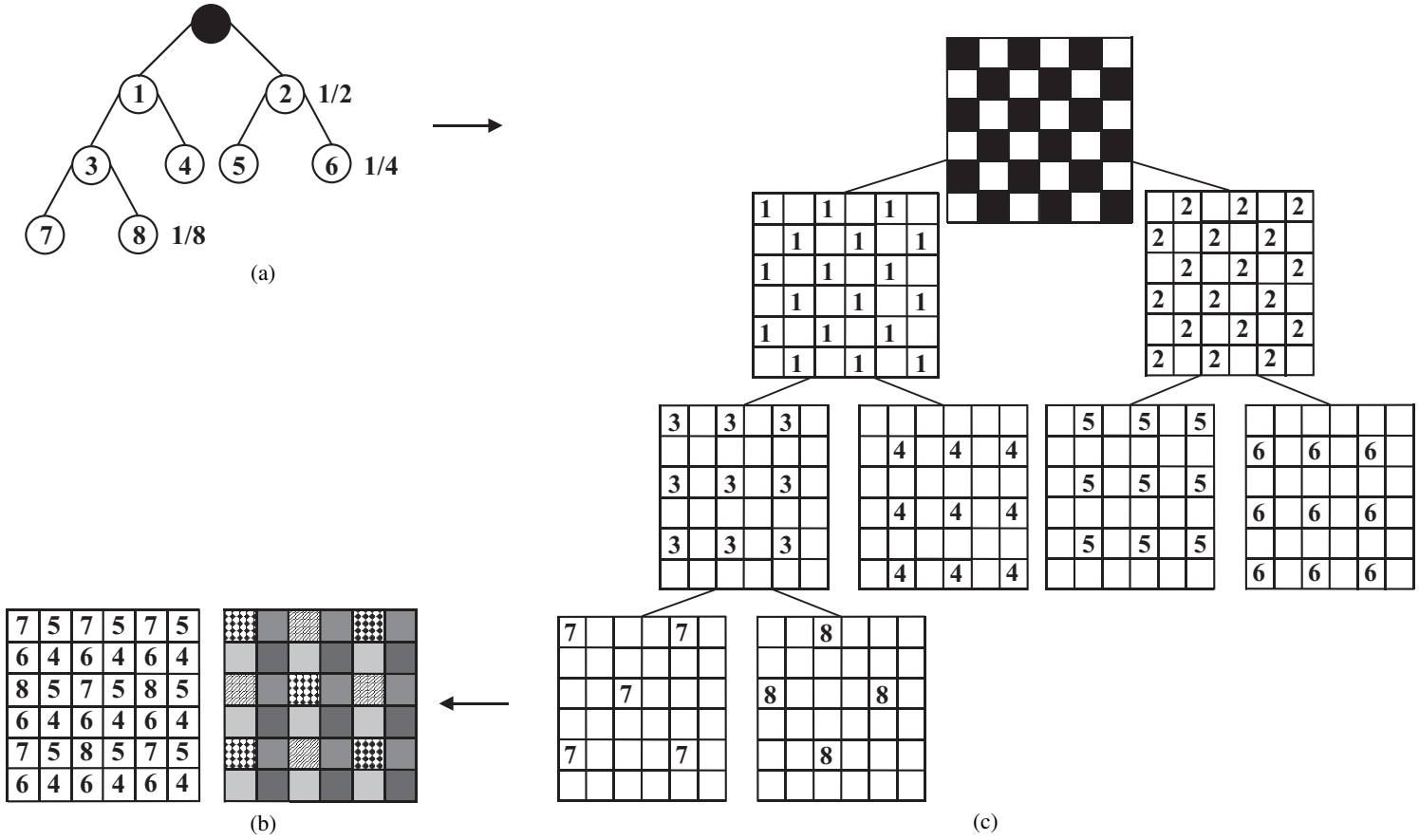


FIGURE 6.6

Generic MSFA generation process: (a) binary tree, (b) checkerboard separation, and (c) five-band MSFA generated by combining all the leaf patterns from (b). © 2006 IEEE

number of spectral bands in the multispectral domain, the resolution of known MSFA samples of each band would be gradually reduced. On one hand, this resolution reduction inevitably introduces severe artifacts which is not desired; however, on the other hand, the reduced spatial resolution brings extra spectral information. The correlation among different spectral bands has the potential of providing more information than the independent demosaicking of individual image planes. As in the color domain, the spectral correlations have been intensively utilized in the CFA demosaicking algorithms to render better reconstructions [51], [56], [57], [58]. In the following, we first discuss the possible spectral correlations that can help the demosaicking process. Then, a demosaicking algorithm involving three interrelated components that facilitate the exploration of spectral correlation will be presented.

6.4.1 Correlation Analysis of Multispectral Images

One commonly used concept of spectral correlations in the CFA demosaicking is the color ratio [59] or the color difference rule [51], which states that within a local image region, the ratios or differences between different color channels are very similar. Instead of estimating the absolute value in the two chromatic color channels (i.e., red and blue), these algorithms estimate the color ratio or difference in order to derive the chrominance value. Since the human visual system is more sensitive to color artifacts than to luminance or saturation errors [60], these schemes can reconstruct full color images with less visible artifacts and sharp edges. Although very promising in the color domain, these rules, however, do not hold in the multispectral domain as we have analyzed in Reference [19].

Another important inter-band correlation in the color domain is that all color bands possess similar edge information [61], [62]. Most wavelet-based demosaicking algorithms explore this correlation [63], [64]. In the multispectral domain, due to the wide wavelength range with each band capturing very different signatures, the edge information of different spectral bands would not be the same. Although it is true that different spectral bands might identify different edge locations, there should be no spurious edges. In other words, if the edges derived in all spectral bands are combined together, the resulting image would present all edge information of the scene. One example is elaborated in Figure 6.7, where we sum up seven edge images (they have intensity **1** at edge locations and **0** everywhere else) generated from a 7-band multispectral image using the Canny edge detector, and different colors are used to denote different intensity values. Note that for the worst case, if all the images possess different edge locations, then the summation image would have thick edges and all edge pixels have intensity one. However, we can see from Figure 6.7 that only a few pixels possess intensity one and most edges still have single-pixel width, resulting from similar edge locations among different spectral bands.

The consistency of edge locations among different spectral bands enables better reconstructions of high frequency information. The essential idea is that we identify a band with rich high frequency details and then use the edge information of this band to help the reconstruction of the other image planes. For this purpose, we developed a generic demosaicking algorithm based on the same tree that generates the MSFAs. The algorithm progressively estimates the missing pixel values, while utilizing the edge correlation information. Three interrelated issues need to be addressed: *band selection* — the determination

FIGURE 6.7 (See color insert.)

Summation of edge images of seven spectral bands. Different colors represent different intensity values (1: red, 2: green, 3: blue, 4: cyan, 5: magenta, 6: yellow, 7: white). © 2006 IEEE

of the interpolation order of different spectral bands; *pixel selection* — the determination of pixel interpolation order within each spectral plane; and *interpolation* — the interpolation algorithm to estimate missing pixels within each spectral band. The following discussion will focus on the rectangular tessellation. The same idea can be extended to the hexagonal domain.

6.4.2 Band Selection

In the multispectral domain, since normally there are more than three spectral bands that need to be processed, the order of spectral band selection for interpolation needs to be predetermined. As illustrated in Section 6.3, different spectral bands possess different POAs. It is intuitive that more detailed information will be preserved in the spectral bands with higher POAs and that these bands contribute more in obtaining a reconstructed image that better resembles the real scene. Moveover, the reconstructed image plane can be utilized to assist the interpolation of other spectral bands with lower POAs based on the spectral correlation of consistent edge locations. For this reason, we start the interpolation by choosing a spectral band with the highest POA.

In the binary tree, band selection can be viewed as a process of selecting leaf nodes at different tree levels. We know the nodes at the same level possess the same POA and the deeper the level, the smaller the POA. To select spectral bands with their POAs in a descending order, we start from the first level of the binary tree. If there is a leaf node at this level, it will be the first selected spectral band for interpolation. This process continues as the tree level goes deeper. If there exists more than one leaf at a certain level, the selection order among these nodes is random. This band selection scheme facilitates our exploitation of spectral correlation. Since the band which preserves the edge information the best will be interpolated first, the estimation of other bands can utilize the edge information of the first interpolated image plane provided that different bands possess similar edge locations.

6.4.3 Pixel Selection

In most demosaicking schemes in the color domain, the missing pixels are estimated only based on known pixel values. However, in the multispectral domain, more missing

pixels are present in each spectral band and only using known MSFA samples will not generate good results. Here, we present a "*progressive*" demosaicking method, taking into consideration that sparse samples exist in MSFA patterns. That is, part of the missing pixel values are estimated first, then the estimated pixel values together with the known MSFA samples are used to estimate other unknown pixel values. In this way, it is very important to determine which pixel locations are estimated first and which are the next.

To effectively utilize the structural features of different patterns presented in the binary tree, we develop a pixel selection scheme, which is a binary tree traversal process. Starting from one of the leaf patterns selected in the band selection component, the algorithm first interpolates the missing band information at pixel locations where its sibling pattern locates, then the algorithm goes up one level of the binary tree and finds the sibling of its parent pattern. If its parent's sibling is an internal node, then the leaf patterns of the subtree under this sibling pattern are investigated. This process continues until the root node is visited. It can be seen that, at each step, after interpolating the selected pixel locations, the resulting pattern is the same as the parent pattern. Thus, the pixel selection scheme guarantees that all the intermediate patterns during the demosaicking process are those present in the binary tree.

Figure 6.8 illustrates an example of the pixel selection process, in which we aim to reconstruct spectral band **7**. Starting from the node **7**, we first select the pixel locations where its sibling pattern **8** locates (Figure 6.8b). We use **7/8** to denote the interpolation of the **7** value at the **8** location. Then we go up one level to node **3** and select pixel locations where its sibling pattern **4** locates (Figure 6.8c). Continuing this process one more level will lead us to the internal node **2**, which is the combination of pixel locations of the pattern **5** and **6** (Figure 6.8d). The directed dash lines in Figure 6.8a indicate the trace of traversal and the resulting pattern at each step is shown in Figure 6.8b to Figure 6.8d, respectively. Note that the intermediate patterns are determined by the traversal trace on the binary tree, and the POA at each step is given by $1/2^{level}, 1/2^{level-1}, \cdots, 1/2$.

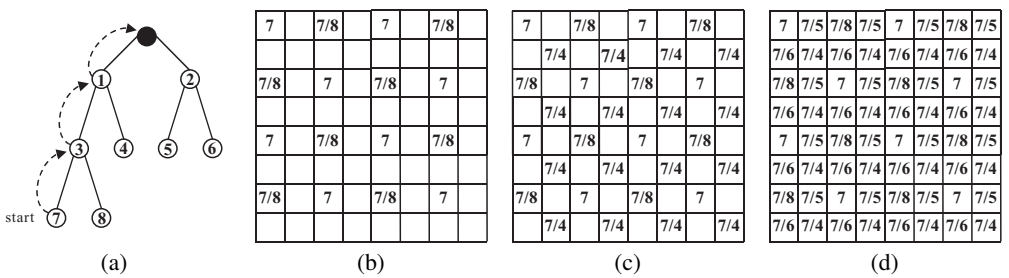


FIGURE 6.8

Illustration of pixel selection process of band **7**. (a) The directed dash lines indicate the trace of traversal. (b) The **7** values at pixel locations with known **8** are first estimated based on known **7**s. (c) The **7** values at pixel locations with known **4** are secondly estimated based on both known and estimated **7**s from (b). (d) At node **1**, pixel locations at **2** positions are selected, which are combinations of pixel locations at node **5** and **6**.

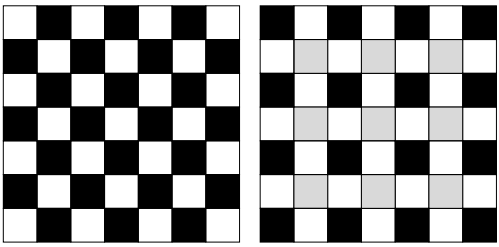


FIGURE 6.9

The basic patterns: (a) quincunx pattern, and (b) rectangular pattern.

6.4.4 Interpolation

Given a certain pixel location within a certain spectral band, selected based on the band and pixel selection scheme described above, the last issue to be investigated is how to estimate the missing pixel values based on neighboring pixel information. The key to the design of a generic demosaicking algorithm is the application of the binary tree. We observe that the set of pixels selected based on the binary tree always form one of the two regular distribution patterns, i.e., the quincunx or the rectangular, as shown in Figure 6.9. It can also be seen that through subsampling, all the patterns in the binary tree can be transformed to these two *basic patterns*. Therefore, the demosaicking of MSFAs eventually relies on the interpolation of these two basic patterns.

In order to preserve edge details, we adopt the idea of edge-sensing interpolation (i.e., weighted sum of neighboring pixels with weights determined based on the edge information), which has been successfully used in CFA demosaicking [51], [59]. Let B denote the spectral band being processed and $B_{i,j}$ the known pixel value at the spatial location (i,j), $\hat{B}_{i,j}$ the corresponding estimate, the missing components of the quincunx pattern shown in Figure 6.9a are estimated using the weighted sum of four nearest neighbors,

$$\hat{B}_{i,j} = \frac{\sum_{s,t} W_{i+s,j+t} B_{i+s,j+t}}{\sum_{s,t} W_{i+s,j+t}} \tag{6.1}$$

with $|s+t| = 1, \forall s,t \in \{-1,0,1\}$. The weights of the two neighboring pixels along the vertical direction are calculated by

$$\begin{aligned} W_{m,n} = (&1 + |B_{m+2,n} - B_{m,n}| + |B_{m-2,n} - B_{m,n}| \\ &+ \frac{1}{2}|B_{m-1,n-1} - B_{m+1,n-1}| + \frac{1}{2}|B_{m-1,n+1} - B_{m+1,n+1}|)^{-1} \end{aligned} \tag{6.2}$$

and that along the horizontal direction is

$$\begin{aligned} W_{m,n} = (&1 + |B_{m,n+2} - B_{m,n}| + |B_{m,n-2} - B_{m,n}| \\ &+ \frac{1}{2}|B_{m+1,n-1} - B_{m+1,n+1}| + \frac{1}{2}|B_{m-1,n-1} - B_{m-1,n+1}|)^{-1} \end{aligned} \tag{6.3}$$

It can be seen that the weight $W_{m,n}$ is inversely proportional to the edge magnitude at location (m,n). By doing so, the unknown pixel is interpolated along the edge direction.

For the rectangular pattern shown in Figure 6.9b, we only need to estimate the set of the shaded pixels as the resulting pattern is again a quincunx distribution. Using the same idea,

the unknown shaded pixel values are estimated by the weighted sum of the four diagonal neighbors. The weights of the left-diagonal are calculated by

$$W_{m,n} = (1 + |B_{m+2,n+2} - B_{m,n}| + |B_{m-2,n-2} - B_{m,n}| + \frac{1}{2}|B_{m,n-2} - B_{m+2,n}| + \frac{1}{2}|B_{m-2,n} - B_{m,n+2}|)^{-1} \tag{6.4}$$

Similarly, the right-diagonal weights are

$$W_{m,n} = (1 + |B_{m+2,n-2} - B_{m,n}| + |B_{m-2,n+2} - B_{m,n}| + \frac{1}{2}|B_{m-2,n} - B_{m,n-2}| + \frac{1}{2}|B_{m,n+2} - B_{m+2,n}|)^{-1} \tag{6.5}$$

This edge-sensing approach interpolates the unknown according to pixel weights derived from edge information. Thus, the estimation of edge information directly affects the quality of reconstructed images. In multispectral imaging, as the number of spectral bands increases, the spatial resolution decreases in certain spectral bands and the edge information based on the low resolution spectral band would not be reliable. As analyzed before, the edge information in different spectral bands is either similar or partly overlapped. The spectral band with the highest POA preserves the edge information the best. Therefore, the edge information in high resolution spectral band can be used to calculate the weights for low resolution bands since the band selection scheme guarantees the high resolution spectral bands are reconstructed first. We refer to the proposed method as the binary tree based edge sensing method (BTES).

6.5 Experiments and Results

In the color domain, the CFA has a significant impact on the quality of reconstructed images [25]. Similarly, the characteristics of MSFAs also play an important role in determining the maximum information that can be reconstructed from the mosaicked data. To fairly evaluate the two components involved in the MSFA technique, i.e., mosaicking and demosaicking, we conduct evaluations from two aspects. First, we carry out evaluations of the intrinsic properties of filter arrays by assessing how well the generated MSFAs satisfy the design criteria. Then, experiments are performed to evaluate the demosaicking algorithms using two performance measures: the commonly used root mean square error (RMSE) and the classification accuracy.

6.5.1 Pure Evaluation of Generated Filter Arrays

From the design requirement analysis, we know that the *spatial uniformity* property guarantees that there is equal amount of information across the image plane such that the missing spectral information can be estimated with the same degrees of fidelity. On the other hand, the *spectral consistency* can counteract the artifacts caused by the optical crosstalk. To assess the intrinsic properties of the filter arrays, we design two performance metrics

to measure the *spatial uniformity* and the *spectral consistency*, referred to as the *static coefficient* (SC) and the *consistency coefficient* (CC), respectively.

6.5.1.1 Static Coefficient

In each filter plane, the pixels with known measurements are called the *active pixels* and all the others are the *dead pixels*. To assess the spatial uniformity of individual band, we only concern the active pixels, which means we will be processing images with a bunch of "holes". In order to illustrate the effect of spatially non-adjacent pixels on one another in a more rational way, we introduce the electrostatic force model, in which the *active pixels* are interpreted as static electric particles with the same polarity of unit charge. We assume all the active pixels in a filter plane create static force fields around them. The joint force exerted on the particle of interest within a certain neighborhood is used to define the SC metric. The size of the neighborhood should be larger than or equal to the minimum distance between any two active pixels, i.e., the minimum number of active pixels included in a neighborhood must be two. Figure 6.10 illustrates one example of the particle interaction, where the black blocks indicate the active pixels. The center pixel i is the particle of interest, which is surrounded by pixels $1, \cdots, 5$. The vector $\mathbf{F}_{ki}$ represents the force exerted on the pixel i by the neighboring pixel k ($k \in \mathcal{N}_i$, $\mathcal{N}_i$ denotes the set of neighboring pixels of i), whose magnitude and direction are determined by

$$\|\mathbf{F}_{ki}\| = \frac{1}{(x_k - x_i)^2 + (y_k - y_i)^2}, \quad \tan\theta = \frac{y_k - y_i}{x_k - x_i} \tag{6.6}$$

where $\|\cdot\|$ denotes the magnitude of a vector and θ the direction. Note that the magnitude of the force is inversely proportional to the square of the distance between the two pixels and the direction of the force is along the axis that connects the two active pixels, pointing away from the pixel of interest. The total force $\mathbf{F}_i$ exerted on the center pixel i by its neighbors is

$$\mathbf{F}_i = \sum_{k \in \mathcal{N}_i} \mathbf{F}_{ki} \tag{6.7}$$

The SC of an MSFA can then be calculated by

$$SC = 1 - \frac{1}{K} \sum_{j=1}^{K} \frac{1}{1 + \mu_j} \tag{6.8}$$

where μ_j denotes the mean magnitude given by $\mu_j = \frac{1}{N_j} \sum_{i=1}^{N_j} \|\mathbf{F}_i\|$, and N_j is the total number of active pixels in spectral band j, K the number of spectral bands. For a certain spectral plane, the more uniformly the active pixels distribute, the smaller the μ_j, and the smaller the SC. Since SC is normalized to be between 0 and 1 in Equation 6.8, a zero SC would indicate a uniform distribution, whereas a maximum SC (SC = 1) implies the least uniformity.

6.5.1.2 Consistency Coefficient

In the mosaicking technique, we expect all the pixels in a certain spectral band always have the same immediate neighbors so that the contamination introduced by optical

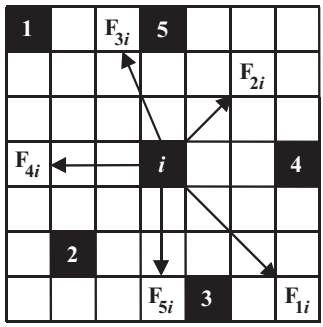


FIGURE 6.10

Illustration of electric force. © 2006 IEEE

crosstalk generates the same effect on all the pixels in this spectral band. To quantify the spectral consistency, we begin by forming the notions of *superpixel* and *template superpixel*. A superpixel is the combination of a center pixel and its immediate neighbors. A template superpixel is a distinguishable superpixel with the center pixel from a specific spectral band. For example, the Bayer CFA shown in Figure 6.11a has four template superpixels, illustrated in Figure 6.11b to Figure 6.11e. The two template superpixels in Figure 6.11b and Figure 6.11c have the same G center pixel, hence, we refer to them as the template superpixels of the G band. Likewise, Figure 6.11d and Figure 6.11e show the template superpixels of the R and the B bands, respectively. For a certain spectral band j, we identify all template superpixels and label them as $T_{j1}, T_{j2}, \cdots, T_{jm}$, where the subscript m denotes the number of different templates in this band. It is known that an optimal design of spectral consistency admits one template superpixel for each spectral band. The more template superpixels a certain band has, the more inconsistent it would be across the image plane, and the worst case is when there are equal number of superpixels matching different templates. Let $n_{j1}, n_{j2}, \cdots, n_{jm}$ denote the number of superpixels matching different templates, and $n_j = n_{j1} + n_{j2} + \cdots + n_{jm}$ is the total number of superpixels in spectral band j. Then the probability of occurrence of template i is $p(T_{ji}) = \frac{n_{ji}}{n_j}$. We define an entropy-like metric as follows:

$$H_j(consistency) = -\sum_{i=1}^{m} p(T_{ji}) \log p(T_{ji}) \tag{6.9}$$

to measure the spectral consistency, referred to as the *consistency entropy*. Note the larger the consistency entropy, the less consistent the pattern.

The consistency entropy is an overall measure which does not indicate to what degree these template superpixels differ from each other. For example, the two template superpixels of the G band in Figure 6.11b and Figure 6.11c have the same number of different neighbors (4 Gs, 2 Rs, and 2 Bs), but their relative positions are different. In other cases, the superpixel might have neighbors of different spectral bands. For the clarity of explanation, we refer to the former as the *relative position difference* (RPD) and the latter the *spectral band difference* (SBD). Intuitively, these two types of differences would cause additional inconsistency aside from that caused by the different numbers of template superpixels, which should be taken into account in the formulation of CC. In addition, the

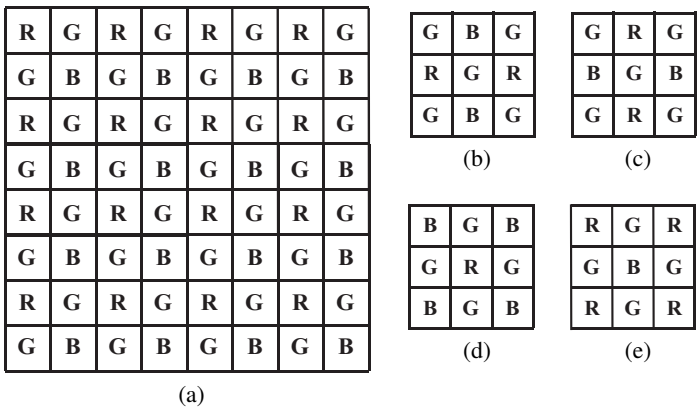


FIGURE 6.11

Illustration of the superpixel: (a) an 8×8 Bayer array, (b, c) two different superpixels of the G spectral band, (d) superpixel of the R spectral band, (e) superpixel of the B spectral band. © 2006 IEEE

inconsistency caused by SBD is more severe than that of RPD. Therefore, we introduce two penalty terms, *pSBD* and *pRPD*, to account for the contamination introduced by SBD and RPD, respectively. Combined with the consistency entropy, the CC of a single band is defined as

$$CC_j = \frac{1}{1 + pRPD_j + pSBD_j} \cdot \frac{1}{1 + H_j(consistency)}$$

and the CC of the entire MSFA is

$$CC = 1 - \frac{1}{K} \sum_{j=1}^{K} CC_j \tag{6.10}$$

It is apparent that a smaller value of CC implies a more consistent pattern and $CC = 0$ indicates the optimal spectral consistency.

The next problem is how to determine the values of *pRPD* and *pSBD*. It is valid to assume that crosstalk only happens between adjacent pixels (here, we use eight-adjacency for rectangular tessellation). Therefore, the window size to analyze *pRPD* and *pSBD* is 3×3. We write the 3×3 template superpixel in a lexicographical form. For example, the template superpixels in Figure 6.11b and Figure 6.11c are $T_{G1} = GRGBGBGRG$ and $T_{G2} = GBGRGRGBG$. With this representation, it is easier to see that finding the difference between template superpixels of a certain spectral band is simply a problem of finding "distance" between strings (or codes). We adopt the Levenshtein distance (or edit distance) [65] to serve this purpose. The Levenshtein distance counts a difference not only when strings have different characters but also when one has a character whereas the other does not. One of the most popular applications of the Levenshtein distance is *spell checking*. It tries to find the most common typing errors, e.g., character omissions, insertions, and substitutions. The idea is to calculate the minimum number of such operations to convert from one string (code) to another. We calculate the Levenshtein distance between different template superpixels. Since the size of the superpixel is always the same in our application, there is no need for *insertion* or *deletion*. Only two operations, *substitution* and *swapping*, are

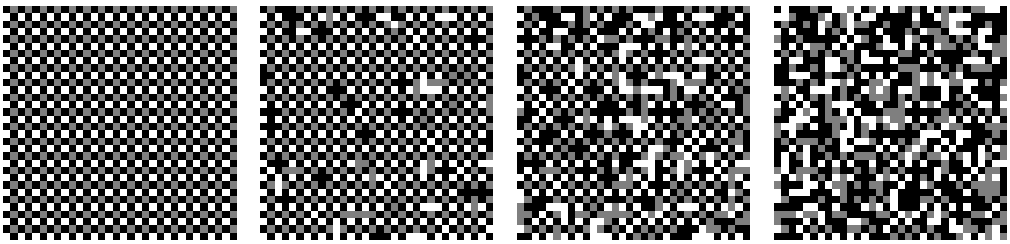


FIGURE 6.12

Three-band filter arrays: (a) periodic pattern, (b-d) randomly permutated versions of the periodic pattern with (b) 50 random pixels, (c) 200 random pixels, and (d) 900 random pixels. © 2006 IEEE

allowed. We produce a numerical score according to the following penalty scores, which is widely used in biological applications:

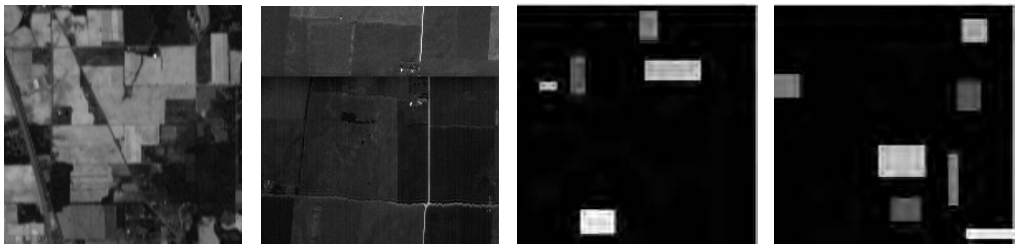
- The penalty for each *match* is 0.
- The penalty for each *swapping* among neighborhood pixels is 1.
- The penalty for each *mismatch* or *substitution* is 2.

We define the summation of minimum *swapping* penalty as *pRPD*, and that of *mismatch* penalty as *pSBD*. For example, if a spectral band has four template superpixels, there are four possible ways to convert them to the same string (or code). Then the *pRPD* and *pSBD* are calculated as the minimum summation of penalty to perform the conversions.

One experiment is conducted by choosing an MSFA generated from the generic method as the initial pattern and then randomly permutating the pixel locations of different spectral bands to produce new random patterns for comparison purpose. The initial pattern tested is a three-band filter array of size 64×64 with probabilities of appearance $\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$ (see Figure 6.12a). The three permutated patterns (Rand1, Rand2 and Rand3) are obtained by randomly permuting 50, 200, and 900 pixels, respectively (see Figure 6.12b to Figure 6.12d). The quantitative comparisons are shown in Table 6.1. Both the SC and CC values in this table show that the initial pattern exhibits the best spatial uniformity and spectral consistency.

TABLE 6.1
Comparison of SC and CC between a three-band MSFA and its three permutated patterns (Rand1, Rand2, Rand3). © 2006 IEEE

	MSFA	Rand1	Rand2	Rand3
SC	0	0.354	0.589	0.760
CC	0.294	0.997	1.000	1.000



(a) (b) (c) (d)

FIGURE 6.13 (See color insert.)

The visualization of the two real multispectral data sets and the corresponding class labels: (a) 92AV3C9 - band 1, (b) FLC1 - band 3, (c) class label of 92AV3C9, red-grass, green-tower, blue-corn, cyan-soil, yellow-hay, (d) class label of FLC1, red-oats, green-corn, blue-red clover, cyan-bare soil, and yellow-wheat.

6.5.2 Evaluation of Mosaicked Multispectral Imaging System

The performance of a certain MSFA demosaicking algorithm can be evaluated from two perspectives: the reconstruction accuracy and the target classification accuracy. There have been several commonly used metrics in literature to measure the reconstruction accuracy, including root mean square error (RMSE), peak signal to noise ratio (PSNR) and subjective comparison, etc. To measure the fidelity of the demosaicked images, we adopt the RMSE metric defined as

$$RMSE = \sqrt{\frac{1}{N_b N_r N_c} \sum_{k=1}^{N_b} \sum_{i=0}^{N_r-1} \sum_{j=0}^{N_c-1} [\hat{f}_k(i,j) - f_k(i,j)]^2}$$

where $\hat{f}_k$ represents the k-th spectral plane of the demosaicked image and f_k that of the original one; N_b, N_r, and N_c denote the number of spectral bands, rows, and columns of the multispectral image, respectively. In order to evaluate the reconstructed images regarding the target detection or recognition performance, classification is carried out on both the full multispectral images and the demosaicked images using a simple k-nearest neighbor (kNN) classifier [66].

Two sets of real multispectral data [67], popularly used in multispectral image analysis, are used to evaluate the proposed method. Figure 6.13a and Figure 6.13b display one spectral band of each data set (Figure 6.13b is only a small segment of the original data set). The 92AV3C9 contains 9 spectral bands selected from a June 1992 AVIRIS data cube [68]. The Flightline C1 (FLC1) image was collected with an airborne scanner in June 1966, which contains 12 spectral bands with the wavelength varying from $0.4\mu m$ to $1.0\mu m$. These two data sets contain a significant number of vegetative species or ground cover classes and have "ground truth" available. We select five ground cover classes from each of the data sets according to the ground truth provided in References [68] and [69]. Figure 6.13c and Figure 6.13d show their corresponding class labels, where the five different colors correspond to the five different classes. For each cover class, we use half of the pixels to train the classifier and the other half serves as the test samples. In addition, we generate eight synthetic data sets by selecting seven spectral bands from the hyperspectral images created by a simulator [70] using the band selection method discussed in Reference [71].

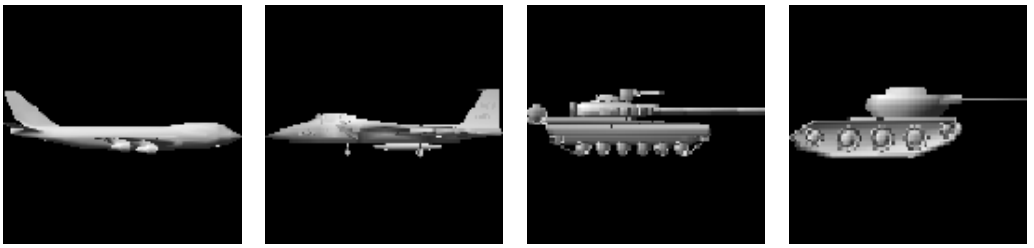


FIGURE 6.14

Four examples of the eight synthetic targets. © 2006 IEEE

Each multispectral image has a different object. Figure 6.14 shows four examples of the eight synthetic targets. We treat each target as one class, which gives us in total eight classes. The training data set also consists of half of the target pixels uniformly selected from each target, and the rest of the target pixels are used as the test data.

To study the MSFA mosaicking and demosaicking performance generalized to different numbers of spectral bands, we form new multispectral images by selecting different numbers of bands from each of the above multispectral data sets. For example, we create five multispectral images from the 92AV3C9 data, and they contain three to seven bands, respectively. The band selection is performed using the multispectral system [67] developed at Purdue University. The created multispectral images are first sampled using the derived MSFAs to generate the mosaicked images. Then we apply different demosaicking algorithms to reconstruct the full multispectral data. We design two sets of experiments to evaluate the performance of the BTES method. In the first experiment, we investigate the effectiveness of incorporating the binary tree and the edge information in the demosaicking process. In the second experiment, we compare the proposed BTES method with three advanced CFA demosaicking approaches published recently.

6.5.2.1 Effectiveness of Binary Tree and Edge Sensing Method

The proposed BTES approach integrates the binary tree-based scheme and the edge-sensing interpolation. In order to investigate the effectiveness of these two components, we implement three demosaicking methods that are variants of BTES, including the classic bilinear interpolation (BI) without using either component, the binary tree-based bilinear interpolation (BTBI), and the edge-sensing interpolation without the binary tree consideration (ES). Edge-sensing based demosaicking methods (i.e., ES and BTES) take into account different weights of each individual neighbors when estimating the missing information, while non-edge-sensing methods simply treat the neighboring pixels equally. Binary tree-based methods (i.e., BTBI and BTES) estimate the missing pixels based on not only *known* MSFA samples, they also use *estimated* MSFA samples obtained following the binary tree structure.

The classification accuracy generated by BTES and its three variants on the real multispectral data is summarized in Table 6.2, and the results of the synthetic data are listed in Table 6.3. Table 6.4 and Table 6.5 show the RMSE of different demosaicking methods on both data sets. From these four tables, we make the following three observations. First of all, among the demosaicking algorithms evaluated, BTES, in most cases and on average,

TABLE 6.2
Classification accuracy (%) of original real multispectral data and reconstructions using different methods. © 2006 IEEE

Images	*92AV3C9*					*FLC1*				
	3-band	4-band	5-band	6-band	7-band	3-band	4-band	5-band	6-band	7-band
ORG	*90.30*	91.80	90.80	*91.63*	89.63	76.74	80.79	82.55	83.19	83.00
BTES	88.96	*93.65*	*91.14*	91.30	*90.47*	*78.43*	*82.56*	*84.39*	*85.93*	*85.44*
ES	88.12	92.30	90.63	89.80	88.46	77.83	82.14	83.88	85.29	84.67
BTBI	86.79	92.81	89.63	90.80	88.63	77.79	82.32	83.75	85.24	84.56
BI	86.78	92.14	90.30	90.97	88.79	77.47	81.84	83.53	84.88	84.17

TABLE 6.3
Classification accuracy (%) of original synthetic data and reconstructions using different methods. © 2006 IEEE

Image	3-band	4-band	5-band	6-band	7-band
ORG	*67.71*	*69.14*	*69.92*	*70.83*	*73.14*
BTES	61.98	66.82	68.79	69.24	70.79
ES	62.04	66.02	66.79	66.48	67.54
BTBI	60.03	65.84	65.90	64.37	65.79
BI	60.54	65.15	65.50	64.80	65.71

outperforms its three variants from both classification accuracy and RMSE perspectives. We also observe that the binary tree-based methods (i.e., BTES and BTBI) outperform the corresponding schemes without binary tree considerations (i.e., ES and BI).

Another important observation is that the classification performance cannot be improved by simply increasing the number of spectral bands. As illustrated in Table 6.2, the four-band image gives the highest accuracy for the 92AV3C9 data, while the six-band image is the best for the FLC1 data. There are two underlying reasons for this phenomenon: first, the newly introduced spectral information does not guarantee to increase the class separability of multispectral data; second, there is a tradeoff between the spectral and the spatial resolution when using the MSFA technique. The extra spectral information is introduced at the cost of reducing the reconstruction performance due to lower spatial resolution. This observation can be further verified by investigating the RMSE of the reconstructed images from Table 6.4 and Table 6.5. It can be seen that the RMSE values increase as the number of spectral bands increases; that is, the lower the spatial resolution, the worse the reconstruction performance.

Our third observation is that the classification accuracy of the demosaicked images is comparable to that of the original data. Interestingly, for the real multispectral scene, in most cases, the reconstructed images present higher classification accuracy. However, this is not true for the synthetic data, for which the original images always generate the highest classification performance. This phenomenon is related to both the characteristics of the selected data sets as well as the intrinsic feature of the mosaicking and the demosaicking process. We realize that the real multispectral images are acquired in real world environment interfered by both the sensor noise and all kinds of other environmental effects, compared to the synthetic data generated with a perfect zero interference. We further notice that

TABLE 6.4
RMSE of reconstructed real multispectral data using different methods. © 2006 IEEE

Images	*92AV3C9*					*FLC1*				
	3-band	4-band	5-band	6-band	7-band	3-band	4-band	5-band	6-band	7-band
BTES	*8.92*	18.88	18.06	*16.96*	*19.28*	*3.50*	*4.27*	*4.48*	*4.65*	*4.77*
ES	9.20	18.85	*17.87*	17.47	19.73	3.85	4.48	4.90	5.24	5.45
BTBI	9.11	*18.82*	18.32	17.30	19.61	4.02	4.60	4.80	5.05	5.21
BI	9.30	18.90	18.39	17.41	19.75	3.99	4.66	4.95	5.27	5.47

TABLE 6.5
RMSE of reconstructed synthetic data using different methods. © 2006 IEEE

Image	3-band	4-band	5-band	6-band	7-band
BTES	*4.10*	*3.78*	*4.36*	*5.40*	*6.63*
ES	4.37	4.05	4.89	6.28	7.80
BTBI	4.39	4.03	4.77	6.16	7.70
BI	4.43	4.07	4.87	6.30	7.88

the mosaicking and the demosaicking process combined together act as a smoothing filter (interpolation of missing pixel information from weighted summation of neighbors), which actually suppresses both noise and outliers in the original images. Therefore, the demosaicked real multispectral images, with less noise and outliers compared with the original data, would be able to generate higher classification accuracy. On the other hand, due to the loss of high frequency information, the demosaicked synthetic data would yield lower classification performance than the original ones, which contain all the information of the demosaicked images.

To validate the above analysis, we add 20dB Gaussian noise to the synthetic images and then perform the mosaicking and demosaicking process. The classification accuracy improvement, defined as $ipr = \frac{acc_{de} - acc_{or}}{acc_{or}} \cdot 100\%$, where acc_{de} and acc_{or} denote the classification accuracy using the demosaicked image and the original data, respectively, with

TABLE 6.6
Classification improvement between demosaicked and original synthetic images of noisy (ipr_n) and noise free (ipr) cases. © 2006 IEEE

Image	3-band	4-band	5-band	6-band	7-band
acc_{or}	67.71	69.14	69.92	70.83	73.14
acc_{de}	61.98	66.82	68.79	69.24	70.79
ipr	*-8.46*	*-3.37*	*-1.62*	*-2.26*	*-3.21*
acc_{orn}	39.67	51.54	56.44	57.00	57.77
acc_{den}	62.18	66.75	67.87	66.58	68.36
ipr_n	*56.75*	*29.5*	*20.25*	*16.82*	*18.34*

and without noise cases is summarized in Table 6.6, in which acc_{den} and acc_{orn} denote the classification accuracy of the noisy data. We relist the classification results of pure signals without noise in Table 6.6 to facilitate comparison. Note that for the images without noise, the classification improvements are all negative, that is, the demosaicked images produce lower classification accuracy than the original data. However, for the noisy data, there exists up to 56.75% improvement on the classification performance of the demosaicked images over the original noisy data. These results verify our previous analysis on why the original data do worse than the demosaicked images. In real world applications, it is impossible to generate a perfect, noise free image. Most likely, the captured images would contain different types of noises, for which the demosaicked images after the process of mosaicking and demosaicking can provide comparable classification performance as the original data.

6.5.2.2 Comparison with Advanced CFA Demosaicking Algorithms

The purpose of this experiment is to evaluate the proposed BTES algorithm with existing rich collection of CFA demosaicking algorithms. We selected three advanced CFA demosaicking approaches [51], [56], [58] recently published in the literature. These techniques effectively utilize the spectral and spatial correlations to suppress artifacts. Algorithm of Reference [51] uses edge-directed interpolation and effectively exploits the color difference correlation, in which the green channel is interpolated first, and the red and blue channels are interpolated with the green band information as a correction term. The postprocessing step uses the color difference information (i.e., green-red and green-blue) to reduce color artifacts. Algorithm of Reference [58] formulates the demosaicking problem as an iterative process of reconstructing correlated signals (i.e., the green plane and the red/blue plane) from their subsampled versions. Another reconstruction approach, Reference [56], introduces wavelet analysis to decompose the original image into detail subbands. The algorithm enforces similar high-frequency information for the three color planes by updating the detail subband of the red and blue channels so that they are within a threshold to that of the green channel.

In order to perform a fair comparison, instead of modifying the algorithms to deal with multiple bands, we choose three adjacent bands (one visual band and two infrared bands) from multispectral images and then treat them as the three color planes. We observe that the visual band contains more detail information, therefore, we use the visual band as the green channel and the other two infrared bands as the red and blue channels. The quantitative comparisons based on the RMSE and the classification accuracy are summarized in Table 6.7 and Table 6.8, respectively. From the RMSE comparison, we see that algorithm

TABLE 6.7
RMSE comparison between BTES and three CFA demosaicking algorithms.

Image	747	dc10	f15	mig	tank0	tank1	tank2	tank3
BTES	3.46	3.77	4.40	**4.08**	6.21	4.11	4.71	3.71
Alg. [56]	7.66	7.37	8.28	8.89	9.64	7.17	8.62	7.33
Alg. [51]	*3.01*	*3.09*	*3.88*	4.79	*5.72*	*2.49*	*4.61*	*2.36*
Alg. [58]	4.67	4.58	5.26	5.71	6.65	4.82	5.60	4.55

TABLE 6.8
Classification accuracy (%) of original and reconstructed image using different demosaicking algorithms. © 2006 IEEE

Alg.	Original	BTES	Alg. [56]	Alg. [51]	Alg. [58]
acc	67.71	61.98	53.77	57.72	50.31

of Reference [51], in general, generates the best results, while the BTES algorithm ranks the second and outperforms algorithm presented in References [56] and [58] by producing lower RMSE. However, by investigating the classification results, we note that the BTES approach performs the best, and gives higher classification accuracy than other CFA demosaicking methods. Algorithm of Reference [51] provides better classification performance than algorithms in References [56] and [58], whose classification accuracy is much lower than that of the original data. In summary, the BTES generic approach provides the highest classification accuracy although with a slightly worse RMSE performance compared to algorithm of Reference [51].

6.6 Conclusions

The primary focus of this chapter is to present a robust and cost-effective solution for multispectral digital cameras. The potential application of MSFA technique is investigated, which uses a mosaic multispectral filter array to cover single CCD sensor resulting in a mosaic-like image. The missing spectral components are reconstructed based on spectral reconstruction algorithms. Two major issues, i.e., the design of MSFAs and the development of effective interpolation algorithms, are discussed in this chapter. The binary tree-driven MSFA generation process guarantees that the pixel distributions of different spectral bands are uniform and highly correlated. These spatial features facilitate the design of the generic demosaicking method based on the same tree, which considers three interrelated issues: band selection, pixel selection and interpolation. The development of a generic algorithm enables the cost-effective multispectral imaging. The experimental results demonstrate that the mosaicking and demosaicking process preserves the classification accuracy effectively for real world data. This result further supports that the MSFA technique is a feasible solution for multispectral cameras.

Acknowledgment

Figure 6.5, Figure 6.10 to Figure 6.12, and Table 6.1 are reprinted from Reference [18], Figure 6.6, Figure 6.7, and Table 6.2 to Table 6.8 are reprinted from Reference [19], with the permission of IEEE.

References

[1] P. Colarusso, L.H. Kidder, I.W. Levin, J.C. Fraser, J.F. Arens, and E.N. Lewis, "Infrared spectroscopic imaging: From planetary to cellular systems," *Applied Spectroscopy*, vol. 52, no. 3, pp. 106A–120A, March 1998.

[2] P.J. Curran, "Imaging spectrometry," *Progress in Physical Geography*, vol. 18, no. 2, pp. 247–266, June 1994.

[3] G.A. Clark, S.K. Sengupta, W.D. Aimonetti, F.Roeske, and J.D. Donetti, "Multispectral image feature selection for land mine detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 1, pp. 304–311, January 2000.

[4] J.S. Salazar, M.W. Koch, and D.A. Yocky, "A novel automatic target recognition approach for multispectral data," in *Proceedings of the SPIE Conference on Imaging Spectrometry VII*, Seattle, Washington, July 2002, vol. 4816, pp. 222–241.

[5] M.E. Dickinson, G. Bearman, S. Tille, R. Lansford, and S.E. Fraser, "Multi-spectral imaging and linear unmixing add a whole new dimension to laser scanning fluorescence microscopy," *Biotechniques*, vol. 31, no. 6, pp. 1274–1276, June 2001.

[6] T. Haraguchi, T. Shimi, T. Koujin, N. Hashiguchi, and Y. Hiraoka, "Spectral imaging fluorescence microscopy," *Genes to Cells*, vol. 7, no. 9, pp. 881–887, September 2002.

[7] Y. Hiraoka, T. Shimi, and T. Haraguchi, "Multispectral imaging fluorescence microscopy for living cells," *Cell Structure and Function*, vol. 27, no. 5, pp. 367–374, October 2002.

[8] T. Zimmermann, J. Rietdorf, and R. Pepperkok, "Spectral imaging and its applications in live cell microscopy," *Federation of European Biochemical Societies Letters*, vol. 546, no. 1, pp. 87–92, July 2003.

[9] H. Qi and N.A. Diakides, "Thermal infrared imaging in early breast cancer detection-a survey of recent research," in *Proceedings of the IEEE International Conference on Engineering in Medicine and Biology Society*, Cancun, Mexico, September 2003, vol. II, pp. 1109–1112.

[10] H. Szu, L. Miao, and H. Qi, "Thermodynamic free-energy minimization for unsupervised fusion of dual-color infrared breast images," in *Proceedings of the Independent Component Analyses, Wavelets, Unsupervised Smart Sensors, and Neural Networks IV at SPIE Defense and Security Symposium*, Orlando, FL, USA, April 2006, vol. 6247, pp. 62470P:1–15.

[11] A. Abrardo, "Color constancy from mulitspectral images," in *Proceedings of the IEEE International Conference on Image Processing*, Kobe, Japan, October 1999, vol. 3, pp. 570–574.

[12] H.M.G. Stokman, T. Gevers, and J.J. Koenderink, "Color measurement by imaging spectrometry," *Computer Vision and Image Understanding*, vol. 79, no. 2, pp. 236–249, August 2000.

[13] M. Yamaguchi, T. Teraji, K. Ohsawa, T. Uchiyama, H. Motomura, Y. Murakami, and N. Ohyama, "Color image reproduction based on the multispectral and multiprimary imaging: Experimental evaluation," in *Proceedings of the SPIE Conference on Color Imaging: Device Independent Color, Color Hardcopy and Applications VII*, San Jose, CA, USA, January 2002, vol. 4663, pp. 15–26.

[14] Y.Y. Schechner and S.K. Nayar, "Generalized mosaicing: Wild field of view multispectral imaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 10, pp. 1334–1348, October 2002.

[15] R.B. Smith, "Introduction to hyperspectral imaging." Available online: http://www.microimages.com/getstart/pdf_new/hyprspec.pdf, 2006.

[16] D.A. Scribner, J. Schuler, and M.R. Kruer, "Infrared multispectral sensors: Re-considering typical design assumptions." Naval Research Lab., Code 5636, 1998.

[17] K. Parulski and K.E. Spaulding, *Digital Color Imaging Handbook*, ch. Color image processing for digital cameras, G. Sharma (ed.), Boca Raton, FL: CRC Press, 2002, pp. 728–757.

[18] L. Miao and H. Qi, "The design and evaluation of a generic method for generating mosaicked multispectral filter arrays," *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2780–2791, September 2006.

[19] L. Miao, H. Qi, R. Ramanath, and W.E. Snyder, "Binary tree-based generic demosaicking algorithm for multispectral filter arrays," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3550–3558, November 2006.

[20] P.I. Shnitser, I.P. Agurok, S. Sandomirsky, and A. Avakian, "Spectrally adaptive imaging camera for automatic target contrast enhancement," in *Proceedings of the SPIE Conference on Algorithms for Multispectral and Hyperspectral Imagery V*, Orlando, FL, USA, April 1999, vol. 3717, pp. 185–195.

[21] D.H. Kim, K. Kolesnikov, A. Kostrzewski, G.S.A.A. Vasiliev, and M.A. Vorontsov, "Adaptive imaging system using image quality metric based on statistical analysis of speckle fields," in *Proceedings of the SPIE Conference on Hybrid Image and Signal Processing VII*, Orlando, FL, USA, April 2000, vol. 4044, pp. 177–186.

[22] Y. Jiao, S.R. Bhalotra, H.L. Kung, and D.A. Miller, "Adaptive imaging spectrometer in a time-domain filtering architecture," *Optics-Express*, vol. 11, no. 17, pp. 1960–1965, August 2003.

[23] "Adaptive focal plane array." Available online: http://www.darpa.mil/mto/afpa/, 2003.

[24] B.K. Gunturk, J. Glotzbach, Y. Altunbasak, and R.W. Schaffer, "Demosaicking: Color filter array interpolation," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 44–54, January 2005.

[25] R. Lukac and K.N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1260–1267, November 2005.

[26] K. Hirakawa and P. Wolfe, "Spatio-spectral color filter array for enhanced image fidelity." in *Proceedings of the IEEE International Conference on Image Processing*, San Antonio, TX, USA, September 2007, vol. II, pp. 81–84.

[27] E.B. Bayer, "Color imaging array." U.S. Patent 3 971 065, July 1976.

[28] "Fillfactory: The color filter array faq." Available online: http://www.fillfactory.com/htm/technology/htm/rgbfaq.htm.

[29] "Sony press release." Available online: http://www.sony.net/SonyInfo/News/Press/200307/03-029E/.

[30] T. Yamagami, T. Sasaki, and A. Suga, "Image signal processing apparatus having a color filter with offset luminance filter elements." U.S. Patent 5 323 233, June 1994.

[31] J.F. Hamilton, J.E. Adams, and D.M. Orlicki, "Particular pattern of pixels for a color filter array which is used to derive luminance and chrominance values," U.S. Patent 6 330 029 B1, December 2001.

[32] E.B. Gindele and A.C. Gallagher, "Sparsely sampled image sensing device with color and luminance photosites," U.S. Patent 6 476 865 B1, November 2002.

[33] O. Packer and D.R. Williams, *The Science of Color*. Amsterdam, Boston: Elsevier, 2003.

[34] S. Otake, P.D. Gowdy, and C.M. Cicerone, "The spatial arrangement of l and m cones in the peripheral human retina," *Vision Research*, vol. 40, no. 6, pp. 677–693, March 2000.

[35] A. Roorda, A.B. Metha, P. Lennie, and D.R. Williams, "Packing arrangement of the three cone classes in primate retina," *Vision Research*, vol. 41, no. 10-11, pp. 1291–1306, May 2001.

[36] A.S. French, A.W. Snyder, and D.G. Stavenga, "Image degradation by an irregular retinal mosaic," *Biological Cybernetics*, vol. 27, no. 4, pp. 229–233, December 1977.

[37] M. Siuta, "Color vision in fish." Available online: http://instruct1.cit.cornell.edu/courses/bionb424/students2004/mas262/neuroanatomy.htm.

[38] G.S. Losey, T.W. Cronin, T.H. Goldsmith, and D. Hyde, "The UV visual world of fishes: A review," *Journal of Fish Biology*, vol. 54, no. 5, pp. 921–943, May 1999.

[39] P.A. Raymond, L.K. Barthel, and G.A. Curran, "Developmental patterning of rod and cone photoreceptors in embryonic zebrafish," *Journal of Comparative Neurology*, vol. 359, no. 4, pp. 537–550, September 1995.

[40] S. Thoya, A. Mochizuki, and Y. Iwasa, "Formation of cone mosaic of zebrafish retina," *Journal of Theoretical Biology*, vol. 200, no. 2, pp. 231–244, September 1999.

[41] Y. Fei, "Development of the cone photoreceptor mosaic in the mouse retina revealed by fluorescent cones in transgenic mice," *Molecular Vision*, vol. 9, no. 6, pp. 31–42, February 2003.

[42] M.A. Raven and B.E. Reese, "Mosaic regularity of horizontal cells in the mouse retina is independent of cone photoreceptor innervation," *Investigative Ophthalmology and Visual Science*, vol. 44, no. 3, pp. 965–973, March 2003.

[43] N. Keshava, "Best bands selection for detection in hyperspectral processing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, AZ, USA, May 2001, vol. V, pp. 3149–3152.

[44] S.B. Serpico and L. Bruzzone, "A new search algorithm for feature selection in hyperspectral remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 7, pp. 1360–1367, July 2001.

[45] J.C. Price, "Spectral band selection for visible-near infrared remote sensing: Spectral-spatial resolution tradeoffs," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 35, no. 5, pp. 1277–1285, September 1997.

[46] T.M. Tu, C.H. Chen, J.L. Wu, and C.I. Chang, "A fast two-stage classification method for high-dimensional remote sensing data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 36, no. 1, pp. 182–191, January 1998.

[47] S.G. Bajwa, P. Bajcsy, P. Groves, and L.F. Tian, "Hyperspectral image data mining for band selection in agricultural applications," *Transactions of the American Society of Agricultural Engineers.*, vol. 47, no. 3, pp. 895–907, May / June 2004.

[48] S.W. Grotta, "Anatomy of a digital camera: Image sensors," Available online: http://www.extremetech.com/article2/0,3973,15465,00.asp, June 2001.

[49] D. Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal." U.S. Patent 4 642 678, February 1987.

[50] R. Lukac, K. Martin, and K.N. Plataniotis, "Demosaicked image postprocessing using local color ratios," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 914–920, June 2004.

[51] W. Lu and Y.P. Tan, "Color filter array demosaicking: New method and performance measures," *IEEE Transactions on Image Processing*, vol. 12, no. 10, pp. 1194–1210, October 2003.

[52] K. Hirakawa and T.W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 360–369, March 2005.

[53] P. Scheunders, "An orthogonal wavelet representation of multivalued images," *IEEE Transactions on Image Processing*, vol. 12, no. 6, pp. 718–725, June 2003.

[54] H.J. Trussell and R.E. Hartwig, "Mathematics for demosaicking," *IEEE Transactions on Image Processing*, vol. 11, no. 4, pp. 485–492, April 2002.

[55] X. Li and M.T. Orchard, "New edge-directed interpolation," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1521–1527, October 2001.

[56] B.K. Gunturk, Y. Altunbasak, and R.M. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 997–1013, September 2002.

[57] X. Wu and N. Zhang, "Primary-consistent soft-decision color demosaicking for digital cameras," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1263–1274, September 2004.

[58] X. Li, "Demosaicing by successive approximation," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 370–379, March 2005.

[59] R. Kimmel, "Demosaicing: Image reconstruction from color ccd samples," *IEEE Transactions on Image Processing*, vol. 8, no. 9, pp. 1221–1228, September 1999.

[60] S.C. Pei and I.K. Tam, "Effective color interpolation in CCD color filter arrays using signal correlation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 6, pp. 503–513, June 2003.

[61] X. Wu, W.K. Choi, and P. Bao, "Color restoration from digital camera data by pattern matching," *Proceedings of the SPIE*, vol. 3018, pp. 12–17, April 1997.

[62] L. Chang and Y.P. Tan, "Effective use of spatial and spectral correlations for color filter array demosaicking," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 355–365, February 2004.

[63] J. Driesen and P. Scheunders, "Wavelet-based color filter array demosaicking," in *Proceedings of the IEEE International Conference on Image Processing*, Singapore, October 2004, vol. V, pp. 3311–3314.

[64] L. Chen, K.H. Yap, and Y. He, "Color filter array demosaicking using wavelet-based subband synthesis," in *Proceedings of the IEEE International Conference on Image Processing*, Genoa, Italy, September 2005, vol. II, pp. 1002–1005.

[65] V.I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Doklady Akademii Nauk SSSR*, vol. 163, no. 4, pp. 845–848, 1965.

[66] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley-Interscience, 2000.

[67] "Laboratory for applications of remote sensing." Available online: http://www.lars.purdue.edu.

[68] D. Landgrebe, "Multispectral data analysis: A signal theory perspective." Available online: http://dynamo.ecn.purdue.edu/ biehl/MultiSpec/Signal_Theory.pdf, 1998.

[69] D. Landgrebe, "Multispectral data analysis: A moderate dimension example." Available online: http://dynamo.ecn.purdue.edu/ biehl/MultiSpec/Moderate_Dimension.pdf, 1997.

[70] R. Ramanath, *A Framework for Object-characterization and Matching in Multi- and Hyperspectral Imaging Systems*. Ph.D. thesis, North Carolina State University, 2003.

[71] R. Ramanath, W.E. Snyder, and H. Qi, "Mosaic multispectral focal plane array cameras," in *Proceedings of the SPIE Defense and Security Symposium*, Orlando, FL, USA, April 2004, vol. 5406, pp. 701–712.

7

Color Filter Array Sampling of Color Images: Frequency-Domain Analysis and Associated Demosaicking Algorithms

Eric Dubois

7.1 Introduction

Color-filter-array (CFA) sampling of color images involves a spatial-domain multiplexing of three or more color components of a color image, each on a subset of the lattice consisting of all sensor elements. In the frequency domain, this same operation can be viewed as the frequency-domain multiplexing of a luma component at baseband and two or more chrominance components centered at certain spatial modulation frequencies. This view leads to some very efficient demosaicking algorithms that would not normally be evident from the spatial-domain representation. This chapter presents the frequency-domain representation for general periodic CFA structures and describes efficient demosaicking algorithms based on spatial filtering derived from this representation.

The chapter is organized as follows. Section 7.2 describes how the geometric structure of a CFA pattern can be specified using the concepts of lattices, sublattices and cosets, and represented using several matrices. Section 7.3 presents the model for the formation of the CFA image and derives the frequency-domain representation for an arbitrary periodic CFA pattern. Three specific examples are examined in detail in addition to the popular Bayer CFA pattern. Section 7.4 then addresses the demosaicking problem and presents algorithms inspired by the frequency-domain representation, namely frequency-division demultiplexing of the luma and modulated chrominance components. The least-squares approach to design the filters used in this demosaicking structure is given in Section 7.5 along with a few examples. Some concluding remarks are given in Section 7.6. The theory of lattices is extensively used in this chapter. A summary of the main notation and properties required is presented in the appendix; more details can be found in References [1] and [2].

7.2 Geometric Structure of the Color-Filter Array

In typical image sensors such as the charge-coupled devices (CCDs), the image window $\mathscr{W}$ is partitioned into a set of sensor elements of the same shape, usually rectangular. Each of these sensor elements is assigned to one of C classes, according to the characteristics of an optical filter placed over that sensor element. For example, the conventional Bayer CFA has three classes, corresponding to red (R), green (G) and blue (B) filters. The sensor elements are assumed to lie on a lattice Λ, and the shape of each sensor element is a subset of a unit cell $\mathscr{P}$ of Λ. The sensor elements are slightly smaller than the unit cell to allow for wiring, but this effect will be ignored without loss of generality in this presentation. A general description of CFA sensors and some specific CFA patterns can be found in References [3] and [4], and an evaluation of several RGB CFA patterns is given in Reference [5]. Several authors have used stacked-matrix formulations to describe CFA patterns (e.g., References [6] and [7]), but in this chapter we give a presentation based on lattices. Figure 7.1 illustrates the setup for the Bayer structure, showing the upper-left corner of the image window. The lattice Λ is a rectangular lattice with equal horizontal and vertical sample spacing X, and the unit cell $\mathscr{P}$ is a square of size $X \times X$. The origin of the coordinate system is placed at the center of the upper-left sensor element, with the y-axis pointing downward. To simplify notation, the unit-cell dimension X is taken as the unit of length, called the pixel height (px), i.e., $X = 1$ px. With this choice, the lattice Λ is simply the integer Cartesian lattice $\mathbb{Z}^2$.

The CFA pattern is assumed to be regular and periodic, as most are. The present development does not apply to non-periodic CFA structures, such as non-periodic pseudo-random structures, but it does apply to periodic ones (see Reference [8] for an example of a periodic pseudo-random CFA). One period of the pattern is replicated on the points of a sublattice Γ of Λ. The number of sensor elements in one period is equal to the index of Γ in Λ, denoted $K = (\Lambda : \Gamma)$. For the example of Figure 7.1, we have $\Gamma = (2\mathbb{Z})^2$, and $(\Lambda : \Gamma) = 4$; one period is indicated by the heavy square in the upper left. Each sensor element in the basic period corresponds to one coset of Γ in Λ. We denote the coset representatives belonging to this

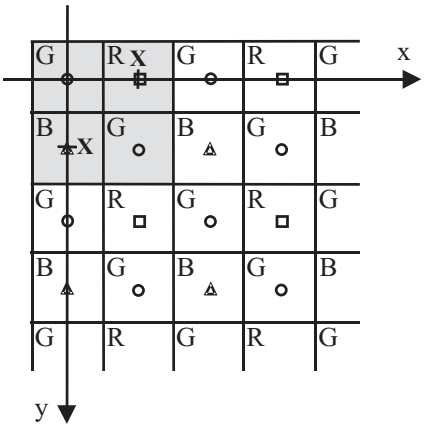


FIGURE 7.1

Upper-left portion of the Bayer CFA sampling structure, showing the constituent sampling structures Ψ_R (□), Ψ_G (○) and Ψ_B (△). The union of these three sampling structures forms the lattice Λ.

basic period as $\mathbf{b}_k, k = 1, \ldots, K$; the corresponding cosets are $\mathbf{b}_k + \Gamma, k = 1, \ldots, K$. The set of coset representatives can be compactly represented with a $2 \times K$ matrix $\mathbf{B} = [\mathbf{b}_1\, \mathbf{b}_2 \ldots \mathbf{b}_K]$. For the Bayer lattice of Figure 7.1, we can choose $\mathbf{b}_1 = [0\ 0]^T$, $\mathbf{b}_2 = [1\ 0]^T$, $\mathbf{b}_3 = [0\ 1]^T$, $\mathbf{b}_4 = [1\ 1]^T$. Each sensor class is associated with one or more of these cosets.

Let the sampling structure for sensor class i be denoted $\Psi_i \subset \Lambda$. By definition, $\Lambda = \bigcup_{i=1}^{C} \Psi_i$ where the Ψ_i are disjoint subsets of Λ, $\Psi_i \cap \Psi_j = \emptyset$ for $i \neq j$. Each of the sampling structures is a union of selected cosets of Γ in Λ. If we define $\mathscr{B}_j = \{k \mid \mathbf{b}_k \in \Psi_j\}$, then we have

$$\Psi_j = \bigcup_{k \in \mathscr{B}_j} (\mathbf{b}_k + \Gamma). \tag{7.1}$$

For the Bayer CFA, $\mathscr{B}_R = \{2\}$, $\mathscr{B}_G = \{1,4\}$, $\mathscr{B}_B = \{3\}$. We can then write explicitly $\Psi_R = (\mathbf{b}_2 + \Gamma)$, $\Psi_G = (\mathbf{b}_1 + \Gamma) \cup (\mathbf{b}_4 + \Gamma) = \Gamma \cup (\mathbf{b}_4 + \Gamma)$, $\Psi_B = (\mathbf{b}_3 + \Gamma)$. There is no unique choice of the $\mathbf{b}_i$ but there is often a natural one, such as the one indicated above. Also, the indexing of the coset representatives is arbitrary, but we choose $\mathbf{b}_1 = \mathbf{0}$. The assignment of sensor classes to cosets can be summarized by a $K \times C$ matrix $\mathbf{J}$ defined by

$$[\mathbf{J}]_{ji} = \begin{cases} 1 & \text{if } j \in \mathscr{B}_i \\ 0 & \text{otherwise} \end{cases} \qquad j = 1, \ldots, K; i = 1, \ldots, C. \tag{7.2}$$

In summary, the geometric structure of a CFA sensor is captured by the number of sensor classes C, the sensor lattice Λ represented by a sampling matrix $\mathbf{V}_\Lambda$, the CFA periodicity lattice Γ represented by a sampling matrix $\mathbf{V}_\Gamma$, and the matrix $\mathbf{J}$ assigning sensor classes to cosets of Γ in Λ.

7.3 Formation and Representation of the CFA Image

7.3.1 Formation of the CFA Image

Assume that $f(\mathbf{x},\lambda)$ is the spectral light intensity (irradiance) projected at position $\mathbf{x}$ on the plane containing the image sensor by an ideal (pinhole) optical system. A single value $f_{\mathrm{CFA}}[\mathbf{x}]$ is measured at each point of $\Lambda \cap \mathscr{W}$, and is approximated by

$$f_{\mathrm{CFA}}[\mathbf{x}] = \int_{\lambda_{\min}}^{\lambda_{\max}} \int_{\mathbb{R}^2} f(\mathbf{x}-\mathbf{s},\lambda) h_a(\mathbf{s})\, d\mathbf{s}\, c_i(\lambda)\, d\lambda, \quad \mathbf{x} \in \Psi_i \cap \mathscr{W}, i = 1,\ldots,C. \tag{7.3}$$

The spatial convolution with h_a accounts for both blurring by the optical system and integration of the optically / spectrally-filtered light irradiance over one sensor element. The filters placed over sensor elements of class i have spectral transmission curve $c_i(\lambda)$, which would also include the effect of any global filter placed in the optical path that affects all classes. It is assumed that all the filters have negligible transmission below $\lambda_{\min}$ and above $\lambda_{\max}$ which correspond to the spectral limits of the human visual system. A typical set of filter spectral responses for RGB can be found in Reference [4]; these responses include a global infrared-stop filter. Note that the measurement of image values may also include a pointwise nonlinearity such as gamma correction [9]. We do not account for such nonlinearities in this chapter since they do not strongly influence the demosaicking process, but a system designer must be aware of them and handle them correctly.

We define $f_i[\mathbf{x}]$ to be the component corresponding to the i^{th} sensor class, defined on the entire lattice Λ,

$$f_i[\mathbf{x}] = \int_{\lambda_{\min}}^{\lambda_{\max}} \int_{\mathbb{R}^2} f(\mathbf{x}-\mathbf{s},\lambda) h_a(\mathbf{s})\, d\mathbf{s}\, c_i(\lambda)\, d\lambda, \quad \mathbf{x} \in \Lambda \cap \mathscr{W}, i = 1,\ldots,C. \tag{7.4}$$

Of course, the signal $f_i[\mathbf{x}]$ is not measured or available off Ψ_i, i.e., on the points $\Lambda \backslash \Psi_i$, and it is necessary to estimate $f_i[\mathbf{x}]$ at these points. The CFA signal can be expressed as

$$f_{\mathrm{CFA}}[\mathbf{x}] = \sum_{i=1}^{C} f_i[\mathbf{x}] m_i[\mathbf{x}], \tag{7.5}$$

where $m_i[\mathbf{x}]$ is the indicator function for Ψ_i,

$$m_i[\mathbf{x}] = \begin{cases} 1, & \mathbf{x} \in \Psi_i; \\ 0, & \mathbf{x} \in \Lambda \backslash \Psi_i. \end{cases} \tag{7.6}$$

This model for the formation of the CFA signal is illustrated in the top portion of Figure 7.2.

7.3.2 Frequency-Domain Representation of the CFA Image

Each of the $m_i[\mathbf{x}]$ is a periodic function on Λ, with periodicity given by Γ (i.e., $m_i[\mathbf{x}+\mathbf{y}] = m_i[\mathbf{x}]$ for all $\mathbf{y} \in \Gamma$), and so can be expressed as a discrete Fourier series. Since Γ is a sublattice of Λ, there is an inverse relationship for the reciprocal lattices, namely $\Lambda^* \subset \Gamma^*$,

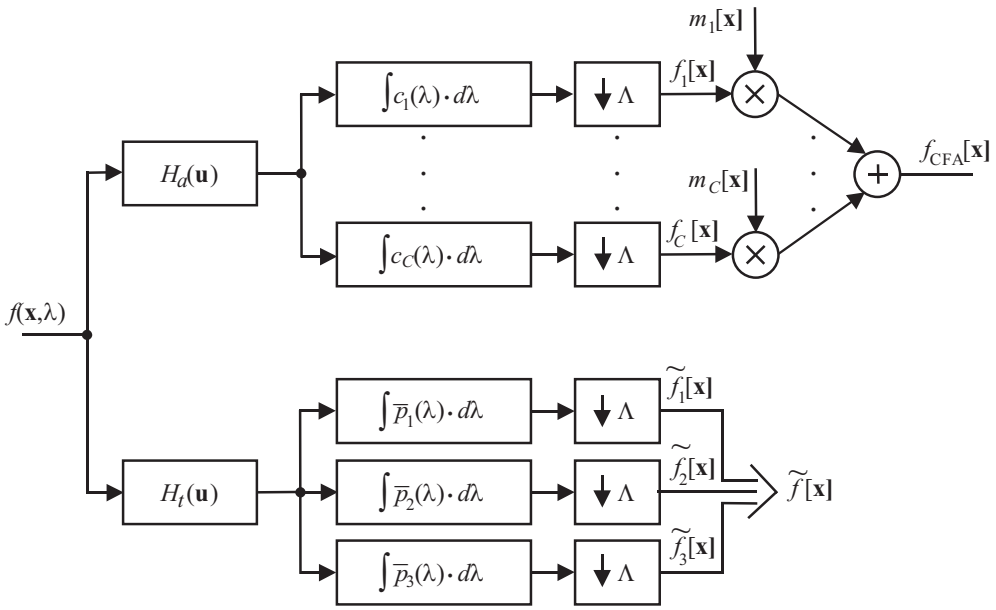


FIGURE 7.2

Block diagram of CFA camera (top) and ideal camera for human observer (bottom).

with $(\Gamma^* : \Lambda^*) = K$. Let $\{\mathbf{d}_1, \ldots, \mathbf{d}_K\}$ be a set of coset representatives for the K cosets of Λ^* in Γ^*, with $\mathbf{D}$ the $2 \times K$ matrix $\mathbf{D} = [\mathbf{d}_1\, \mathbf{d}_2 \ldots \mathbf{d}_K]$. Again, this choice of coset representatives is not unique, but we choose $\mathbf{d}_1 = \mathbf{0}$ and choose the others to lie in a Voronoi unit cell of Λ^*. Then, the discrete Fourier series representation of $m_i[\mathbf{x}]$ is given by [2]:

$$m_i[\mathbf{x}] = \sum_{k=1}^{K} M_{ki} \exp(j2\pi \mathbf{x} \cdot \mathbf{d}_k) \tag{7.7}$$

where

$$M_{ki} = \frac{1}{K} \sum_{j=1}^{K} m_i[\mathbf{b}_j] \exp(-j2\pi \mathbf{b}_j \cdot \mathbf{d}_k). \tag{7.8}$$

We can limit the sum to the non-zero terms only,

$$M_{ki} = \frac{1}{K} \sum_{j \in \mathscr{B}_i} \exp(-j2\pi \mathbf{b}_j \cdot \mathbf{d}_k). \tag{7.9}$$

We can equivalently define the binary matrix $\mathbf{J}$ of Equation 7.2 by $[\mathbf{J}]_{ji} = m_i[\mathbf{b}_j]$; then we can define the matrix $\mathbf{M} = [M_{ki}]$ by

$$\mathbf{M} = \frac{1}{K} [\exp(-j2\pi \mathbf{D}^T \mathbf{B})] \mathbf{J} \tag{7.10}$$

where the exponential of the matrix is carried out term by term, and postmultiplication by $\mathbf{J}$ is matrix multiplication.

With this representation, we can express the CFA signal as

$$\begin{aligned} f_{\text{CFA}}[\mathbf{x}] &= \sum_{i=1}^{C} f_i[\mathbf{x}] \sum_{k=1}^{K} M_{ki} \exp(j2\pi\mathbf{x}\cdot\mathbf{d}_k) \\ &= \sum_{k=1}^{K} \left(\sum_{i=1}^{C} M_{ki} f_i[\mathbf{x}] \right) \exp(j2\pi\mathbf{x}\cdot\mathbf{d}_k) \\ &= \sum_{k=1}^{K} q_k[\mathbf{x}] \exp(j2\pi\mathbf{x}\cdot\mathbf{d}_k) \\ &= \sum_{k=1}^{K} r_k[\mathbf{x}] \end{aligned} \tag{7.11}$$

where we have identified the new signals

$$q_k[\mathbf{x}] = \sum_{i=1}^{C} M_{ki} f_i[\mathbf{x}], \quad k = 1, \ldots, K, \quad \text{or} \tag{7.12}$$

$$\mathbf{q}[\mathbf{x}] = \mathbf{M}\mathbf{f}[\mathbf{x}], \tag{7.13}$$

which are different linear combinations of the original components. Here, $\mathbf{q}[\mathbf{x}] = [q_1[\mathbf{x}] \ldots q_K[\mathbf{x}]]^T$ and $\mathbf{f}[\mathbf{x}] = [f_1[\mathbf{x}] \ldots f_C[\mathbf{x}]]^T$. The $r_k[\mathbf{x}] = q_k[\mathbf{x}] \exp(j2\pi\mathbf{x}\cdot\mathbf{d}_k)$ are the modulated versions of these components. The matrix $\mathbf{M}$ represents a linear transformation from $\mathbb{R}^C$ to $\mathbb{R}^K$, where $C \leq K$. If $C = K$, this transformation is invertible using the matrix inverse. If $C < K$, the transformation can be inverted using the pseudo-inverse [10] as follows:

$$\mathbf{f}[\mathbf{x}] = (\mathbf{M}^H\mathbf{M})^{-1}\mathbf{M}^H\mathbf{q}[\mathbf{x}] = \mathbf{M}^\dagger\mathbf{q}[\mathbf{x}], \tag{7.14}$$

where H denotes conjugate transpose of a matrix (the conjugate must be used if $\mathbf{M}$ is complex). This expression gives the least-squares estimate if $\mathbf{q}$ is not in the range (column space) of the matrix $\mathbf{M}$.

Taking the Fourier transform of Equation 7.11, and using the standard modulation property,

$$F_{\text{CFA}}(\mathbf{u}) = \sum_{k=1}^{K} Q_k(\mathbf{u} - \mathbf{d}_k). \tag{7.15}$$

Noting that $\mathbf{d}_1 = \mathbf{0}$ and that the other $\mathbf{d}_k$ are non-zero, the CFA signal is the sum of $q_1[\mathbf{x}]$ at zero frequency (DC) and each of the $q_k[\mathbf{x}]$ modulated at non-zero frequency $\mathbf{d}_k$. The DC (or baseband) component $q_1[\mathbf{x}]$ has a particularly simple form, since $\mathbf{d}_1 = \mathbf{0}$. From Equation 7.9,

$$M_{1i} = \frac{1}{K} \sum_{j \in \mathscr{B}_i} \exp(0) = \frac{|\mathscr{B}_i|}{K} \tag{7.16}$$

where $|\mathscr{B}_i|$ represents the number of elements in the set $\mathscr{B}_i$. In other words, the baseband component is a weighted sum of the original components, where the positive weights are the relative sampling densities of the corresponding components. Since the sum of these weights is 1.0, we see that if all the input components are equal, then the baseband component is equal to the CFA signal, which is the same as all the individual input components.

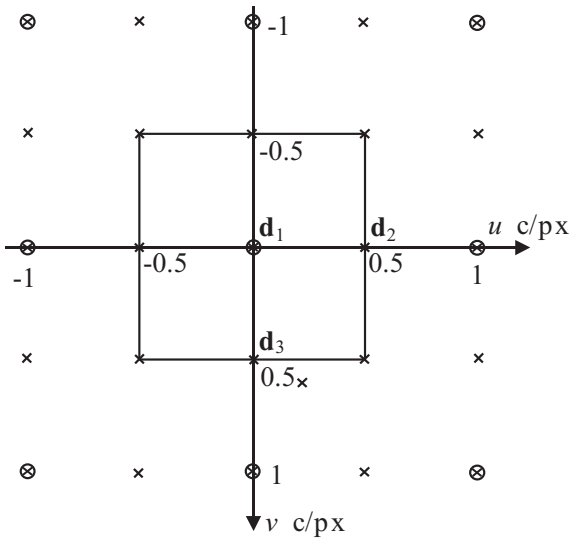


FIGURE 7.3

Reciprocal lattices Λ^* ($\circ$) and Γ^* ($\times$) and a suitable choice of representatives for the cosets of Λ^* in Γ^* for the Bayer sampling structure.

This formulation has already been reported for the Bayer RGB CFA mosaic [11], which is a simplification of an equivalent formulation previously reported in Reference [12]. In general, a CFA signal is characterized in the frequency domain by the set of modulating frequencies $\{\mathbf{d}_k, k = 1, \ldots, K\}$ and the matrix $\mathbf{M} = [M_{ki}]$ that defines the transformed components via the matrix equation $\mathbf{q}[\mathbf{x}] = \mathbf{Mf}[\mathbf{x}]$, which are all determined from the geometric structure of the CFA pattern. Viewed from the frequency domain perspective, the CFA signal is equivalent to a frequency division multiplexing of the $q_k[\mathbf{x}]$. The implied demosaicking algorithm involves separating these components and then obtaining the desired components by a matrixing operation.

The above development can be illustrated by continuing the example of the Bayer RGB CFA to reproduce the results presented in Reference [11], but with the notation of this chapter. The coset representatives of Γ in Λ were given above, resulting in

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{7.17}$$

The reciprocal lattices are easily seen to be $\Lambda^* = \mathbb{Z}^2$ and $\Gamma^* = (\frac{1}{2}\mathbb{Z})^2$. Figure 7.3 illustrates these reciprocal lattices and a suitable choice of coset representatives for Λ^* in Γ^*, yielding the matrix

$$\mathbf{D} = \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}. \tag{7.18}$$

The matrix $\mathbf{J}$ defining the three input channels is

$$\mathbf{J} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{7.19}$$

so that application of Equation 7.10 yields

$$\mathbf{M} = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & -1 \\ -1 & 2 & -1 \end{bmatrix}. \tag{7.20}$$

We then see that the four transformed signals q_i are given explicitly by

$$q_1[\mathbf{x}] = \tfrac{1}{4} f_1[\mathbf{x}] + \tfrac{1}{2} f_2[\mathbf{x}] + \tfrac{1}{4} f_3[\mathbf{x}] \tag{7.21}$$
$$q_2[\mathbf{x}] = -\tfrac{1}{4} f_1[\mathbf{x}] + \tfrac{1}{4} f_3[\mathbf{x}] \tag{7.22}$$
$$q_3[\mathbf{x}] = \tfrac{1}{4} f_1[\mathbf{x}] - \tfrac{1}{4} f_3[\mathbf{x}] \tag{7.23}$$
$$q_4[\mathbf{x}] = -\tfrac{1}{4} f_1[\mathbf{x}] + \tfrac{1}{2} f_2[\mathbf{x}] - \tfrac{1}{4} f_3[\mathbf{x}]. \tag{7.24}$$

Note that $q_3 = -q_2$. The pseudo-inverse $\mathbf{M}^\dagger$ is given by

$$\mathbf{M}^\dagger = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & -1 & -1 \end{bmatrix} \tag{7.25}$$

so that the inverse relationship is

$$f_1[\mathbf{x}] = q_1[\mathbf{x}] - q_2[\mathbf{x}] + q_3[\mathbf{x}] - q_4[\mathbf{x}] \tag{7.26}$$
$$f_2[\mathbf{x}] = q_1[\mathbf{x}] + q_4[\mathbf{x}] \tag{7.27}$$
$$f_3[\mathbf{x}] = q_1[\mathbf{x}] + q_2[\mathbf{x}] - q_3[\mathbf{x}] - q_4[\mathbf{x}]. \tag{7.28}$$

Imposing the constraint $q_3[\mathbf{x}] = -q_2[\mathbf{x}]$, this simplifies to

$$f_1[\mathbf{x}] = q_1[\mathbf{x}] - 2q_2[\mathbf{x}] - q_4[\mathbf{x}] \tag{7.29}$$
$$f_2[\mathbf{x}] = q_1[\mathbf{x}] + q_4[\mathbf{x}] \tag{7.30}$$
$$f_3[\mathbf{x}] = q_1[\mathbf{x}] + 2q_2[\mathbf{x}] - q_4[\mathbf{x}]. \tag{7.31}$$

These four transformed signals are modulated at the frequencies $(0.0, 0.0)$, $(0.5, 0.0)$, $(0.0, 0.5)$ and $(0.5, 0.5)$ (obtained from $\mathbf{D}$), so that

$$F_{\mathrm{CFA}}(u,v) = Q_1(u,v) + Q_2(u-0.5,v) - Q_2(u,v-0.5) + Q_4(u-0.5,v-0.5). \tag{7.32}$$

We note that there are two separate and independent copies of $Q_2(u,v)$ at $(0.5, 0.0)$ and $(0.0, 0.5)$ respectively. The input components f_1, f_2 and f_3 correspond respectively to f_R,

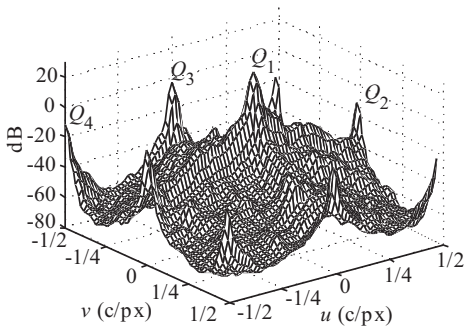


FIGURE 7.4

Two-dimensional power density spectrum estimate of a CFA image with the Bayer sampling structure.

f_G and f_B in Reference [11], and the output components q_1, q_2 and q_4 correspond respectively to f_L, f_{C2} and f_{C1}. One period of the two-dimensional power-density spectrum of a sample CFA image with the Bayer CFA structure is shown in Figure 7.4. This spectrum is obtained using the method of averaging modified periodograms [2]. The different components are easily identified on this figure; compare with Figure 7.3. This spectral diagram also serves to explain the artifacts commonly seen when Bayer CFA images are demosaicked [12]. High-frequency luma patterns intrude into the chrominance bands, resulting in false colors. High-frequency chrominance information intrudes into the luma band, resulting in false luma patterns, often having a zipper-like appearance. These effects are very similar to the luma-chrominance crosstalk familiar in NTSC and PAL composite television signals [13].

7.3.3 Examples

To illustrate these concepts, three additional examples of CFA structures that have been proposed are presented. There is no implication that these are the best of the many proposed structures; rather, they have been selected because they illustrate different scenarios. These structures are: i) a hexagonal array, as used in the Super CCD proposed and manufactured by Fujifilm, that is like a Bayer pattern rotated by 45° and has much in common with the Bayer example already presented [3]; ii) a diagonal stripe pattern with $C = K = 3$ [8]; and iii) a four-color pattern with $C = K = 4$.

7.3.3.1 Hexagonal Pattern

The first example concerns a sensor where the sensor elements are placed on a hexagonal lattice. This has been referred to as the pixel interleaved array CCD (PIACCD) or as the SuperCCD [3]. The lattice and CFA structure are shown in Figure 7.5. The Voronoi unit cell has a square shape (rotated by 45°), but the PIACCD sensor elements are in fact octagonal-shaped subsets of the unit cell. Again, this detail does not affect our analysis; it simply contributes to the precise form of $h_a(\mathbf{x})$. This sampling structure is seen to be equivalent to a Bayer structure rotated by 45°, so there are $C = 3$ sensor classes and $K = 4$ elements in a period of the CFA. One period used here is outlined by the thick border in the top left of

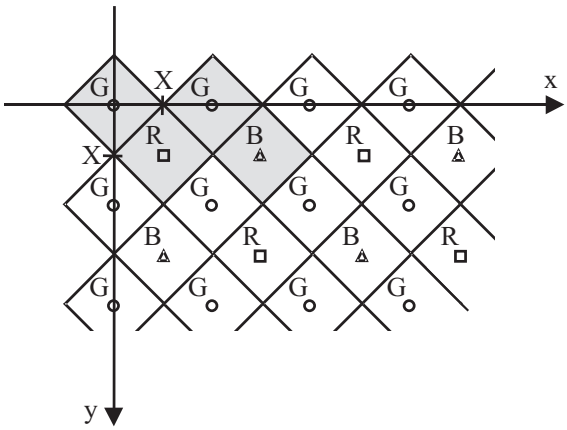


FIGURE 7.5

PIACCD CFA structure showing the constituent sampling structures Ψ_R (□), Ψ_G (○) and Ψ_B (△). The union of these three sampling structures forms the lattice Λ.

Figure 7.5. We use the distance X indicated in Figure 7.5 as the unit of length (1 px) in the following. Typically, the demosaicked signal on Λ would be upsampled to a square lattice with spacing X, but we do not consider that step here.

By inspection of Figure 7.5, we identify

$$\Lambda = \mathrm{LAT}\left(\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}\right), \qquad \Gamma = \mathrm{LAT}\left(\begin{bmatrix} 4 & 2 \\ 0 & 2 \end{bmatrix}\right), \tag{7.33}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 2 & 1 & 3 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \qquad \text{and} \qquad \mathbf{J} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7.34}$$

The corresponding reciprocal lattices can be found to be

$$\Lambda^* = \mathrm{LAT}\left(\begin{bmatrix} 1 & 0.5 \\ 0 & 0.5 \end{bmatrix}\right) \qquad \Gamma^* = \mathrm{LAT}\left(\begin{bmatrix} 0.5 & 0.25 \\ 0 & 0.25 \end{bmatrix}\right). \tag{7.35}$$

These lattices are illustrated in Figure 7.6. Then, we choose the set of coset representatives for Λ^* in Γ^* as indicated in the figure, yielding the matrix

$$\mathbf{D} = \begin{bmatrix} 0 & 0.5 & 0.25 & 0.25 \\ 0 & 0 & 0.25 & -0.25 \end{bmatrix}. \tag{7.36}$$

Substitution into Equation 7.10 gives the matrix

$$\mathbf{M} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ -1 & 2 & -1 \\ -1 & 0 & 1 \\ 1 & 0 & -1 \end{bmatrix}, \tag{7.37}$$

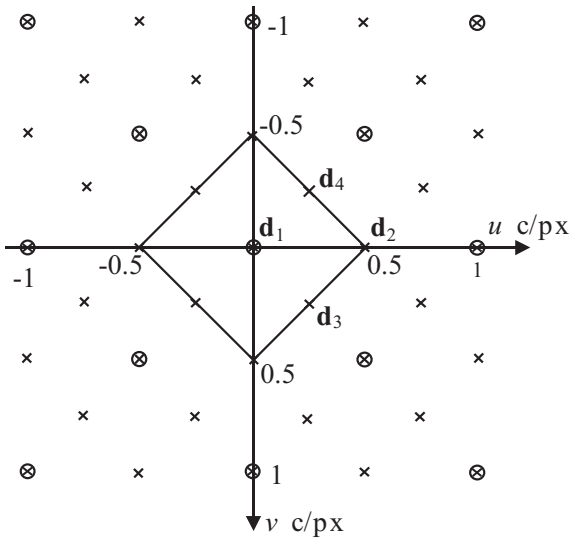


FIGURE 7.6

Reciprocal lattices Λ^* ($\circ$) and Γ^* ($\times$) and a suitable choice of representatives for the cosets of Λ^* in Γ^* for the PIACCD structure.

and the resulting transformed signals

$$q_1[\mathbf{x}] = \tfrac{1}{4} f_1[\mathbf{x}] + \tfrac{1}{2} f_2[\mathbf{x}] + \tfrac{1}{4} f_3[\mathbf{x}] \tag{7.38}$$

$$q_2[\mathbf{x}] = -\tfrac{1}{4} f_1[\mathbf{x}] + \tfrac{1}{2} f_2[\mathbf{x}] - \tfrac{1}{4} f_3[\mathbf{x}] \tag{7.39}$$

$$q_3[\mathbf{x}] = -\tfrac{1}{4} f_1[\mathbf{x}] + \tfrac{1}{4} f_3[\mathbf{x}] \tag{7.40}$$

$$q_4[\mathbf{x}] = \tfrac{1}{4} f_1[\mathbf{x}] - \tfrac{1}{4} f_3[\mathbf{x}] \tag{7.41}$$

where we note that $q_4 = -q_3$. In the frequency domain, we have

$$\begin{aligned} F_{\mathrm{CFA}}(u,v) = {} & Q_1(u,v) + Q_2(u-0.5,v) \\ & + Q_3(u-0.25,v-0.25) - Q_3(u-0.25,v+0.25). \end{aligned} \tag{7.42}$$

7.3.3.2 Diagonal Stripe Pattern

The second example is a diagonal stripe pattern that contains $C = 3$ sensor classes (RGB) and only $K = 3$ elements in a period of the CFA structure. A portion of this CFA pattern using the same structure as Reference [8] is shown in Figure 7.7. Again, with X as the unit of length, we identify

$$\Lambda = \mathrm{LAT}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = \mathbb{Z}^2, \qquad \Gamma = \mathrm{LAT}\left(\begin{bmatrix} 3 & 1 \\ 0 & 1 \end{bmatrix}\right), \quad \text{and} \tag{7.43}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix}. \tag{7.44}$$

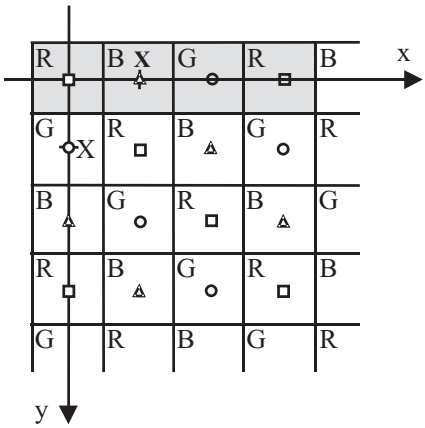


FIGURE 7.7

Stripe RGB CFA structure showing the constituent sampling structures Ψ_R (□), Ψ_G (○) and Ψ_B (△). The union of these three sampling structures forms the lattice Λ.

The matrix $\mathbf{J}$ defining the CFA structure is

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{7.45}$$

The corresponding reciprocal lattices

$$\Lambda^* = \mathrm{LAT}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right), \qquad \Gamma^* = \mathrm{LAT}\left(\begin{bmatrix} 1 & \frac{1}{3} \\ 0 & -\frac{1}{3} \end{bmatrix}\right), \tag{7.46}$$

are illustrated in Figure 7.8 along with a suitable choice of coset representatives for Λ^* in Γ^*, giving

$$\mathbf{D} = \begin{bmatrix} 0 & \frac{1}{3} & -\frac{1}{3} \\ 0 & -\frac{1}{3} & \frac{1}{3} \end{bmatrix}. \tag{7.47}$$

Substitution into Equation 7.10 gives the matrix

$$\mathbf{M} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -\frac{1}{2} + j\frac{\sqrt{3}}{2} & -\frac{1}{2} - j\frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} - j\frac{\sqrt{3}}{2} & -\frac{1}{2} + j\frac{\sqrt{3}}{2} \end{bmatrix}. \tag{7.48}$$

The three transformed signals are thus

$$q_1[\mathbf{x}] = \tfrac{1}{3} f_1[\mathbf{x}] + \tfrac{1}{3} f_2[\mathbf{x}] + \tfrac{1}{3} f_3[\mathbf{x}] \tag{7.49}$$

$$q_2[\mathbf{x}] = \tfrac{1}{3} f_1[\mathbf{x}] + \left(-\tfrac{1}{6} + j\tfrac{1}{2\sqrt{3}}\right) f_2[\mathbf{x}] + \left(-\tfrac{1}{6} - j\tfrac{1}{2\sqrt{3}}\right) f_3[\mathbf{x}] \tag{7.50}$$

$$q_3[\mathbf{x}] = \tfrac{1}{3} f_1[\mathbf{x}] + \left(-\tfrac{1}{6} - j\tfrac{1}{2\sqrt{3}}\right) f_2[\mathbf{x}] + \left(-\tfrac{1}{6} + j\tfrac{1}{2\sqrt{3}}\right) f_3[\mathbf{x}]. \tag{7.51}$$

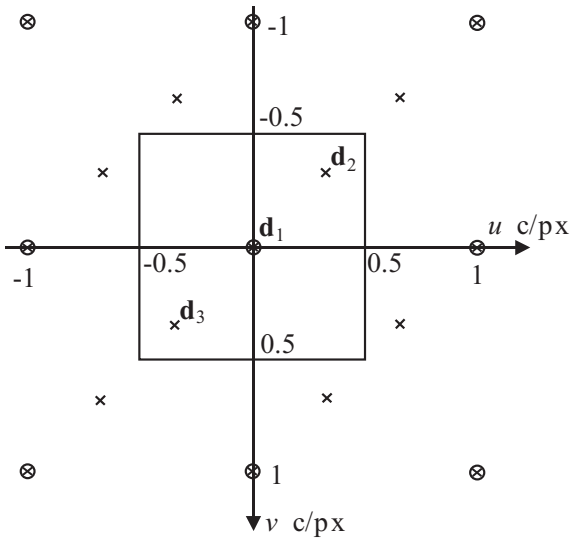


FIGURE 7.8

Reciprocal lattices Λ^* ($\circ$) and Γ^* ($\times$) and a suitable choice of representatives for the cosets of Λ^* in Γ^* for the stripe structure.

We see that $q_2[\mathbf{x}]$ and $q_3[\mathbf{x}]$ are complex and that $q_3[\mathbf{x}] = q_2^*[\mathbf{x}]$. The inverse transformation recovers the original components using the matrix $\mathbf{M}^{-1}$,

$$\mathbf{M}^{-1} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -\frac{1}{2} - j\frac{\sqrt{3}}{2} & -\frac{1}{2} + j\frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} + j\frac{\sqrt{3}}{2} & -\frac{1}{2} - j\frac{\sqrt{3}}{2} \end{bmatrix}. \tag{7.52}$$

In the frequency domain

$$F_{\mathrm{CFA}}(u,v) = Q_1(u,v) + Q_2(u-\tfrac{1}{3}, v+\tfrac{1}{3}) + Q_3(u+\tfrac{1}{3}, v-\tfrac{1}{3}). \tag{7.53}$$

Although there is no problem with this complex formulation, we can avoid the use of complex signals by expressing the modulation as a quadrature modulation of real signals. Thus, rather than considering the two complex modulated signals

$$q_2[\mathbf{x}] \exp\left(-j2\pi\left(\tfrac{x}{3} - \tfrac{y}{3}\right)\right) + q_3[\mathbf{x}] \exp\left(-j2\pi\left(-\tfrac{x}{3} + \tfrac{y}{3}\right)\right), \tag{7.54}$$

we can consider the equivalent real, quadrature modulated signals

$$q_2'[\mathbf{x}] \cos\left(2\pi\left(\tfrac{x}{3} - \tfrac{y}{3}\right)\right) + q_3'[\mathbf{x}] \sin\left(2\pi\left(\tfrac{x}{3} - \tfrac{y}{3}\right)\right) \tag{7.55}$$

where

$$q_2'[\mathbf{x}] = 2\Re\{q_2[\mathbf{x}]\} = \tfrac{2}{3} f_1[\mathbf{x}] - \tfrac{1}{3} f_2[\mathbf{x}] - \tfrac{1}{3} f_3[\mathbf{x}] \tag{7.56}$$

$$q_3'[\mathbf{x}] = 2\Im\{q_2[\mathbf{x}]\} = \tfrac{1}{\sqrt{3}} f_2[\mathbf{x}] - \tfrac{1}{\sqrt{3}} f_3[\mathbf{x}], \tag{7.57}$$

and where $\Re$ and $\Im$ extract the real and imaginary part of a complex number.

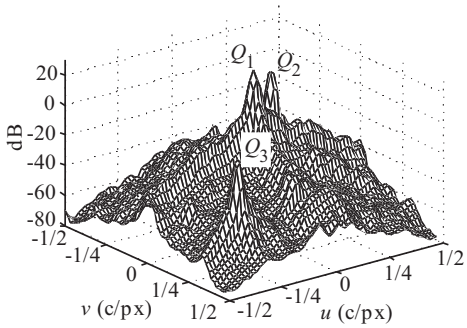


FIGURE 7.9

Two-dimensional power density spectrum estimate of a CFA image with the stripe sampling structure.

One period of the two-dimensional power density spectrum of a sample CFA image with the stripe CFA structure is shown in Figure 7.9. The different components are easily identified on this figure; compare with Figure 7.8.

7.3.3.3 Four-Color Pattern

The final example considers a four-color pattern as illustrated in Figure 7.10. A number of these have been proposed in recent years including CMYB [14], RBG1G2 [15], RGB + gray [16] and the RGB + Emerald as used in the Sony Cybershot DSC-F828 digital camera. In this case, $C = K = 4$. The lattices $\Lambda = \mathbb{Z}^2$ and $\Gamma = (2\mathbb{Z})^2$ are the same as for the Bayer structure and the same matrix $\mathbf{B}$ can be used. The matrix $\mathbf{J}$ is the 4×4 identity matrix $\mathbf{I}_4$. The reciprocal lattices $\Lambda^* = \mathbb{Z}^2$ and $\Gamma^* = (\frac{1}{2}\mathbb{Z})^2$ are again the same as for the Bayer structure and the same $\mathbf{D}$ can be used, so Figure 7.3 applies to this case.

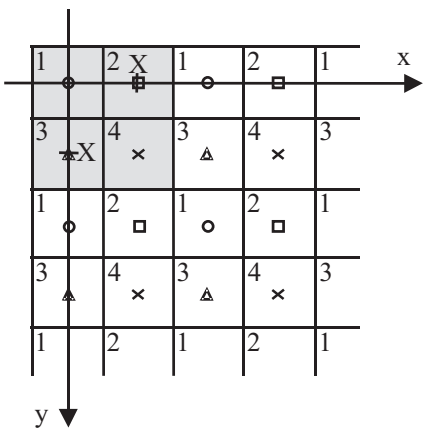


FIGURE 7.10

Four-color CFA structure showing the constituent sampling structures Ψ_1 (○), Ψ_2(□), Ψ_3 (△) and Ψ_4 (×). The union of these four sampling structures forms the lattice Λ.

Applying Equation 7.10 to the above, we obtain

$$\mathbf{M} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}, \tag{7.58}$$

the transformed signals

$$q_1[\mathbf{x}] = \tfrac{1}{4}f_1[\mathbf{x}] + \tfrac{1}{4}f_2[\mathbf{x}] + \tfrac{1}{4}f_3[\mathbf{x}] + \tfrac{1}{4}f_4[\mathbf{x}] \tag{7.59}$$

$$q_2[\mathbf{x}] = \tfrac{1}{4}f_1[\mathbf{x}] + \tfrac{1}{4}f_2[\mathbf{x}] - \tfrac{1}{4}f_3[\mathbf{x}] - \tfrac{1}{4}f_4[\mathbf{x}] \tag{7.60}$$

$$q_3[\mathbf{x}] = \tfrac{1}{4}f_1[\mathbf{x}] - \tfrac{1}{4}f_2[\mathbf{x}] + \tfrac{1}{4}f_3[\mathbf{x}] - \tfrac{1}{4}f_4[\mathbf{x}] \tag{7.61}$$

$$q_4[\mathbf{x}] = \tfrac{1}{4}f_1[\mathbf{x}] - \tfrac{1}{4}f_2[\mathbf{x}] - \tfrac{1}{4}f_3[\mathbf{x}] + \tfrac{1}{4}f_4[\mathbf{x}] \tag{7.62}$$

and in the frequency domain

$$F_{\mathrm{CFA}}(u,v) = Q_1(u,v) + Q_2(u-0.5,v) + Q_3(u,v-0.5) + Q_4(u-0.5,v-0.5). \tag{7.63}$$

We need to know what the four signals (sensor classes) are to proceed further. Note that if $c_1(\lambda) = c_4(\lambda)$, we revert to the case of the Bayer array.

7.3.4 Summary

The analysis in this section has shown that the spatial multiplexing of pixels corresponding to different sensor classes, such as red, green and blue, can be equivalently viewed as the frequency domain multiplexing of transformed components obtained as linear combinations of the original input components. These components generally consist of a luma at baseband and several chrominance components modulated at certain frequencies. The baseband component is a weighted average of the input components, where the weights are the relative sampling densities of the given components. This baseband component is similar to the luminance component of human vision, which arises in a similar fashion from the retinal mosaic of cones [12]. Since this baseband component is *not* luminance, we use the term *luma* as advocated by Poynton [9]. The chrominance components are various differences of the input components, and all are identically zero in the case where the inputs from all sensor classes are equal (a gray-scale condition for these sensor classes).

The analysis presented shows how to use lattice theory to determine these transformed components and the frequencies at which they are modulated. The interested reader can apply the analysis to the other RGB CFA patterns of [5]. The author's analysis can be found at the companion webpage for this chapter [17]. The next section shows how these components can be demultiplexed using spatial filtering and subsequently transformed to yield the desired tristimulus values at all points on the sampling lattice, as required to further process and display the image.

As seen in Section 7.3.3.2, some of the transformed chrominance components may be complex. However, this can be avoided by using the quadrature modulation representation. For any coset representative $\mathbf{d}_k$ of Λ^* in Γ^*, there are two possible situations, that correspond to real and complex chrominance components respectively. The first situation is

when $\mathbf{d}_k$ and $-\mathbf{d}_k$ belong to the same coset, i.e., $\mathbf{d}_k - (-\mathbf{d}_k) \in \Lambda^*$, or equivalently $2\mathbf{d}_k \in \Lambda^*$. This always applies to $\mathbf{d}_1 = \mathbf{0}$, but it also applies to all the $\mathbf{d}_k$ for the Bayer structure and for Sections 7.3.3.1 and 7.3.3.3. Since $\mathbf{d}_k$ and $-\mathbf{d}_k$ must give the same result in Equation 7.10, it follows that $M_{ki} = M_{ki}^*$, $i = 1,\ldots,C$, and so $q_k = q_k^*$. Thus all chrominance components for these CFA structures are real. In the second situation, $-\mathbf{d}_k \notin \mathbf{d}_k + \Lambda^*$, and so $-\mathbf{d}_k \in \mathbf{d}_l + \Lambda^*$ for some $l \neq k$. Thus $M_{li} = M_{ki}^*$ for $i = 1,\ldots,C$ and $q_l = q_k^*$. The corresponding two terms in Equation 7.11 can be written as

$$q_k[\mathbf{x}]\exp(j2\pi\mathbf{x}\cdot\mathbf{d}_k) + q_k^*[\mathbf{x}]\exp(-j2\pi\mathbf{x}\cdot\mathbf{d}_k) = q_k'\cos(2\pi\mathbf{x}\cdot\mathbf{d}_k) + q_l'\sin(2\pi\mathbf{x}\cdot\mathbf{d}_k) \quad (7.64)$$

where $q_k' = 2\Re\{q_k[\mathbf{x}]\}$ and $q_l' = 2\Im\{q_k[\mathbf{x}]\}$. Note that q_k' and q_l' are still linear combinations of the original components.

7.4 Demosaicking Based on the Frequency-Domain Representation

7.4.1 The Demosaicking Problem

The digital camera with CFA sensor is an approximation to the ideal camera, whose objective is to accurately capture spatial color patterns to be reproduced for a human observer on a color display device. The bottom part of Figure 7.2 shows a model of an ideal camera designed to produce sampled color images for human viewing. The spatio-spectral light intensity is passed through a linear shift invariant spatial camera aperture $h_t(\mathbf{x})$, with frequency response $H_t(\mathbf{u})$, that is adapted to the sampling lattice Λ and possibly the assumed viewing setup [18]. The resulting signal is passed through three spectral filters $\bar{p}_i(\lambda)$, $i = 1,2,3$, that correspond to three primaries $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ of the human visual color space, and the total power is measured to produce the samples on Λ. This yields the vector signal $\tilde{\mathbf{f}}[\mathbf{x}]$, $\mathbf{x} \in \Lambda$, with three components

$$\tilde{f}_i[\mathbf{x}] = \int_{\lambda_{\min}}^{\lambda_{\max}} \int_{\mathbb{R}^2} f(\mathbf{x}-\mathbf{s},\lambda) h_t(\mathbf{s})\, d\mathbf{s}\, \bar{p}_i(\lambda)\, d\lambda, \quad i = 1,2,3. \quad (7.65)$$

The functions $\bar{p}_i(\lambda)$ are known in colorimetry as color-matching functions, and are a property of the human visual system [19]. The three components $\tilde{f}_i[\mathbf{x}], i = 1,2,3$ are called tristimulus values with respect to the given primaries. The problem at hand is to estimate $\tilde{\mathbf{f}}[\mathbf{x}]$ for $\mathbf{x} \in \Lambda$ from the observed scalar signal $f_{\mathrm{CFA}}[\mathbf{x}]$.

This is an ill-posed inverse problem with at least three separate aspects: (i) Only one component is measured at each spatial location and two others must be estimated. This operation is often called demosaicking or color-plane interpolation. (ii) The actual aperture $h_a(\mathbf{x})$ is different from the ideal one $h_t(\mathbf{x})$ and may introduce excessive resolution loss; compensating for this is image restoration or aperture correction. (iii) The $\bar{p}_i(\lambda)$ cannot be expressed as a linear combination of the actual recording filters $c_i(\lambda)$, thus introducing color errors. In this case, which is the normal situation, the camera filters are said to be non-colorimetric. In this chapter, we concentrate on the demosaicking problem. We do not consider aperture correction (which must also account for noise in the capture process), and standard solutions for the color error problem are used.

Consider for now only the color aspect of the problem. In the ideal situation, the spectral responses of the color filters for the C classes span a space that contains the color-matching functions:

$$\bar{p}_i(\lambda) \in \text{span}(\{c_j(\lambda), j = 1,\ldots,C\}), \quad i = 1,2,3. \tag{7.66}$$

In this case, we can express these functions as linear combinations of the $c_j(\lambda)$,

$$\bar{p}_i(\lambda) = \sum_{j=1}^{C} a_{ij} c_j(\lambda), \quad i = 1,2,3, \tag{7.67}$$

where the a_{ij} are unique if the $c_j(\lambda)$ are linearly independent, which is a reasonable assumption in practice. Then, the tristimulus values for a color with spectral distribution $f(\lambda)$ are given by

$$\tilde{f}_i = \int_{\lambda_{\min}}^{\lambda_{\max}} f(\lambda)\bar{p}_i(\lambda)\,d\lambda = \sum_{j=1}^{C} a_{ij} \int_{\lambda_{\min}}^{\lambda_{\max}} f(\lambda)c_j(\lambda)\,d\lambda, \tag{7.68}$$

so that the desired tristimulus values can be obtained from the values measured with the given color filters as

$$\tilde{\mathbf{f}} = \mathbf{A}\mathbf{f} \tag{7.69}$$

where $\mathbf{A} = [a_{ij}]$ is the $3 \times C$ matrix implied by (7.67) and $[\mathbf{f}]_i = \int f(\lambda)c_i(\lambda)\,d\lambda$. The primaries for the desired signal could be the CIEXYZ primaries for a device-independent representation, or some standard RGB space such as the ITU Recommendation 709 RGB primaries [9], also used in the sRGB representation.

However, in the practical situation,

$$\bar{p}_i(\lambda) \notin \text{span}(\{c_j(\lambda), j = 1,\ldots,C\}), \quad i = 1,2,3, \tag{7.70}$$

so that color errors are inevitable. In particular, two different but metamerically equivalent color spectra with equal tristimulus values will in general give different measured values, while visually different colors can give the same measured values. Essentially the human and camera visual systems are different. In this case, a transformation mapping $\mathbf{f}$ to $\tilde{\mathbf{f}}$ is required, that should minimize the expected color error according to an appropriate color distance metric over some suitable ensemble of spectral densities $f(\lambda)$ [20]. Although the best such transformation is not necessarily linear, it has been found that linear transformations can give excellent results. See Reference [21] for a review of standard techniques to solve the problem. A typical one is to project the $\bar{p}_i(\lambda)$ onto $\text{span}(\{c_j(\lambda), j = 1,\ldots,C\}$ and to use the projected color-matching functions to compute the approximate tristimulus values. Thus, we will still assume that the desired tristimulus values are obtained from the measured values using Equation 7.69 and that this is the desired signal we are trying to measure. The resulting colorimetric errors are not considered further in this chapter.

7.4.2 Algorithms Derived from the Frequency-Domain Representation

The premise of the demosaicking algorithms based on the frequency-domain representation is to extract the luma and modulated chrominance components from the CFA signal using spatial filters, and then to transform the luma and demodulated chrominance components to the desired estimated tristimulus values using the appropriate linear transformation.

The specific structure of the CFA signal in the frequency domain should be exploited. For example, with the Bayer structure, one component is modulated at two different frequencies, and either can be used to reconstruct the signal. However, if locally one suffers from crosstalk, the other is often relatively free of crosstalk. By adaptively selecting which of the two candidates should be used, superior results can be obtained [11], [22].

The basic algorithm is derived from Equation 7.15. The CFA signal is passed through a series of bandpass filters to extract the modulated chrominance components, which are demodulated to yield the estimated chrominance components. These are then linearly transformed to give the required tristimulus values. Let $H_k(\mathbf{u})$ be a two-dimensional linear shift-invariant bandpass filter with center frequency $\mathbf{d}_k$. The shape of the passband would depend on the expected support of the spectrum of the corresponding chrominance signal. Then, $\hat{r}_k = f_{\text{CFA}} * h_k$, and the resulting signal is demodulated to baseband to obtain $\hat{q}_k[\mathbf{x}] = \hat{r}_k[\mathbf{x}]\exp(-j2\pi\mathbf{x}\cdot\mathbf{d}_k)$. The baseband luma component $q_1[\mathbf{x}]$ at frequency $\mathbf{d}_1 = \mathbf{0}$ does not need to be demodulated, and can be obtained by subtracting the estimated modulated chrominance components from the CFA signal

$$\hat{q}_1[\mathbf{x}] = f_{\text{CFA}}[\mathbf{x}] - \sum_{k=2}^{K} \hat{r}_k[\mathbf{x}]. \tag{7.71}$$

When $C < K$, the transformed components q_k are linearly dependent (e.g., $q_3[\mathbf{x}] = -q_2[\mathbf{x}]$ for the Bayer CFA). Since the estimated transformed components obtained as above will not in general satisfy this constraint (e.g., $\hat{q}_3[\mathbf{x}] \neq -\hat{q}_2[\mathbf{x}]$), the constraint needs to be imposed, either explicitly or implicitly. This step is often the key to a successful result.

The approach is illustrated for the Bayer CFA structure of Figure 7.1. The global spectrum shows substantial overlap between both the modulated $Q_2(u-0.5,v)$ and $Q_2(u,v-0.5)$, and the luma component at baseband, as can be seen from Figure 7.3. However, locally, the overlap tends to be only with one of these components and the luma, depending on the local image content. This is illustrated schematically in Figure 7.11 which shows the hypothesized local spectrum for two different scenarios. By identifying which of the two versions locally has less crosstalk and using that one for reconstruction, a much better

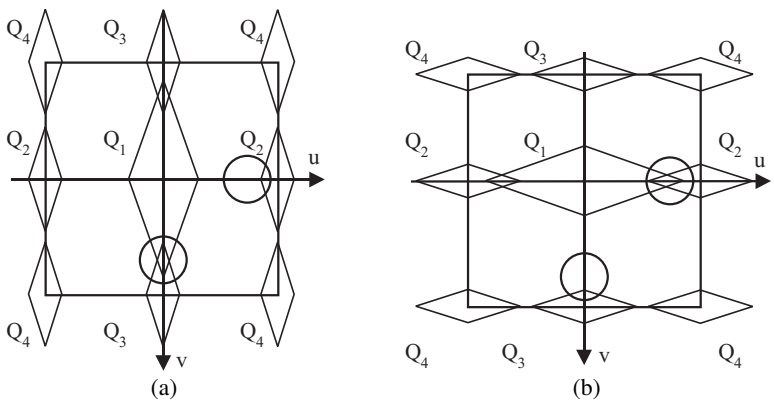


FIGURE 7.11

Local spectrum scenarios schematically illustrated for Bayer CFA pattern: (a) scenario with Q_2 being better estimate, and (b) scenario with Q_3 being better estimate.

(a) (b)

FIGURE 7.12 (See color insert.)

Reconstruction of lighthouse image (a) using only $Q_2(u-0.5,v)$ and (b) using only $Q_2(u,v-0.5)$.

result can be obtained than by using both of them equally. In fact, Figure 7.12 shows that if the lighthouse image is reconstructed using only $Q_2(u-0.5,v)$ (Figure 7.12a) and only $Q_2(u,v-0.5)$ (Figure 7.12b), all areas of the image are well reconstructed in one or the other of these two images; we just need a genie to identify which one is better for each pixel. Note in particular the house at the left where the first scenario applies, and the picket fence where the second scenario applies.

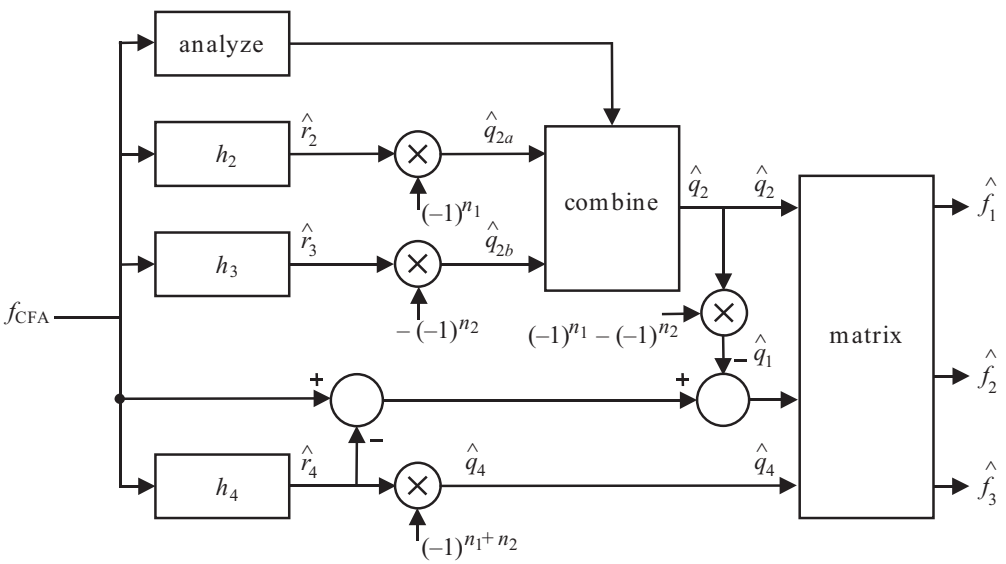


FIGURE 7.13

Block diagram of adaptive demosaicking algorithm for the Bayer CFA structure.

As described in Reference [11], we can make this decision by measuring the average local energies e_X and e_Y in the vicinity of the circular regions in Figure 7.11 centered at frequencies $(u_m, 0)$ and $(0, v_m)$. The direction (horizontal or vertical) with the lower energy is assumed to suffer less from crosstalk and is given more weight in the reconstruction. The proposed algorithm from Reference [11] with notation adapted to that of this chapter is summarized below (Algorithm 7.1) and illustrated in Figure 7.13. Note that here, with $\mathbf{x} = [n_1\ n_2]^T$, we have $\exp(\pm j2\pi\mathbf{x}\cdot\mathbf{d}_2) = (-1)^{n_1}$, $\exp(\pm j2\pi\mathbf{x}\cdot\mathbf{d}_3) = (-1)^{n_2}$, and $\exp(\pm j2\pi\mathbf{x}\cdot\mathbf{d}_4) = (-1)^{n_1+n_2}$.

ALGORITHM 7.1 Adaptive demosaicking algorithm for the Bayer CFA.

1. Filter f_{CFA} with a bandpass filter h_4 centered at frequency $(0.5, 0.5)$ to extract $\hat{r}_4 = f_{\text{CFA}} * h_4$, and shift it to baseband to estimate $\hat{q}_4[n_1, n_2] = \hat{r}_4[n_1, n_2] \cdot (-1)^{n_1+n_2}$.
2. Filter f_{CFA} with h_2 to get $\hat{r}_2 = f_{\text{CFA}} * h_2$ and demodulate to baseband, $\hat{q}_{2a}[n_1, n_2] = \hat{r}_2[n_1, n_2] \cdot (-1)^{n_1}$. Similarly, $\hat{r}_3 = f_{\text{CFA}} * h_3$ and $\hat{q}_{2b}[n_1, n_2] = -\hat{r}_3[n_1, n_2](-1)^{n_2}$ (using $q_2 = -q_3$).
3. The local average energies e_X and e_Y are estimated using modulated Gaussian filters with standard deviations of r_{G1} and r_{G2} px along major and minor axes, centered at frequencies $(\pm u_m, 0.0)$ and $(0.0, \pm v_m)$ c/px respectively. The filter at $(0.0, \pm v_m)$ is the transpose of the filter at $(\pm u_m, 0.0)$. This is followed by smoothing of the squared output with a 5×5 moving average filter.
4. The final estimate of q_2 is obtained as $\hat{q}_2[n_1, n_2] = w[n_1, n_2] \cdot \hat{q}_{2a}[n_1, n_2] + (1 - w[n_1, n_2])\hat{q}_{2b}[n_1, n_2]$ using the weighting coefficient $w = e_Y/(e_X + e_Y)$.
5. Estimate the luma by $\hat{q}_1[n_1, n_2] = f_{\text{CFA}}[n_1, n_2] - \hat{r}_4[n_1, n_2] - \hat{q}_2[n_1, n_2]((-1)^{n_1} - (-1)^{n_2})$.
6. Estimate the RGB components $\hat{f}_1$, $\hat{f}_2$, $\hat{f}_3$ from $\hat{q}_1$, $\hat{q}_2$ and $\hat{q}_4$ using Equations 7.29 to 7.31, and Equation 7.69 if needed.

A similar approach can be used for the PIACCD structure (Section 7.3.3.1), since in this case there are two separate copies of the component q_3. However, for the other two examples of the stripe pattern (Section 7.3.3.2) and the four-color patterns (Section 7.3.3.3), there is no duplication of components and all components are required to reconstruct the image. Thus, the above approach is not directly applicable. The basic algorithm can be used for the stripe pattern, while other approaches can be considered for the four-color patterns.

We consider the stripe pattern of Section 7.3.3.2 in more detail to illustrate the case of complex modulation. Using complex processing, a single complex filter h_2 is required to extract $\hat{r}_2$. The algorithm is straightforward (see Algorithm 7.2 and Algorithm 7.3).

ALGORITHM 7.2 Complex demosaicking algorithm for stripe CFA pattern.

1. Filter f_{CFA} with a complex bandpass filter h_2 centered at frequency $(\frac{1}{3}, -\frac{1}{3})$ to extract $\hat{r}_2 = f_{\text{CFA}} * h2$, and shift it to baseband to estimate $\hat{q}_2[n_1, n_2] = \hat{r}_2[n_1, n_2] \cdot \exp\left(-j2\pi\left(\frac{n_1}{3} - \frac{n_2}{3}\right)\right)$.
2. Estimate the luma component $\hat{q}_1 = f_{\text{CFA}} - \hat{r}_2 - \hat{r}_2^* = f_{\text{CFA}} - 2\Re\{\hat{r}_2\}$.
3. Estimate $\hat{q}_3 = \hat{q}_2^*$.
4. Estimate the RGB components, $\hat{f}_1$, $\hat{f}_2$, $\hat{f}_3$ from $\hat{q}_1$, $\hat{q}_2$, $\hat{q}_3$ using the inverse matrix from Equation 7.52, and Equation 7.69 if needed.

ALGORITHM 7.3 Real version of the demosaicking algorithm for the stripe CFA pattern.

$$
\begin{aligned}
&h_2 = h_{2R} + jh_{2I} \\
&c_2[n_1, n_2] = \cos\left(2\pi\left(\tfrac{n_1}{3} - \tfrac{n_2}{3}\right)\right) \\
&s_2[n_1, n_2] = \sin\left(2\pi\left(\tfrac{n_1}{3} - \tfrac{n_2}{3}\right)\right) \\
&\hat{r}_{2R} = f_{\text{CFA}} * h_{2R} \\
&\hat{r}_{2I} = f_{\text{CFA}} * h_{2I} \\
&\hat{q}'_1 = f_{\text{CFA}} - 2\hat{r}_{2R} \\
&\hat{q}'_2 = \hat{r}_{2R} \cdot c_2 + \hat{r}_{2I} \cdot s_2 \\
&\hat{q}'_3 = \hat{r}_{2R} \cdot s_2 - \hat{r}_{2I} \cdot c_2 \\
&\hat{f}_1 = \hat{q}'_1 + 2\hat{q}'_2 \\
&\hat{f}_2 = \hat{q}'_1 - \hat{q}'_2 - \sqrt{3}\hat{q}'_3 \\
&\hat{f}_3 = \hat{q}'_1 - \hat{q}'_2 + \sqrt{3}\hat{q}'_3
\end{aligned}
$$

7.5 Filter Design for CFA Signal Demultiplexing

As presented in the previous section, the demosaicking methods based on the frequency-domain representation require the use of two-dimensional bandpass filters centered at the frequencies $\mathbf{d}_k$, $k = 2, \ldots, K$. The design method for these filters is a crucial step for the successful implementation of these demosaicking algorithms. In Reference [12], the use of Gaussian filters has been proposed. Essentially, baseband Gaussian lowpass filters are first generated using two parameters, the horizontal and vertical standard deviation. These lowpass filters are then converted to bandpass filters by modulating the unit-sample response with a complex sinusoid of the given frequency. Thus two parameters, and the support of the filter, need to be selected. In Reference [11], the use of frequency selective filters obtained with the window design method was presented. This was essentially a proof of concept and used 21×21 filters. The results with that method gave better performance than other methods known at that time for the Bayer structure when used with the adaptive algorithm described in the previous section. In Reference [22] it was shown that a

least-squares design method could give comparable or better results to the window method described in Reference [11], but with much lower complexity. Thus, only the least-squares methodology is presented in this chapter.

According to Equation 7.15, the observed CFA signal is the sum of K constituent modulated signals, each occupying a distinct frequency band, that we can estimate by a linear filter. Let r_Y represent any one of these signals, that we estimate by filtering f_{CFA} with the linear, shift-invariant filter having unit-sample response h_Y: $\hat{r}_Y = f_{\text{CFA}} * h_Y$. If we assume that the difference between $\hat{r}_Y$ and r_Y can be modelled as a stationary random field, then we could choose h_Y to minimize the expected squared error: $h_Y = \arg\min_h E[(r_Y[n_1,n_2]-(f_{\text{CFA}}*h)[n_1,n_2])^2]$. In practice, since we don't have a good random field model of the estimation error, we can minimize the actual error over a training set of typical color images: this becomes a least-squares problem. Note that other approaches to demosaicking using linear least-squares, Wiener filtering or similar techniques have been presented in References [6], [7], [23], and [24].

Assume that we have a training set of color images for which all the desired sensor-class components are available on the full lattice Λ. For each image in this training set, we can form both the CFA signal and the desired signal r_Y at full resolution. Suppose that we partition the training set into P sub-images, where the i^{th} sub-image is defined on the spatial block $\mathscr{W}^{(i)} \subset \Lambda$. Assume that the desired filter h_Y is a finite impulse response (FIR) 2D filter with region of support $\mathscr{S} \subset \Lambda$. Then, the least-squares filter can be obtained as solution to

$$h_Y = \arg\min_h \sum_{i=1}^{P} \sum_{(n_1,n_2)\in\mathscr{W}^{(i)}} \left(r_Y^{(i)}[n_1,n_2] - \sum_{(k_1,k_2)\in\mathscr{S}} h[k_1,k_2] f_{\text{CFA}}^{(i)}[n_1-k_1,n_2-k_2] \right)^2. \quad (7.72)$$

This expression can easily be cast in matrix form to simplify the solution with standard matrix packages such as MATLAB®. Let $N_B = |\mathscr{S}|$ be the number of filter coefficients to be determined, and $N_W = |\mathscr{W}^{(i)}|$ be the number of samples in the sub-images (the same for all i). We form an $N_B \times 1$ column vector $\mathbf{h}$ from the filter coefficients by scanning the region $\mathscr{S}$ in some fixed order, say column by column from left to right. Similarly, we form an $N_W \times 1$ column vector $\mathbf{r}_Y^{(i)}$ from $r_Y^{(i)}[n_1,n_2]$ by scanning the pixels of $\mathscr{W}^{(i)}$ in a fixed order. Finally, we form an $N_W \times N_B$ matrix $\mathbf{Z}^{(i)}$ from the elements of $f_{\text{CFA}}^{(i)}$ as follows: each column of $\mathbf{Z}^{(i)}$ corresponds to an element $(k_1,k_2) \in \mathscr{S}$ scanned in the same order as used to form $\mathbf{h}$; this column is obtained by scanning the elements of $f_{\text{CFA}}^{(i)}[n_1-k_1,n_2-k_2]$ for $(n_1,n_2) \in \mathscr{W}^{(i)}$ in the same order used to form $\mathbf{r}_Y^{(i)}$. In this way, equation (7.72) can be written in matrix form as

$$\mathbf{h}_Y = \arg\min_{\mathbf{h}} \sum_{i=1}^{P} \|\mathbf{Z}^{(i)}\mathbf{h} - \mathbf{r}_Y^{(i)}\|^2. \quad (7.73)$$

This is a standard least-squares problem with solution [10]

$$\mathbf{h}_Y = \left[\sum_{i=1}^{P} \mathbf{Z}^{(i)^H} \mathbf{Z}^{(i)} \right]^{-1} \left[\sum_{i=1}^{P} \mathbf{Z}^{(i)^H} \mathbf{r}_Y^{(i)} \right]. \quad (7.74)$$

The result $\mathbf{h}_Y$ is then reshaped to give the desired filter $h_Y[\mathbf{x}]$.

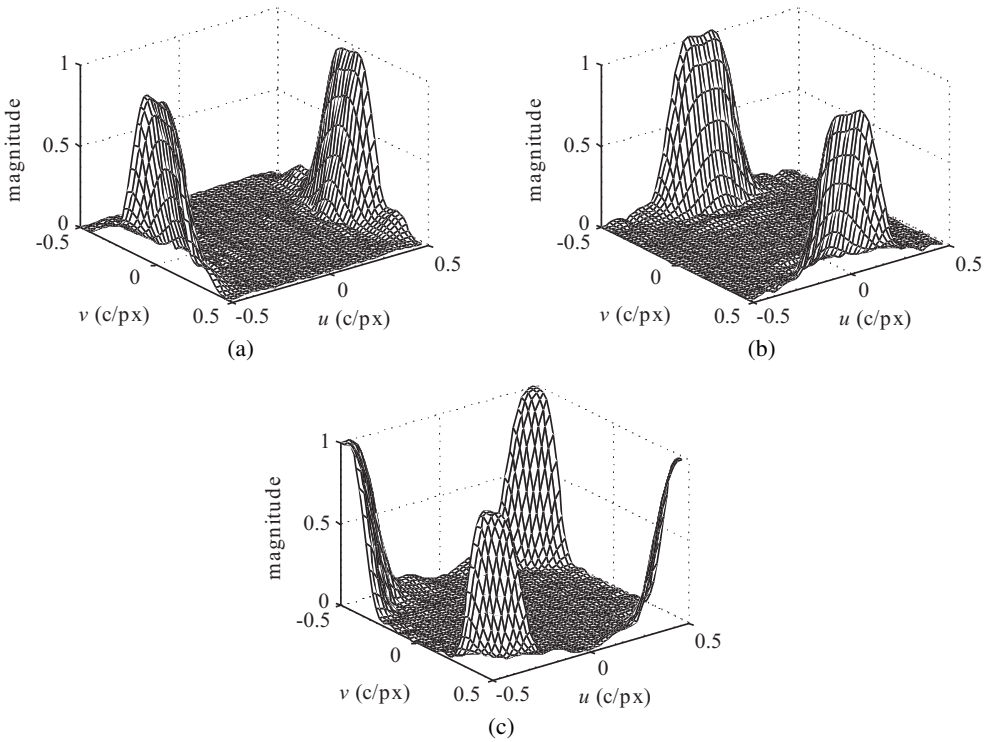


FIGURE 7.14

Frequency response of 11×11 filters designed by the adaptive least-squares method: (a) h_2, (b) h_3, and (c) h_4.

For the Bayer structure, this approach can be used to determine the three filters h_2, h_3 and h_4 required in the adaptive frequency domain algorithm. Although it is probably quite adequate for determining h_4, it does not account for the adaptive nature of the algorithm used to estimate q_2. The following describes a least-squares algorithm to simultaneously determine h_2 and h_3 to minimize the squared error in the estimation of q_2 with the adaptive algorithm of Section 7.4.

Referring to the algorithm description, the estimate of $q_2[n_1,n_2]$ is obtained by

$$\hat{q}_2[n_1,n_2] = w[n_1,n_2](-1)^{n_1} \times \sum_{(k_1,k_2)\in\mathscr{S}} h_2[k_1,k_2] f_{\mathrm{CFA}}[n_1-k_1,n_2-k_2] - (1-w[n_1,n_2])(-1)^{n_2} \times \sum_{(k_1,k_2)\in\mathscr{S}} h_3[k_1,k_2] f_{\mathrm{CFA}}[n_1-k_1,n_2-k_2], \quad (7.75)$$

and we can choose h_2 and h_3 jointly to minimize the total squared error between q_2 and $\hat{q}_2$ over the training set. Again, we cast this squared error in matrix form. Let $\mathbf{h}_{23}$ be the $2N_B \times 1$ column vector obtained by stacking $\mathbf{h}_2$ on top of $\mathbf{h}_3$. The column vector $\mathbf{q}_2^{(i)}$ is obtained by scanning the elements of $q_2[n_1,n_2]$ over $\mathscr{W}^{(i)}$ in the same order as described above. Finally, we form a $N_W \times 2N_B$ matrix $\mathbf{W}^{(i)}$ as follows: the first N_B columns are formed by reshaping $w^{(i)}[n_1,n_2](-1)^{n_1} f_{\mathrm{CFA}}^{(i)}[n_1-k_1,n_2-k_2]$ for each $(k_1,k_2) \in \mathscr{S}$ while the second N_B columns are formed by reshaping $-(1-w^{(i)}[n_1,n_2])(-1)^{n_2} f_{\mathrm{CFA}}^{(i)}[n_1-k_1,n_2-k_2]$ in the

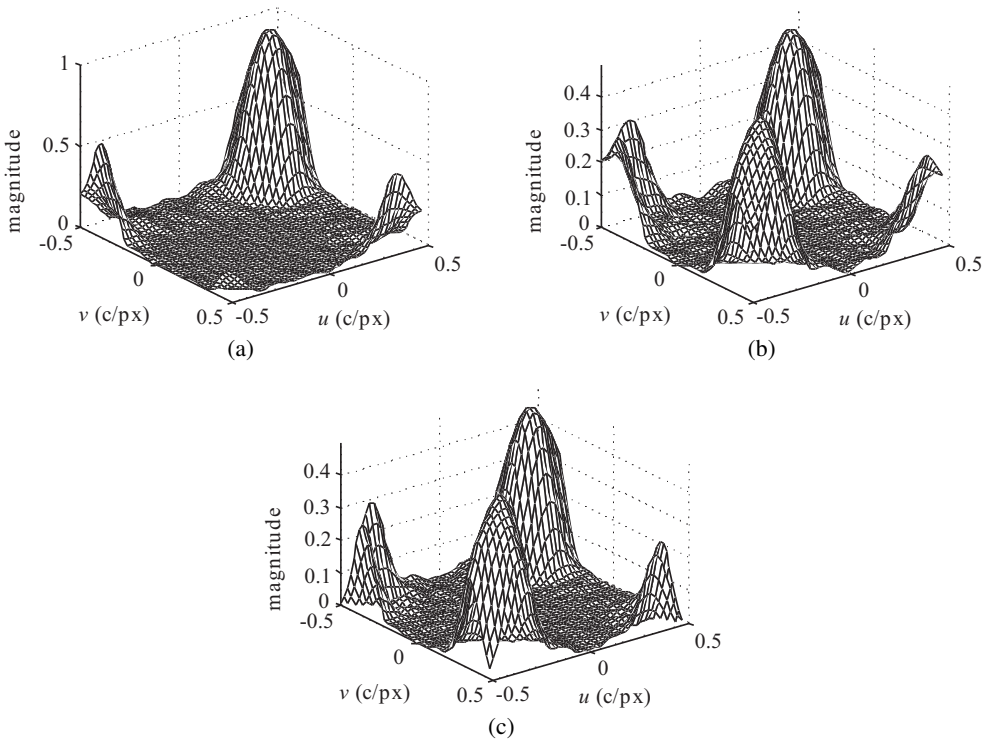


FIGURE 7.15

Frequency response of 11×11 filters designed by the least-squares method for the stripe pattern: (a) h_2, (b) h_{2R}, and (c) h_{2I}.

same order. Once again, this leads to a least squares problem of the form

$$\mathbf{h}_{23} = \arg\min_{\mathbf{h}} \sum_{i=1}^{P} \|\mathbf{W}^{(i)}\mathbf{h} - \mathbf{q}_2^{(i)}\|^2 \tag{7.76}$$

$$= \left[\sum_{i=1}^{P} \mathbf{W}^{(i)H}\mathbf{W}^{(i)}\right]^{-1} \left[\sum_{i=1}^{P} \mathbf{W}^{(i)H}\mathbf{q}_2^{(i)}\right]. \tag{7.77}$$

Finally, $\mathbf{h_2}$ and $\mathbf{h_3}$ are extracted from $\mathbf{h}_{23}$ and reshaped to give the optimized filters $h_2[\mathbf{x}]$ and $h_3[\mathbf{x}]$.

Figure 7.14 shows the frequency response of the filters obtained by applying these procedures, using Equation 7.74 to obtain $h_4[\mathbf{x}]$ and Equation 7.77 to obtain $h_2[\mathbf{x}]$ and $h_3[\mathbf{x}]$. The training set consists of images 1-12 of the standard Kodak database commonly used to test demosaicking algorithms [25], where all filters have a support of 11×11. As reported in Reference [22], these filters give equivalent or better mean-squared error (MSE) than the 21×21 window-designed filters of Reference [11], which in turn gave equal or better MSE results than all the techniques compared in the review paper [25]. The results of demosaicking the twenty-four Kodak images with these 21×21 filters used in the adaptive algorithm can be seen on the website associated with Reference [11]. The results with the least-squares filters are visually very similar.

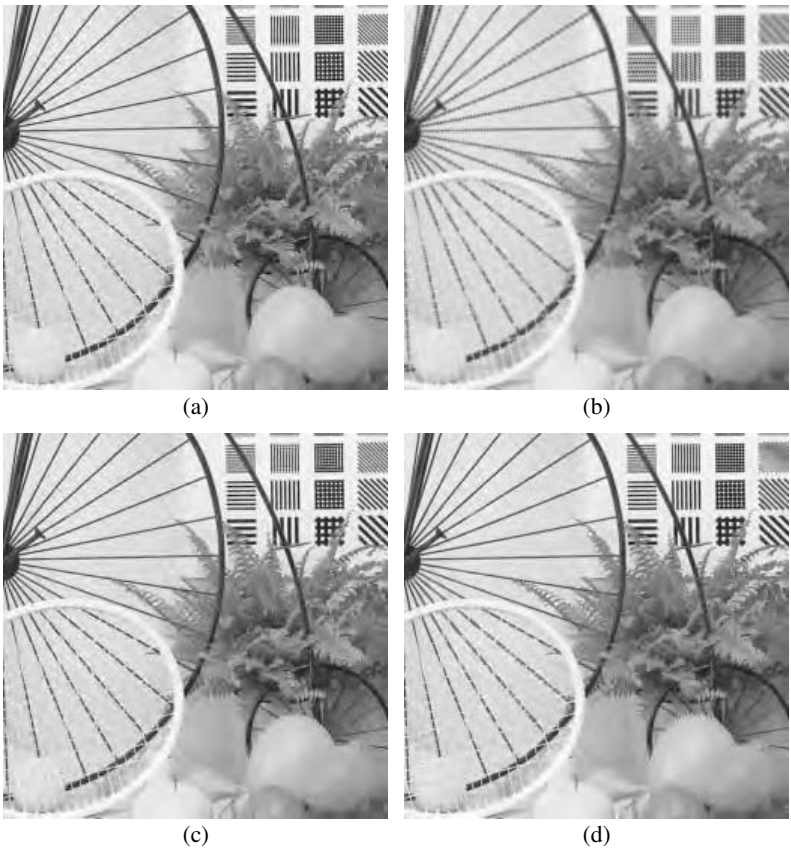


FIGURE 7.16 (See color insert.)

Portion of JPEG2000 test image bike, downsampled by four in each direction: (a) original color image, (b) image reconstructed from Bayer CFA using bilinear interpolation, (c) image reconstructed from Bayer CFA using the adaptive frequency demultiplexing algorithm, and (d) image reconstructed from the stripe CFA using the least-squares frequency demultiplexing.

An extensive study, to be reported elsewhere, examined the effect of the various parameters and filter support regions and identified good choices for these parameters. Specifically, in step 3 of the algorithm, we recommend $u_m = v_m = 0.375$ c/px, $r_{G1} = 3.0$, $r_{G2} = 1.0$, and a maximum support for the Gaussian filters of 11×3. Filters h_2, h_3 and h_4 have maximum support of 11×11, as use of larger support gave no improvement in performance.

The least-squares design of Equation 7.74 can be applied directly to complex signals without modification. This is illustrated for the stripe CFA pattern. Applying this design algorithm to the Kodak dataset yields the complex filter h_2 whose magnitude response is illustrated in Figure 7.15a. The constituent real and imaginary parts h_{2R} and h_{2I} are shown in Figure 7.15b and Figure 7.15c. While this pattern and filters give reasonable results, they are in general inferior to the results obtained with the best adaptive algorithm used with the Bayer pattern. However, certain areas are improved, such as the problematic picket fence in the lighthouse image. This is to be expected from inspection of the spectral plots for the two CFA patterns.

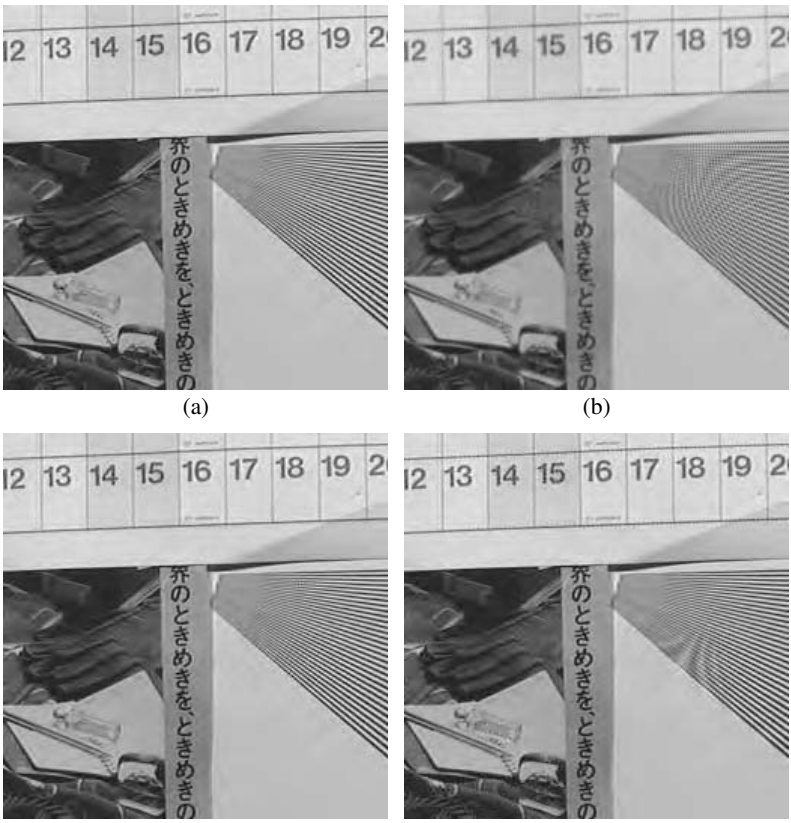


FIGURE 7.17 (See color insert.)

Portion of Spincalendar: (a) original color image, (b) image reconstructed from Bayer CFA using bilinear interpolation, (c) image reconstructed from Bayer CFA using the adaptive frequency demultiplexing algorithm, and (d) image reconstructed from the stripe CFA using the least-squares frequency demultiplexing.

The frequency-domain approach can be illustrated with critical areas taken from two standard images: a portion of the JPEG 2000 test image Bike (downsampled by four in each direction), and a portion of the Spincalendar HDTV test sequence. Figure 7.16 and Figure 7.17 each show the original image, the image reconstructed from the Bayer CFA using bilinear interpolation, the image reconstructed from the Bayer CFA using the adaptive least-squares algorithm, and the image reconstructed from the stripe pattern using the least-squares algorithm. The bilinear reconstruction clearly shows the artifacts due to luma-chrominance crosstalk for the Bayer pattern in both images. The test-pattern portion of the Bike image has very high horizontal frequencies, right up to the Nyquist frequency. An excellent result is obtained with the adaptive least-squares frequency demultiplexing algorithm, but a small amount of residual cross color remains. For the stripe CFA, there is much less crosstalk in the horizontal-frequency components, due to the location of the chrominance components, and an even better result is obtained for these portions of the Bike image. However, other portions with diagonal structures do not fare as well. For the Spincalendar image, there are strong diagonal frequencies, and the Bayer CFA with the

adaptive least-squares frequency demultiplexing algorithm gives a much better result than the stripe pattern. Overall, the Bayer pattern is better when evaluated over all 24 Kodak images. Other demosaicking results can be found at the companion webpage [17] for this chapter.

7.6 Concluding Remarks

This chapter has presented a mathematical framework to analyze periodic CFA patterns (which includes most that have been proposed) using a frequency-domain approach. This framework serves to explain the typical artifacts observed in demosaicked images with these patterns and to inspire new demosaicking methods with good performance and moderate complexity. The analysis has been illustrated with the common Bayer pattern and with three other CFA patterns that bring out several aspects of the theory. Many more structures have been proposed and can be analyzed with these methods. Detailed numerical and visual results for these methods have not been presented here as that was not the goal; many such results can be found elsewhere. The adaptive method for the Bayer CFA described in this chapter is currently highly competitive with the state of the art with respect to demosaicked image quality and computational complexity. However, the work on this topic is not concluded. Among other directions, the extension of the adaptive algorithm to the case of four sensor classes should be pursued, as well as integrating the method with up-sampling and super-resolution techniques.

Acknowledgments

The author thanks Markus Beermann, Stéphane Coulombe and Brian Leung for their helpful comments and corrections on earlier drafts of this chapter. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Appendix: Lattices and Two-Dimensional Signals on Lattices

Lattices have been widely used to describe sampled multidimensional signals with non-rectangular sampling structures. For the purposes of this chapter, we are concerned with discrete-space two-dimensional (2D) still images. This appendix summarizes the key concepts and notations used in this chapter. Detailed expositions and illustrations can be found in References [1] and [2]. The discussion is limited to the 2D case, since that is all we require here.

1. A *lattice* Λ in two-dimensions is the set of all linear combinations, with integer coefficients, of two linearly independent vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ in $\mathbb{R}^2$,

$$\Lambda = \{n_1\mathbf{v}_1 + n_2\mathbf{v}_2 \mid n_1, n_2 \in \mathbb{Z}\}. \tag{7.78}$$

   The basis vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are expressed as 2×1 column matrices, and thus so are the elements of Λ.

2. The 2×2 matrix $\mathbf{V} = [\mathbf{v}_1 \mid \mathbf{v}_2]$ is referred to as a *sampling matrix* for Λ. Then, we write

$$\Lambda = \mathrm{LAT}(\mathbf{V}) = \{\mathbf{V}\mathbf{n} \mid \mathbf{n} \in \mathbb{Z}^2\}. \tag{7.79}$$

   The sampling matrix for a given lattice is not unique; $\mathrm{LAT}(\mathbf{V}_1) = \mathrm{LAT}(\mathbf{V}_2)$ if and only if $\mathbf{E} = \mathbf{V}_1^{-1}\mathbf{V}_2$ is unimodular, i.e., an integer matrix such that $|\det\mathbf{E}| = 1$.

3. A *unit cell* of a lattice Λ is a set $\mathscr{P} \subset \mathbb{R}^2$ such that copies of $\mathscr{P}$ centered on each lattice point tile all of $\mathbb{R}^2$ without overlap. The unit cell is not unique. The area of any unit cell is $d(\Lambda) = |\det\mathbf{V}|$ for any sampling matrix $\mathbf{V}$. The *Voronoi* cell is a unit cell in which no point is closer to any non-zero element of Λ than to the origin $\mathbf{0}$.

4. The set $\Lambda^* = \{\mathbf{r} \mid \mathbf{r}\cdot\mathbf{x} \in \mathbb{Z} \text{ for all } \mathbf{x} \in \Lambda\}$ is a lattice known as the *reciprocal lattice*. If $\Lambda = \mathrm{LAT}(\mathbf{V})$, then $\Lambda^* = \mathrm{LAT}(\mathbf{V}^{-T})$ where $\mathbf{V}^{-T}$ denotes $(\mathbf{V}^T)^{-1}$, T denotes matrix transpose and $\mathbf{r}\cdot\mathbf{x}$ denotes the matrix product $\mathbf{r}^T\mathbf{x}$. $d(\Lambda^*) = 1/d(\Lambda)$.

5. The set Γ is a *sublattice* of Λ if both Λ and Γ are lattices, and every point of Γ belongs to Λ. We write $\Gamma \subset \Lambda$. $\Gamma = \mathrm{LAT}(\mathbf{V}_\Gamma)$ is a sublattice of $\Lambda = \mathrm{LAT}(\mathbf{V}_\Lambda)$ if and only if $\mathbf{M} = (\mathbf{V}_\Lambda)^{-1}\mathbf{V}_\Gamma$ is an integer matrix.

6. If $\Gamma \subset \Lambda$, then $\Lambda^* \subset \Gamma^*$.

7. If $\Gamma \subset \Lambda$, so that $\mathbf{V}_\Gamma = \mathbf{V}_\Lambda\mathbf{M}$ for an integer matrix $\mathbf{M}$, then $d(\Gamma) = |\det\mathbf{M}|d(\Lambda)$ where $K = |\det\mathbf{M}|$ is an integer. $K = d(\Gamma)/d(\Lambda)$ is called the index of Γ in Λ, denoted $(\Lambda : \Gamma)$.

8. If $\Gamma \subset \Lambda$, the set

$$\mathbf{c} + \Gamma = \{\mathbf{c} + \mathbf{x} \mid \mathbf{x} \in \Gamma\} \tag{7.80}$$

   for any $\mathbf{c} \in \Lambda$ is called a *coset* of Γ in Λ. Two cosets are either identical or disjoint: $\mathbf{c} + \Gamma = \mathbf{d} + \Gamma$ if $\mathbf{c} - \mathbf{d} \in \Gamma$; otherwise $(\mathbf{c} + \Gamma) \bigcap (\mathbf{d} + \Gamma) = \emptyset$. There are $K = (\Lambda : \Gamma)$ distinct cosets of Γ in Λ. If $\mathbf{b}_1, \ldots, \mathbf{b}_K$ are arbitrary elements of these K cosets, denoted *coset representatives*, we have

$$\Lambda = \bigcup_{k=1}^{K} (\mathbf{b}_k + \Gamma). \tag{7.81}$$

9. Let $f[\mathbf{x}]$, $\mathbf{x} \in \Lambda$ be a scalar signal defined on a lattice Λ. We define the *Fourier transform* of $f[\mathbf{x}]$ to be

$$F(\mathbf{u}) = \sum_{\mathbf{x}\in\Lambda} f[\mathbf{x}] \exp(-j2\pi\mathbf{u}\cdot\mathbf{x}) \tag{7.82}$$

where $\mathbf{u} = [u \;\; v]^T$ is a two-dimensional frequency vector, expressed in cycles per unit of length. The Fourier transform is a periodic function of the continuous frequency vector, with periodicity given by the reciprocal lattice: $F(\mathbf{u}) = F(\mathbf{u}+\mathbf{r})$ for all $\mathbf{r} \in \Lambda^*$. The Fourier transform of $f[\mathbf{x}]\exp(j2\pi\mathbf{u}_0 \cdot \mathbf{x})$ is $F(\mathbf{u}-\mathbf{u}_0)$ for an arbitrary fixed frequency vector $\mathbf{u}_0$; this is the modulation property.

10. A signal $f[\mathbf{x}]$, $\mathbf{x} \in \Lambda$ is periodic with periodicity lattice Γ if $f[\mathbf{x}+\mathbf{c}] = f[\mathbf{x}]$ for all $\mathbf{c} \in \Gamma$, where $\Gamma \subset \Lambda$. There are $K = (\Lambda : \Gamma)$ distinct values of this signal, which form one period. These are $f[\mathbf{b}_1], \ldots, f[\mathbf{b}_K]$, where $\mathbf{b}_1, \ldots, \mathbf{b}_K$ is an arbitrary set of coset representatives of Γ in Λ. The periodic signal is constant on cosets of Γ in Λ.

11. A periodic signal $f[\mathbf{x}]$, $\mathbf{x} \in \Lambda$, with periodicity lattice $\Gamma \subset \Lambda$ has the discrete Fourier series representation

$$f[\mathbf{x}] = \sum_{k=1}^{K} F[k]\exp(j2\pi\mathbf{x}\cdot\mathbf{d}_k), \quad \mathbf{x} \in \Lambda \tag{7.83}$$

where

$$F[k] = \frac{1}{K}\sum_{j=1}^{K} f[\mathbf{b}_j]\exp(-j2\pi\mathbf{b}_j \cdot \mathbf{d}_k), \quad k = 1, \ldots, K. \tag{7.84}$$

In these expressions, $K = (\Lambda : \Gamma)$, $\mathbf{b}_1, \ldots, \mathbf{b}_K$ are coset representatives for Γ in Λ and $\mathbf{d}_1, \ldots, \mathbf{d}_K$ are coset representatives for Λ^* in Γ^*.

References

[1] E. Dubois, "The sampling and reconstruction of time-varying imagery with application in video systems," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 502–522, April 1985.

[2] D.E. Dudgeon and R.M. Mersereau, *Multidimensional Digital Signal Processing*. Englewood Cliffs, New Jersey: Prentice-Hall, 1984.

[3] T. Yamada, *Image Sensors and Signal Processing for Digital Still Cameras*, ch. CCD image censors, J. Nakamura (ed.), Boca Raton, FL: CRC Press, 2005, pp. 95–141.

[4] K. Parulski and K.E. Spaulding, *Digital Color Imaging Handbook*, ch. Color image processing for digital cameras, G. Sharma (ed.), Boca Raton, FL: CRC Press, 2002, pp. 727–757.

[5] R. Lukac and K.N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1260–1267, November 2005.

[6] H.J. Trussell and R.E. Hartwig, "Mathematics for demosaicking," *IEEE Transactions on Image Processing*, vol. 11, no. 4, pp. 485–492, April 2002.

[7] D. Taubman, "Generalized Wiener reconstruction of images from colour sensor data using a scale invariant prior," in *Proceedings of the IEEE International Conference on Image Processing*, Vancouver, Canada, September 2000, vol. III, pp. 801–804.

[8] R. Lukac and K.N. Plataniotis, "Universal demosaicking for imaging pipelines with an RGB color filter array," *Pattern Recognition*, vol. 38, no. 11, pp. 2208–2212, November 2005.

[9] C.A. Poynton, *Digital Video and HDTV: Algorithms and Interfaces*. San Francisco, California: Morgan Kaufmann, 2003.

[10] T.K. Moon and W.C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Upper Saddle River, New Jersey: Prentice Hall, 2000.

[11] E. Dubois, "Frequency-domain methods for demosaicking of Bayer-sampled color images," *IEEE Signal Processing Letters*, vol. 12, no. 12, pp. 847–850, December 2005.

[12] D. Alleysson, S. Süsstrunk, and J. Hérault, "Linear demosaicing inspired by the human visual system," *IEEE Transactions on Image Processing*, vol. 14, no. 4, pp. 439–449, April 2005.

[13] E. Dubois and W.F. Schreiber, "Improvements to NTSC by multidimensional filtering," *SMPTE Journal*, vol. 97, no. 6, pp. 504–511, June 1988.

[14] J.J. Bean, "Cyan-magenta-yellow-blue color filter array," U.S. Patent 6 628 331-B1, September 2003.

[15] H. Hoshuyama, "Color image sensor, color filter array, and color imaging device," U.S. Patent application 2005/0212934, September 2005.

[16] S. Saito, "Solid state image pickup device having primary color and gray color filters and processing means thereof," U.S. Patent 7 126 633, October 2006.

[17] E. Dubois, "Color-filter-array sampling of color images: Frequency-domain analysis and associated demosaicking algorithms: Additional results," http://www.site.uottawa.ca/~edubois/SingleSensorImaging/, 2007.

[18] H.A. Aly and E. Dubois, "Design of optimal camera apertures adapted to display devices over arbitrary sampling lattices," *IEEE Signal Processing Letters*, vol. 11, no. 4, pp. 443–445, April 2004.

[19] G. Sharma, *Digital Color Imaging Handbook*, ch. Color fundamentals for digital imaging, G. Sharma Ed., CRC Press, Boca Raton, FL, 2002, pp. 1–114.

[20] G. Sharma and H.J. Trussell, "Figures of merit for color scanners," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 990–1001, July 1997.

[21] R. Bala, *Digital Color Imaging Handbook*, ch. Device characterization, G. Sharma Ed., CRC Press, Boca Raton, FL, 2002, pp. 269–382.

[22] E. Dubois, "Filter design for adaptive frequency-domain Bayer demosaicking," in *Proceedings of the IEEE International Conference on Image Processing*, Atlanta, GA, USA, October 2006, pp. 2705–2708.

[23] H.S. Malvar, L.W. He, and R. Cutler, "High-quality linear interpolation for demosaicing of Bayer-patterned color images," *Proceedings of the IEEE International Conference on Acoustics Speech Signal Processing*, Montreal, Canada, May 2004, vol. III, pp. 485–488.

[24] B. Chaix de Lavarène, D. Alleysson, and J. Hérault, "Practical implementation of LMMSE demosaicing using luminance and chrominance spaces," *Computer Vision and Image Understanding, Special Issue on Color Image Processing*, vol. 107, no. 1-2, pp. 3–13, July/August 2007.

[25] B.K. Gunturk, J. Glotzbach, Y. Altunbasak, R.W. Schafer, and R.M. Mersereau, "Demosaicking: Color filter array interpolation," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 44–54, January 2005.

8

Linear Minimum Mean Square Error Demosaicking

David Alleysson, Brice Chaix de Lavarène, Sabine Süsstrunk, and Jeanny Hérault

8.1 Introduction

Today, digital cameras are ubiquitous. One can be purchased for a price equivalent to a night's stay in a hotel in occidental countries, or it may be included for "free" in a mobile phone or a laptop computer. The existence of such a common and ultimately cheap device is due to fundamental developments in the way an image is acquired by the sensor and processed by the microcomputer embedded in the camera. These developments would not have been possible without recent breakthroughs in digital color image processing [1].

In this chapter, we focus on the sampling of images by a single camera sensor and on the subsequent digital image processing called *demosaicking* necessary to mimic the fact that a color image has three different color responses at each spatial position [2]. We highlight the properties of the human visual system (HVS) that have been exploited in the development

of digital cameras in general and demosaicking in particular. The workings of the human visual system are still a source of inspiration today because they have capabilities that are not yet taken into account in digital cameras. In particular, we discuss how the random nature of the arrangement of chromatic samples in the retina can improve color sampling by digital cameras.

This chapter is organized as follows. In the first section, we recall the properties of the human visual system that have been exploited in the development of digital color image processing. How the trichromacy of color vision has been discovered and is used in digital image processing is discussed. We describe how digital images are either acquired through three sensors or through a single sensor. In the second section, a model of spatio-chromatic sampling applicable to the human visual system and digital cameras is described. This model enables the understanding of the signal content in single-chip digital cameras. In the third section, methods of reconstructing color information from a mosaic through linear approaches are discussed. In the fourth section, we extend these approaches with adaptive processes.

8.1.1 Trichromacy in Human Vision

We can state that the history of color science started in the 16th century with the discovery made by Newton [3] who showed that sunlight was composed of several colors, the colors of the rainbow. By extension, the light is a combination of all the monochromatic lights of the visible spectrum. In practice, this discovery permits us to reproduce color sensations by modulating the intensity of different monochromatic primaries. However, it is not necessary to modulate the entire wavelength domain to reproduce a color sensation because, as discovered by Young [4] and then confirmed by Helmholtz [5], the human visual system is *trichromatic*. The property of trichromacy is that it is sufficient to modulate only three primaries to mimic the sensation of any light with arbitrary continuous spectrum, provided that none of the primaries can be matched with a combination of the other two. In general, we use a red (R), a green (G), and a blue (B) primary. These primaries are indicative of their maximum intensity in the visible spectrum: blue is the color of short wavelength radiation, green of middle wavelengths, and red of long wavelengths.

Trichromacy was established long before it was known that the human retina was composed of three kinds of cones sensitive to three different wavelength ranges [6], [7]. These cones are called L, M, and S for their respective sensitivity to long, middle and short wavelengths. Two lights that have the same L, M, and S cone responses give the same color sensation (such lights are called metamers). Thus, the dimension of the space that represents color in the human visual system is three. This property is used in digital capture systems and image displays.

Trichromacy is also the basis for colorimetry [8], the science of color measurement. The principle of colorimetry is based on color matching experiments. The experiments consist of comparing a color with an additive mixture of monochromatic primaries. The intensity of the primaries needed to obtain a match with a color sensation serves as an indicator of the color content in terms of these primaries. This method was used to standardize color spaces such as CIE-RGB and CIE-XYZ by the Commission Internationale de l'Éclairage (CIE) in 1931.

(a) (b) (c) (d)

FIGURE 8.1

Example of a color image decomposition into its red, green, and blue components: (a) color image, (b) red channel, (c) green channel, and (d) blue channel. Color version of the original image is available in Figure 8.2.

An important property of color spaces is that the mixture of light is a linear process. The color-space position of a light mixture can be derived by adding the coordinate vectors of the lights that make up that mixture. Also, in a color space defined by the φ_R, φ_G, and φ_B spectral sensitivities of a camera sensor, any light is defined by a linear combination of the Red (R), Green (G), and Blue (B) primaries. Thus, the RGB values of a digital image define the corresponding color as a linear mixture of the R, G, and B primaries.

8.1.2 Digital Color Image Encoding

Trichromacy is exploited in the formation, rendering, and reproduction of color images. As shown in Figure 8.1, a color image is a matrix with three components, one for R, one for G, and one for B, respectively. The rendering of these three components on a video screen, which has three RGB phosphors or three color filters, allows reproduction of the color sensation equivalent to that produced by the natural scene itself. Thus, the color processing chain from acquisition of a color image, coding of digital values, and rendering to the display can be designed using a three-dimensional space for all color representations.

An RGB representation is not the only color representation used in digital video and imaging. Chromatic information in images is often reduced to achieve smaller sized files. The human visual system is less sensitive to high frequencies in chrominance than in luminance. In other words, the spatial resolution of chrominance can be quite a bit lower than the resolution of luminance without observing any visual degradation in the image. A luminance-chrominance representation is analogous to the receptive field encoding at the ganglion cells in the human retina. Luminance represents spatial information in terms of light-dark changes, such as edges, while chrominance represents the hue and saturation of a color (see Section 8.2.1).

There are several ways to construct a luminance-chrominance representation of a color image. For example, we can transform the R, G and B values into a triplet called Y, Cb, and Cr, where Y represents the luma and is a positive combination of R, G, and B values. In general, the RGB values are already "gamma-corrected," meaning that a nonlinear encoding has been applied to the color channels, and thus Y is not representative of the physically measurable luminance anymore. Cr and Cb are two opponent chromatic channels. The amount of image data can be reduced by retaining all Y values, but subsampling Cb and Cr by a factor of two or four without significant visual loss.

8.1.3 Image Acquisition through Single Chip Digital Camera

Similarly to the human retina, where we have only one cone type per spatial location, most digital cameras today use a single sensor to capture color images. The sensor is covered by an array matrix of color filters to allow the acquisition of different chromatic contents of the scene. In general, the filters transmit either blue, green, or red light. Consequently, a single chromatic value is sampled at each spatial location. To reconstruct three chromatic values from the mosaic of single values, we need to use a signal processing method that is called demosaicking.

There are many color filter arrays (CFAs) for digital cameras. The problem to take into account when designing a CFA is the ability to fully reconstruct a color image with three chromatic components from the mosaic of single color values. The sampling of a scene with a single color filter per spatial location results in a compromise in the representation of spatial versus chromatic information of the image. Spatial and chromatic information is present as a mixture in a CFA image. Thus, can we design an arrangement of color filters that maximizes the ability to fully reconstruct the spatial and chromatic content of the scene? This question is still unresolved, but we discuss the properties of several CFA's in terms of their spatial and chromatic representation of light in Section 8.2.2.

The first proposed color filter array was composed of vertical stripes of red, green, and blue columns. It has also been proposed that the stripes could be oriented. These arrangements, even if they are easy to build, have not been used extensively because the color sampling frequencies are not the same in horizontal and vertical direction.

Another CFA was proposed by Bayer [9] in 1976. It fulfills two constraints. It has a color sampling frequency that is the same in vertical and horizontal direction for all three colors. It also has two times more green pixels than red and blue, favoring the sampling of luminance as stated by the inventor. The diagonal sampling frequency is thus higher for green than for red and blue. In the beginning, this CFA was not as successful as today. Recall that in 1976, most electronic image capture was for analog color television, which used an interlaced image encoding. Interlacing video means that the first frame displays only even image lines, and then the second frame displays the odd lines. This method reduces the amount of data to be processed at one time, and it became a standard for color television. The problem with the Bayer CFA is that the red or blue values are sampled only on either the even or the odd lines, respectively. This has the consequence that the red and blue colors flickered when video frames were displayed in interlaced mode. To compensate for that Dillon [10] proposed another CFA where red and blue are present at every line.

However, the Bayer CFA is certainly nowadays the most popular CFA in digital cameras because it has a good representation of chromatic and spatial information. We will also discuss a CFA proposed by Lukac [11] that is an improvement of the Bayer CFA for the horizontal representation of luminance values.

There exist several other CFAs, which use either four colors or a hexagonal arrangement of the color filters. We do not discuss these CFAs in this chapter. Also, a different method for acquiring a color image with a single sensor has recently been proposed by Foveon [12]. This method uses the fact that the penetration of light in silicon depends on the wavelength of the light. It allows the separation of red, green, and blue at each pixel by reading the responses at different well depths, and is not further discussed.

Finally, similarly to the human visual system that has a random arrangement [13] of cones at the surface of the retina, we study the case of a random arrangement of chromatic samples for digital cameras. The problem with a random arrangement is that the neighborhood of a sample changes from location to location and uniform space invariant reconstruction methods cannot be used. However, using a pattern of random color filters periodically repeated on the sensor surface, we benefit from the nonaliasing properties of random sampling and are able to reconstruct the color with a linear method.

8.2 Color Filter Array Signal Representation

In this section, we show that the representation of a color image in luminance and opponent chromatic channels is better than in red, green, and blue when considering demosaicking. This representation allows distinguishing the spatial and chromatic contents of a scene. Moreover, it is still effective even in the case of a mosaic image with a single chromatic value per spatial position.

8.2.1 Luminance-Chrominance Representation of Color Images

Considering the trichromatic properties of the human visual system discussed in the previous section, it has become natural to think that a color image should be represented by three components. These components are representative of the energy measured by three sensors with different sensitivities, usually red, green, and blue.

An RGB representation of a color image does not highlight the two main properties of our visual perception of scenes: the ability to account for both the intensity of the light source on an object surface and the colors of the object. However, since the tristimulus values form a vector space, a linear transformation from this space to another with different properties is possible without drastically changing the nature of the data. A linear transformation also allows for an inverse transformation.

The transformation should decorrelate as much as possible the spatial information from chromatic content to allow for processing the color data without aliasing effects. The spatial component should contain no chromatic content, while the chromatic component should be free of any intensity information. Intuitively, the transformation should be a positive combination of red, green, and blue values for the achromatic channel, whereas it should be a difference of these values for the chromatic channel, with an average value that is zero. There exists an infinite number of possible transformations from RGB to achromatic and chromatic color channels following these rules. Actually, the achromatic information of a tristimulus value is not uniquely defined. The achromatic channel has a spectral sensitivity response that is defined by the respective contributions of the spectral sensitivity responses of each R, G and B channel.

Let $I = \{C_R, C_G, C_B\}$ be a color image with three color planes Red, Green, and Blue. The projection of the color channels on an achromatic axis is given by the following sum, with the assumption that p_i (for $i \in \{R, G, B\}$ and $\sum_i p_i = 1$) is the proportion of the chromatic

(a) (b) (c)

FIGURE 8.2 (See color insert.)

Example of color image decomposition into its luminance and chrominance components. (a) Color image with three chromatic values R, G and B at each spatial location. (b) The luminance component, a scalar value per spatial position corresponding to the mean of RGB values. (c) The chrominance values having three components per spatial position corresponding to the difference of each RGB component with the luminance.

component i in the achromatic signal ϕ:

$$\phi = \sum_i p_i C_i \tag{8.1}$$

The content of the signal C_i can be expressed by the irradiance of the scene $S(x,y,\lambda)$ through the spectral sensitivity responses of the filters φ_i $(i = R,G,B)$ of the acquisition process:

$$C_i(x,y) = \int_\lambda S(x,y,\lambda)\varphi_i(\lambda)d\lambda \tag{8.2}$$

From Equations 8.1 and 8.2, note that luminance has a spectral sensitivity response given by the spectral sensitivity functions φ_i and the weights p_i of the chromatic channels. The spectral sensitivity function for luminance is given by $\sum_i p_i\varphi_i$. This means that the luminance is directly dependent on both the spectral sensitivity functions of the chromatic channels R, G, and B and the proportion of each of these channels that give the luminance signal.

For the human visual system, the luminance signal is not well defined. The CIE recognizes the $V(\lambda)$ function [8] as being the luminosity function of the standard observer, which should represent the human ability to assess the sensation of luminosity. It is therefore possible to simulate this function during the luminance computation by choosing the appropriate coefficients p_i that match as best as possible the spectral sensitivity response of the luminance function to the shape of $V(\lambda)$.

Concerning chrominance, the debate is even more open. There is no real consensus on what the human visual system computes as chromatic channels. Some indications are given by the work of Hering [14] and Jameson and Hurvich [15], which estimated opponent color responses psychophysically.

Recall that our chrominance is defined as having a zero mean to avoid any intensity information. Also, since the dimension of the chromatic color space is three, consider that the chrominance space is also of dimension three to take into account the variability along each of the chromatic axes. However, these variables are linearly dependent and the

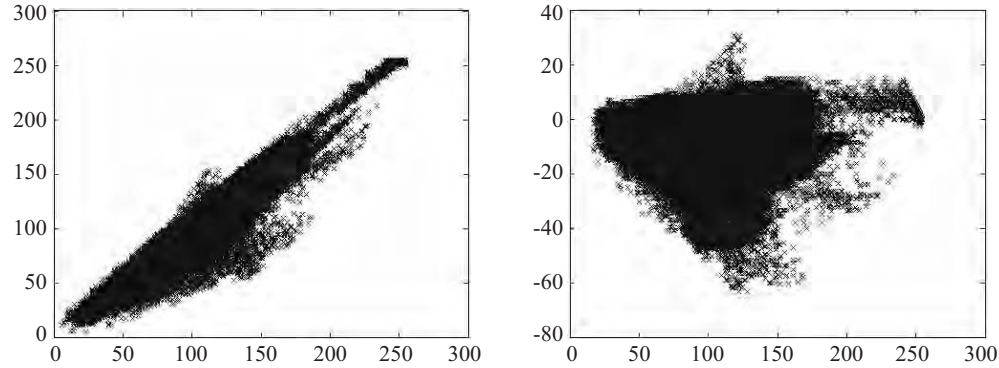


FIGURE 8.3

Plots of B as a function of G (left) and ψ_B as a function of ϕ (right) for the example of Figure 8.2. We can see that B and G have a principal component in $y = x$, while ψ_B and ϕ are more decorrelated.

intrinsic dimension is thus only two. If we consider the difference between each chromatic channel and the luminance function defined in Equation 8.1, we have:

$$\psi_i = C_i - \phi = C_i - \sum_i p_i C_i = (1 - p_i)C_i - \sum_{j \neq i} p_j C_j \tag{8.3}$$

Since $\sum_i p_i = 1$, it follows that $\sum_i p_i \psi_i = 0$, meaning that the weighted average of the chrominance components is zero at each pixel. Our definitions of luminance and chrominance respect the condition we have imposed. Figure 8.2 shows an example of a luminance chrominance decomposition. Note the separate representation of spatial and chromatic information compared to Figure 8.1, where spatial information is present in all three of the R, G and B channels. Figure 8.3 plots B values as a function of G values, and ψ_B as a function of ϕ. We can observe that chrominance and luminance are effectively more decorrelated than the color planes B and G.

8.2.2 Luminance-Chrominance in Color Filter Arrays

To model the sampling of chromatic values by a mosaic, we use the formal construction of the mosaic itself. A mosaic m is a spatial arrangement of chromatic samples. With the assumption that the samples are on a grid (either square or hexagonal) and that all the positions on the grid are filled by one filter color, we can decompose the global mosaic into submosaics, m_R, m_G and m_B. Each one corresponds to the spatial arrangement of one type of color filter, having a value of one where the filter is present and a value of zero at the empty positions:

$$m(x,y) = m_R(x,y) + m_G(x,y) + m_B(x,y) \tag{8.4}$$

with $(x,y) \in \mathbb{N}^2$ the integer coordinates on the grid.[1]

[1]Note that in this chapter, x and y are oriented using Matlab convention, where the origin is at the top left and where x is the vertical axis and y the horizontal axis.

Since the mosaics are periodic, their Fourier transforms result in Dirac impulses in frequency domain. One period of the mosaics' Fourier transforms is given by:

$$\hat{m} = \delta_0 \qquad \text{and} \qquad \begin{cases} \hat{m}_R = r_0\delta_0 + \sum_{n\neq 0} r_n\delta_n \\ \hat{m}_G = g_0\delta_0 + \sum_{n\neq 0} g_n\delta_n \\ \hat{m}_B = b_0\delta_0 + \sum_{n\neq 0} b_n\delta_n \end{cases} \tag{8.5}$$

where $\delta_0 = \delta(\nu_x, \nu_y)$ is the Dirac distribution for spatial frequency dimension ν_x and ν_y of spatial dimension x and y and $\delta_n = \delta(n_x - \nu_x, n_y - \nu_y)$. Thus, δ_n denotes the Dirac impulse at spatial frequency $n = (n_x, n_y)$. From the analysis of the Bayer CFA pattern, for example, n describes the set of frequency dimensions [16]:

$$(\nu_x, \nu_y) \in \{-1, 0, 1\} \text{ and } (\nu_x, \nu_y) \neq (0, 0) \tag{8.6}$$

where 1 is the normalized Nyquist frequency.

The terms r_0, g_0 and b_0 denote the mean values of each submosaic, i.e., the probability at each spatial location to contain a sample of the respective color. We call them p_R, p_G and p_B, respectively. By unicity of the Fourier transform, we can conclude:

$$\begin{cases} p_R + p_G + p_B = 1 \\ r_n + g_n + b_n = 0 \qquad (\forall n \neq 0) \end{cases} \tag{8.7}$$

The resulting CFA image I_m (a single channel image containing the different color submosaics) is obtained by:

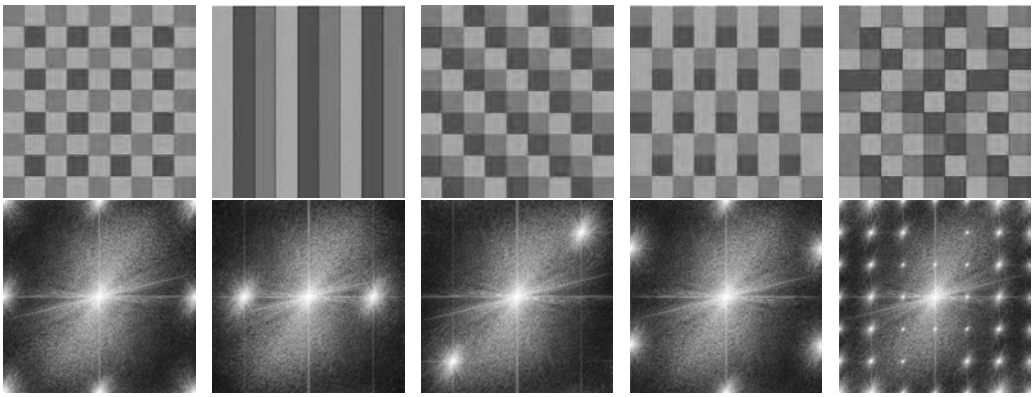
$$I_m(x, y) = \sum_{i\in\{R,G,B\}} C_i(x, y) m_i(x, y) \tag{8.8}$$

The multiplication in Equation 8.8 becomes a convolution product in the Fourier domain. Hence, using Equation 8.5 and given that a convolution product with a Dirac impulse corresponds to a translation to the corresponding frequency of the Dirac impulse, we obtain:

$$\hat{I}_m(\nu) = \underbrace{\sum_i p_i \hat{C}_i(\nu)}_{\hat{\phi}(\nu)} + \sum_{n\neq 0} \underbrace{\begin{matrix} r_n\hat{C}_R(n-\nu) \\ + g_n\hat{C}_G(n-\nu) \\ + b_n\hat{C}_B(n-\nu) \end{matrix}}_{\hat{\psi}_n(n-\nu)} \tag{8.9}$$

Since ϕ is a linear combination of color signals with positive weights, it represents luminance. The term $\psi_n(n-\nu)$ is a linear combination of color signals with coefficients whose sum vanishes, modulated at frequency n. It represents modulated chrominance.

Figure 8.4 shows five examples of CFAs and the amplitude spectra of an image sampled by those CFAs. Note that the CFA pattern determines the location of the chrominance, and thus controls the amount of aliasing between the baseband luminance and the modulated chrominances. For additional information on CFAs refer to Chapters 1 and 5.



(a) (b) (c) (d) (e)

FIGURE 8.4

CFA patterns and amplitude spectra of corresponding CFA images: (a) Bayer pattern, (b) vertical stripe pattern, (c) diagonal stripe pattern, (d) Lukac pattern [11], and (e) tiling of a 6×6 pseudorandom pattern. Figure 5.2 shows presented CFAs in color.

8.2.3 Examples of Practical CFAs

The formalism developed in Section 8.2.2 allows revisiting the notion of opponent chromatic channels in the case of digital camera acquisition and of the human visual system. The fact that the mosaic construction, i.e., the arrangement of chromatic samples of each type, defines the nature of the luminance and chromatic channels is very important and could be used either for the design of appropriate cameras or to understand the nature of opponent chromatic channels in the visual system.

It was shown in Equation 8.9 that the position of chrominance in the Fourier spectrum depends on the spatial arrangement of each corresponding chromatic value in the CFA. For the Bayer CFA, for example, the repetition of the R color filter is one out of two pixels in horizontal and vertical directions. This places the red chrominance at the border of the Fourier spectrum. To show precisely where the chrominances are located, it is useful to mathematically describe the arrangement of the chromatic values in the CFA. For the Bayer pattern, if we consider the pixel at position (0,0) to be a red pixel, we have:

$$\begin{cases} m_R(x,y) = (1+\cos\pi x)(1+\cos\pi y)/4 \\ m_G(x,y) = (1-\cos\pi(x+y))/2 \\ m_B(x,y) = (1-\cos\pi x)(1-\cos\pi y)/4 \end{cases} \tag{8.10}$$

This equation can be rewritten separating the constant part and the modulation part given by the cosine function:

$$\begin{cases} m_R(x,y) = 1/4 + \underbrace{(\cos\pi x + \cos\pi y + \cos\pi x\cos\pi y)/4}_{\tilde{m}_R} \\ m_G(x,y) = 1/2 \underbrace{-\cos\pi(x+y))/2}_{\tilde{m}_G} \\ m_B(x,y) = 1/4 + \underbrace{(-\cos\pi x - \cos\pi y + \cos\pi x\cos\pi y)/4}_{\tilde{m}_B} \end{cases} \tag{8.11}$$

which shows that the decomposition of luminance and chrominance is given by the mosaic arrangement. It follows that the luminance part in the Bayer CFA is defined as:

$$\phi_{Bayer}(x,y) = \sum_i p_i C_i(x,y) = \frac{R+2G+B}{4} \tag{8.12}$$

with $R = C_R(x,y)$, $G = C_G(x,y)$ and $B = C_B(x,y)$. The chrominance in the Bayer CFA is then defined as the difference between the CFA image and the luminance image:

$$\begin{aligned} \psi_{Bayer}(x,y) &= I_m(x,y) - \phi_{Bayer}(x,y) \\ &= \sum_i m_i(x,y) C_i(x,y) - p_i C_i(x,y) \\ &= \sum_i \tilde{m}_i(x,y) C_i(x,y) \end{aligned} \tag{8.13}$$

The Fourier transform $\hat{\psi}_{Bayer}$ can be explicitly calculated and decomposed into two modulated chrominance components:

$$\hat{\psi}_{Bayer}(\nu_x,\nu_y) = \hat{\psi}_1(\nu) * \sum_{n\in N_1} \delta_n + \hat{\psi}_2(\nu) * \sum_{n\in N_2} \delta_n \tag{8.14}$$

where $*$ denotes a convolution, with the set of frequencies $N_1 = \{(1,1)\}$ and $N_2 = \{(0,1),(1,0)\}$ and with

$$\begin{cases} \hat{\psi}_1(\nu) = (\hat{R} - 2\hat{G} + \hat{B})/16 \\ \hat{\psi}_2(\nu) = (\hat{R} - \hat{B}/8) \end{cases} \tag{8.15}$$

The chrominance ψ_{Bayer} actually reflects a three dimensional space of chromatic opponent channels, which is subsampled and coded in a single lattice. Let us compute the product $\psi_{Bayer}(x,y)m_i(x,y)$, which selects from the chrominance image the values corresponding to the chromatic value i in the Bayer CFA. Using $m_i = p_i + \tilde{m}_i$ and the fact that $m_i m_j = 0$ for $i \neq j$, we have:

$$\begin{aligned} \psi_{Bayer}(x,y) m_i(x,y) &= \sum_j \tilde{m}_j(x,y) m_i(x,y) C_j(x,y) \\ &= \underbrace{\left[(1-p_i) C_i(x,y) - \sum_{j\neq i} p_j C_j(x,y) \right]}_{\psi_i} m_i(x,y) \end{aligned} \tag{8.16}$$

Chrominance is thus composed of three different opponent chromatic channels. For the Bayer CFA, these channels are:

$$\begin{cases} \psi_R = (3R - 2G - B)/4 \\ \psi_G = (-R + 2G - B)/4 \\ \psi_B = (-R - 2G + 3B)/4 \end{cases} \tag{8.17}$$

The principle of recovering the three chromatic opponent channels from the chrominance image ψ_{Bayer} can be done in two equivalent ways. First, demultiplexing (i.e., multiplying) with the function m_i will bring back the chrominances ψ_R, ψ_G and ψ_B to the center of the

spectrum of each color plane [16]. Since m_i also modulates high frequencies, the operation has to be followed by a low-pass filter. Another way is to directly estimate the modulated chrominances ψ_1 and ψ_2 using two different filters, and then to demodulate them to low frequencies [17], [18]. Refer to Chapters 5 and 7 for additional information on frequency multiplexing and demultiplexing.

The same equations can be derived for Lukac's CFA. The mosaics can be written formally as:

$$\begin{cases} m_R(x,y) = 1/4 + \cos(\pi y)\cos(\frac{\pi}{2}x)/2 + \cos(\pi x)/4 \\ m_G(x,y) = 1/2 - \cos(\pi y)\cos(\pi x/2)/2 - \cos(\pi y)\cos(\frac{\pi}{2}(x-1))/2 \\ m_B(x,y) = 1/4 + \cos(\pi y)\cos(\frac{\pi}{2}(x-1))/2 + \cos(\pi(x-1))/4 \end{cases} \tag{8.18}$$

which yields the modulated chrominances $\hat{\psi}_{Lukac}(\nu_x,\nu_y) = \hat{\psi}_1(\nu) * \sum_{n\in N_1}\delta_n + \hat{\psi}_2(\nu) * \sum_{n\in N_2}\delta_n + \hat{\psi}_3(\nu) * \sum_{n\in N_3}\delta_n$:

$$\begin{cases} \hat{\psi}_1(\nu) = (\hat{R} - (1+j)\hat{G} + j\hat{B})/8 \\ \hat{\psi}_2(\nu) = (\hat{R} - \hat{B})/8 \\ \hat{\psi}_3(\nu) = (\hat{R} - (1-j)\hat{G} - j\hat{B})/8 \end{cases} \tag{8.19}$$

with $j = \sqrt{(}-1))$ and $N_1 = \{(1,\frac{1}{2})\}$, $N_2 = \{(0,1)\}$ and $N_3 = \{(1,-\frac{1}{2})\}$. The chrominances are also located at the border of the spectrum (see Figure 8.4), but contrary to the Bayer CFA, there is maximum resolution in the horizontal direction. The luminance and demodulated chrominances are actually the same as for the Bayer CFA, since the proportions p_i are the same in both CFAs.

8.3 Linear Systems for Luminance-Chrominance Estimation

The missing values in a mosaic of chromatic samples can be reconstructed through a linear combination of the neighboring pixel values. In this section, we describe several methods, ranging from the simple copy of pixels and bilinear interpolation to more sophisticated methods taking into account the properties of CFA images. We also describe demosaicking as a generalized inverse problem.

According to Reference [9], demosaicking was originally not a priority in the 1970's. Cameras were essentially designed for color television and the number of pixels in the camera was designed to match the number of lines of the television format. Moreover the TV color format was YCbCr 4:2:2, thus at least at acquisition no interpolation was required.

Later, the reconstruction of missing color values became necessary to display appropriate color on the screen. The first method was to simply copy the values of the neighboring pixel. Figure 8.5a depicts the method for the Bayer CFA whereas Figure 8.6a shows the result of applying this method to an image. Soon after, bilinear interpolation was proposed, which consists of averaging the values from the closest neighbors in the horizontal and vertical directions. Following the Bayer CFA depicted in Figure 8.5b, the linear interpolation of the green and blue pixels at position (2,2) is given by:

$$G_{22} = (G_{12} + G_{21} + G_{32} + G_{23})/4, \quad B_{22} = (B_{11} + B_{31} + B_{13} + B_{33})/4 \tag{8.20}$$

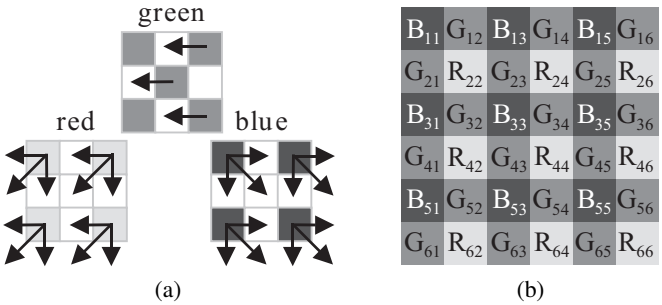


FIGURE 8.5

Illustration of pixel copy and bilinear interpolation.

Bilinear interpolation operates on the three color channels in isolation. If we consider each channel with its existing values and zeros at the missing values, it can easily be shown that the following convolution filter allows interpolating the missing color by bilinear interpolation.

$$\mathbf{F}_G = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} /4; \quad \mathbf{F}_{RB} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} /4 \tag{8.21}$$

Both methods can produce significant color artifacts (see Figure 8.6b) and are thus not always suitable. The development of digital cameras for general use and the possibility of embedding a microprocessor in the camera have encouraged investigations into more sophisticated demosaicking algorithms, as discussed in the next sections.

8.3.1 Linear Estimation Using Constant Ratio Hypothesis

To improve the result of bilinear interpolation, Cok [19], [20] proposed interpolating hue instead of each color channel separately. This method is based on the observation that hue tends to remain constant over a local neighborhood. Hue is calculated as the ratio between red and green (R/G) and between blue and green (B/G). The method is as follows. First, the missing green pixel values are calculated with bilinear interpolation. Then, the ratios R/G and B/G are computed at the respective pixel positions. The missing hue values are then interpolated, and the respective color (red or blue) is obtained by multiplying each hue value with the corresponding green values. For example, the interpolation of the blue pixel at position (2,2) is given by:

$$B_{22} = G_{22} \frac{\frac{B_{11}}{G_{11}} + \frac{B_{13}}{G_{13}} + \frac{B_{31}}{G_{31}} + \frac{B_{33}}{G_{33}}}{4} \tag{8.22}$$

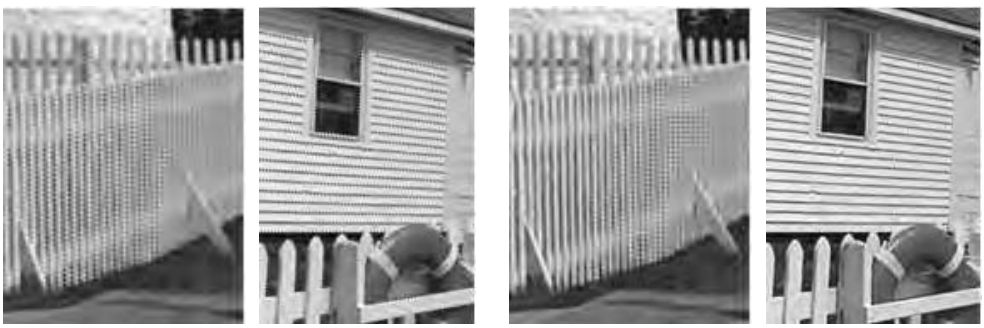
Figure 8.7a shows a result of demosaicking with this method.

An interesting exploitation of the constant hue algorithm was proposed by Crane et al. [21]. They used the constant hue hypothesis to derive convolution filters, which apply directly on the CFA rather than separately in each color plane. They originally designed their method for a CFA called Chromaplex; here we show the algorithm for the Bayer CFA.

(a) (b)

FIGURE 8.6 (See color insert.)

Examples of demosaicking by (a) pixel copy and (b) bilinear interpolation.



(a) (b)

FIGURE 8.7 (See color insert.)

Examples of demosaicking by (a) constant hue and (b) predetermined coefficients.

Suppose that we want to interpolate the red pixel at position (3,3). We denote with capital letters the existing color values and with lowercase letters the missing values, and write

$$r_{33} = B_{33} - \bar{B} + \bar{R} \tag{8.23}$$

where $\bar{B}$ and $\bar{R}$ are the averages of blue and red in the neighborhood. These averages can be expressed with existing values in the neighborhood as follows:

$$r_{33} = B_{33} - (B_{33} + B_{31} + B_{13} + B_{35} + B_{53})/5 + (R_{22} + R_{24} + R_{42} + R_{44})/4 \tag{8.24}$$

We obtain the following filter with integer coefficients by multiplying by a factor of 20:

$$\frac{1}{20}\begin{bmatrix} 0 & 0 & -4 & 0 & 0 \\ 0 & 5 & 0 & 5 & 0 \\ -4 & 0 & 16 & 0 & -4 \\ 0 & 5 & 0 & 5 & 0 \\ 0 & 0 & -4 & 0 & 0 \end{bmatrix}$$

Note that there are many different filters possible, dependent on the size of the extension we want to give the filters. We have tested many and report only those filters that give the best results for the images we used.

For the Bayer CFA, we need to study several positions. The previous example can be used to find blue values at red pixel positions. Namely, using the following filter

$$f_1 = \frac{1}{80}\begin{bmatrix} 0 & 0 & -4 & 0 & 0 \\ 0 & 5 & 0 & 5 & 0 \\ -4 & -16 & 12 & -16 & -4 \\ 0 & 30 & 64 & 30 & 0 \\ -4 & -16 & 12 & -16 & -4 \\ 0 & 5 & 0 & 5 & 0 \\ 0 & 0 & -4 & 0 & 0 \end{bmatrix}$$

results in red values at positions (2,3) and (4,5) and blue values at positions (3,2) and (3,4). The second filter

$$f_2 = \frac{1}{80}\begin{bmatrix} 0 & 0 & -4 & 0 & -4 & 0 & 0 \\ 0 & 5 & -16 & 30 & -16 & 5 & 0 \\ -4 & 0 & 12 & 64 & 12 & 0 & -4 \\ 0 & 5 & -16 & 30 & -16 & 5 & 0 \\ 0 & 0 & -4 & 0 & -4 & 0 & 0 \end{bmatrix}$$

is suitable to obtain red values at positions (3,2) and (3,4) and blue values at positions (2,3) and (4,3). Finally, the third filter

$$f_3 = \frac{1}{100}\begin{bmatrix} 0 & 0 & 4 & 0 & 4 & 0 & 0 \\ 0 & -5 & 0 & -10 & 0 & -5 & 0 \\ 4 & 0 & -8 & 25 & -8 & 0 & 4 \\ 0 & -10 & 25 & 60 & 25 & -10 & 0 \\ 4 & 0 & -8 & 25 & -8 & 0 & 4 \\ 0 & -5 & 0 & -10 & 0 & -5 & 0 \\ 0 & 0 & 4 & 0 & 4 & 0 & 0 \end{bmatrix}$$

can be used to recover all missing green values. Figure 8.7b shows a reconstruction example with these filters.

8.3.2 Filter Design from the CFA Spectrum

The CFA sampling model in Section 8.2.1 shows that the luminance and chrominance are localized in the Fourier spectrum. Thus, a filter that is able to select the frequency components of the luminance independently from those of the chrominance can act as a luminance selection filter. We can even derive a linear space-invariant filter for estimating luminance [16], [22]; the processing then does not depend on the position considered in the image.

The method uses a linear shift invariant finite response filter to estimate the luminance in the CFA image. Generally, a filter of size 9×9 gives accurate enough results. Once the luminance is estimated, it is subtracted from the CFA image. The resulting image now only contains chrominance. This is equivalent to applying the orthogonal filter for estimating the chrominance. Then, the chrominance is demultiplexed according to the red, green, and blue CFA arrangements, resulting in three images containing opponent chromatic channels. Demultiplexed chrominance is then interpolated using simple bilinear filters, such as of Equation 8.21. The principle of demosaicking by frequency selection is shown in Figure 8.8.

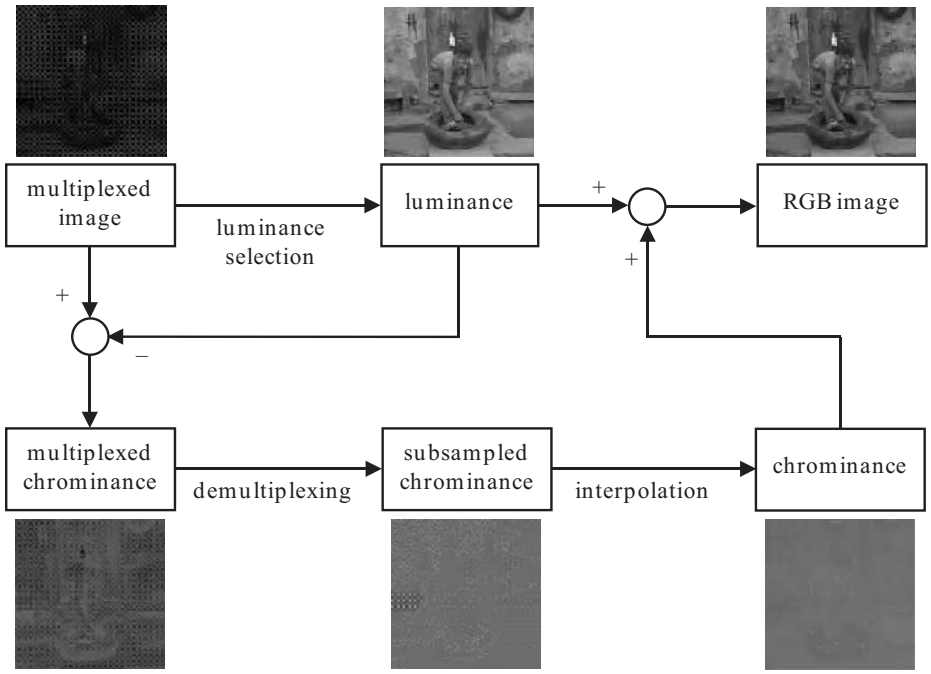


FIGURE 8.8

Synopsis of the demosaicking by frequency selection.

The results of this method depend on the linear space-invariant filter, which needs to accurately estimate the luminance information. As shown in Figure 8.4, the luminance is a wide-band signal that needs to be estimated in frequencies close to the Nyquist frequency while avoiding frequencies containing chrominance. The filter design thus depends on the spatial and color characteristics of the camera. A generic filter was proposed in Alleysson et al. [16] for general purpose use.

In terms of reconstruction quality, using an invariant filter is not the best solution because the structure of the CFA favors some positions, such as the green pixels in the Bayer CFA. This will be discussed in the next section.

8.3.3 Wiener Estimation

This method allowing for a direct estimation of a space variant filter was originally proposed by Trussell and Taubman [23], [24]. The idea is to express demosaicking as an inverse problem. Given the data acquired by the sensor, is it possible to reconstruct the original scene? The problem could then be solved with a Wiener approach, which supposes that the original data can be retrieved from a linear combination of the acquired data. The calculation of the linear parameters can be performed with a linear minimum mean square error between acquired data and original data.

Trussell [23] formalized Wiener demosaicking by considering that the original data is the multispectral content of the scene. He also takes into account the optics of the camera. Taubman [24] proposes a practical method to solve the system by taking into account that the process of acquisition is space invariant.

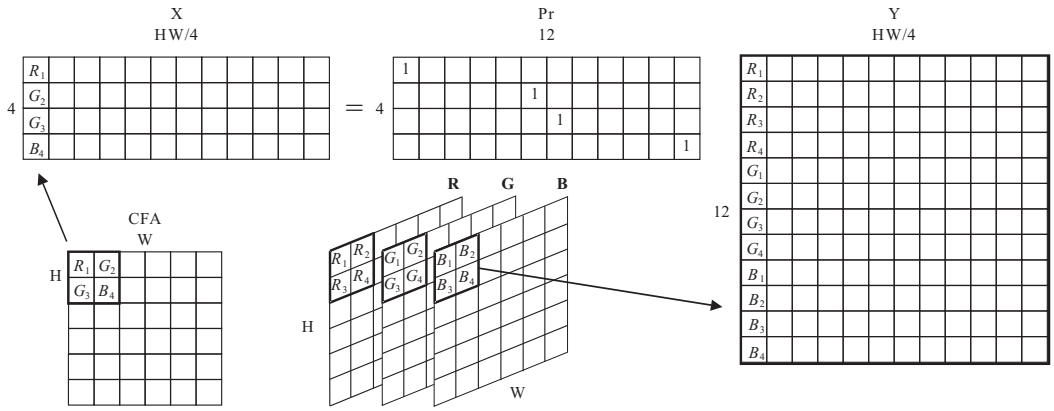


FIGURE 8.9

Illustration that a CFA image **X** is constructed from a matrix multiplication between **Pr** and the color image **Y** if they are represented as column-stacked superpixels.

In this section we discuss the principles of Wiener demosaicking by simplifying the model, considering only the transformation of the RGB image to a mosaic, as previously described in Chaix et al. [25]. We show how we can use an image database to constrain the solution. Also, we describe a luminance-chrominance approach to Wiener demosaicking. Finally, we use this method to compare the performances of different CFA's.

8.3.3.1 Direct RGB Estimation

The principle of the formalism of Wiener demosaicking is the use of stacked notation that unfolds the color image of size $H \times W \times 3$ into a column vector of size $HW3 \times 1$, where H, W, and 3 are respectively the height, the width, and the number of color channels in the image. This allows us to express the model of image formation as a matrix multiplication between the original image and the CFA sampling matrix. In Reference [24], Taubman introduced the concept of *superpixel*. A superpixel is a group of pixels that matches the basic pattern of the CFA. In the Bayer CFA, the basic pattern is composed of four pixels arranged on a 2×2 square: one red, two green, and one blue. At the scale of the superpixel, the mosaic is regular, a tiling of superpixels. With the assumption that the acquisition process is invariant over the image, which is widely used, it allows the design of space-invariant filters at that scale, i.e., of block shift-invariant filters [26] at the scale of a pixel. Thus, the stacked notation should be expressed at the scale of a superpixel, as shown in Figure 8.9 and the following equation:

$$\mathbf{X} = \mathbf{PrY} \tag{8.25}$$

with **Pr** being a projection operator that represents the sampling process, converting four pixels of the image with three colors per pixel of **Y** to four single-color pixels of the CFA image **X**.

The goal of linear demosaicking is to find a matrix **D** that will recover the color image $\tilde{\mathbf{Y}}$ from the CFA image **X**:

$$\tilde{\mathbf{Y}} = \mathbf{DX} \tag{8.26}$$

minimizing the mean square error e with the original color image $\mathbf{Y}$:

$$e = E[\|\mathbf{Y} - \tilde{\mathbf{Y}}\|^2] \tag{8.27}$$

The classical solution to this equation is the Wiener solution given by:

$$\mathbf{D} = (E[\mathbf{Y}\mathbf{X}^T])(E[(\mathbf{X}\mathbf{X}^T)])^{-1} \tag{8.28}$$

We can compute matrix $\mathbf{D}$ by applying Equation 8.28 over a database of full resolution color images. The use of a database means that we explicitly know $\mathbf{Y}$ and that we simulate the CFA image $\mathbf{X}$ using Equation 8.25. This computation requires only the inversion of a matrix of size $4n^2 \times 4n^2$ (n being the size of the neighborhood in superpixels). The details of the method are described by Chaix et al. in Reference [25]. A similar approach was recently used in Reference [27] by defining spatio-chromatic covariance matrices for the four elements of the superpixel.

8.3.3.2 Estimation through Luminance and Chrominance

Instead of directly estimating the color image from the CFA, as described in the previous section, we can in a first step estimate the luminance $\tilde{\Phi}$ from the CFA:

$$\tilde{\Phi} = \mathbf{H}_\Phi \mathbf{X} \tag{8.29}$$

where $\mathbf{H}_\Phi$ is the luminance filter (Figure 8.10). Once the luminance is estimated, we recover the modulated chrominance as the difference between the CFA image and the luminance $\tilde{\Psi} = (\mathbf{X} - \tilde{\Phi})$. As suggested, we demultiplex the chrominance by multiplying it by $\mathbf{Pr}^T$ before interpolating it to obtain the full chrominance $\tilde{\Psi}_c$:

$$\tilde{\Psi}_c = \mathbf{H}_\Psi \mathbf{Pr}^T \tilde{\Psi} \tag{8.30}$$

where $\mathbf{H}_\Psi$ is the matrix containing the three chrominance interpolating filters. Finally, the reconstructed color image $\tilde{\mathbf{Y}}$ is the sum of both parts:

$$\tilde{\mathbf{Y}} = \tilde{\Phi}_c + \tilde{\Psi}_c \tag{8.31}$$

where $\tilde{\Phi}_c = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T \otimes \tilde{\Phi}$.

We thus have to train two filters over the image database:

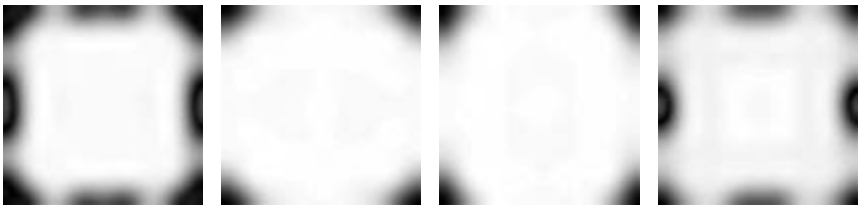
- the luminance estimator, calculated from the CFA image $\mathbf{X}$ (which is simulated from the database by setting the appropriate chromatic values to zero) and the luminance Φ (which is also computed from the database):

$$\mathbf{H}_\Phi = (E[\Phi \mathbf{X}^T])(E[(\mathbf{X}\mathbf{X}^T)])^{-1} \tag{8.32}$$

- and the chrominance interpolator, calculated from the chrominance Ψ_c and the subsampled chrominance Ψ (both computed from the database):

$$\mathbf{H}_\Psi = (E[\Psi_c(\mathbf{Pr}^T \Psi)^T])(E[(\mathbf{Pr}^T \Psi)(\mathbf{Pr}^T \Psi)^T)])^{-1} \tag{8.33}$$

  with $\Phi = \mathbf{P}\mathbf{Y}$, $\Psi_c = \mathbf{M}_c \mathbf{Y}$ and $\Psi = \mathbf{X} - \Phi$.



(a) (b) (c) (d)

FIGURE 8.10

Amplitude spectra of the luminance filter for each position (1, 2, 3 and 4) in the superpixel. At position 2 and 3 (G pixels), luminance can be retrieved with a maximal horizontal and vertical acuity.

The great advantage of using such a decomposition is that the chrominance has narrow bandwidths with respect to the Nyquist frequency of each demultiplexed plane. It thus requires only small order filters for interpolation. However, the luminance estimator needs to have high gradients at the frequencies located on the border between luminance and modulated chrominance. Therefore, it requires a high order filter for estimation (typically 7×7 or 9×9), but at least this estimation is performed only once. This property makes the algorithm much more computationally efficient than the direct RGB estimation.

8.3.3.3 Performance Comparison of Different CFAs

The Wiener approach combined with the estimation of the filters taken from a database can be used to compare different CFA performances. The procedure is automatic and since we consider the same neighborhood size, the only difference is the capability of the CFA to represent spatial and chromatic information. The advantage of using a linear method for reconstruction is that it does not favor a particular matching between the CFA and a particular nonlinear method. We used the leave-one-out method, so that the tested image was not included in the training set. We chose the Kodak image database, which is widely used in the demosaicking community.

We performed the comparison on the CFAs shown in Figure 8.4. The basic pattern of the random CFA is of size 6×6 pixels, with the same proportions of R, G and B pixels. This pattern was generated randomly and then manually readjusted in order to avoid any cluster of the same color filter. The results of the demosaicking process are given in Table 8.1 in

TABLE 8.1

Average PSNR values (dB) and standard deviations between the original and the reconstructed images of the 24 Kodak images and for each CFA configuration of Figure 8.4.

CFA	R	G	B	Average
Bayer	38.53 (±2.67)	41.22 (±2.47)	37.25 (±2.59)	39.00 (±2.58)
Vertical stripes	34.50 (±2.81)	34.61 (±2.82)	34.50 (±2.69)	34.54 (±2.77)
Horizontal stripes	32.95 (±2.46)	33.09 (±2.48)	33.16 (±2.26)	33.07 (±2.40)
Diagonal stripes	38.17 (±2.48)	38.84 (±2.42)	38.20 (±2.59)	38.40 (±2.50)
Lukac	38.69 (±2.45)	41.24 (±2.37)	38.25 (±2.51)	39.39 (±2.44)
Lukac (90° rotated)	38.50 (±2.42)	40.96 (±2.34)	38.07 (±2.53)	39.18 (±2.43)
Pseudorandom	38.90 (±2.50)	40.12 (±2.41)	39.44 (±2.67)	39.49 (±2.53)
Pseudorandom (90° rotated)	38.87 (±2.49)	40.16 (±2.40)	39.51 (±2.64)	39.51 (±2.51)

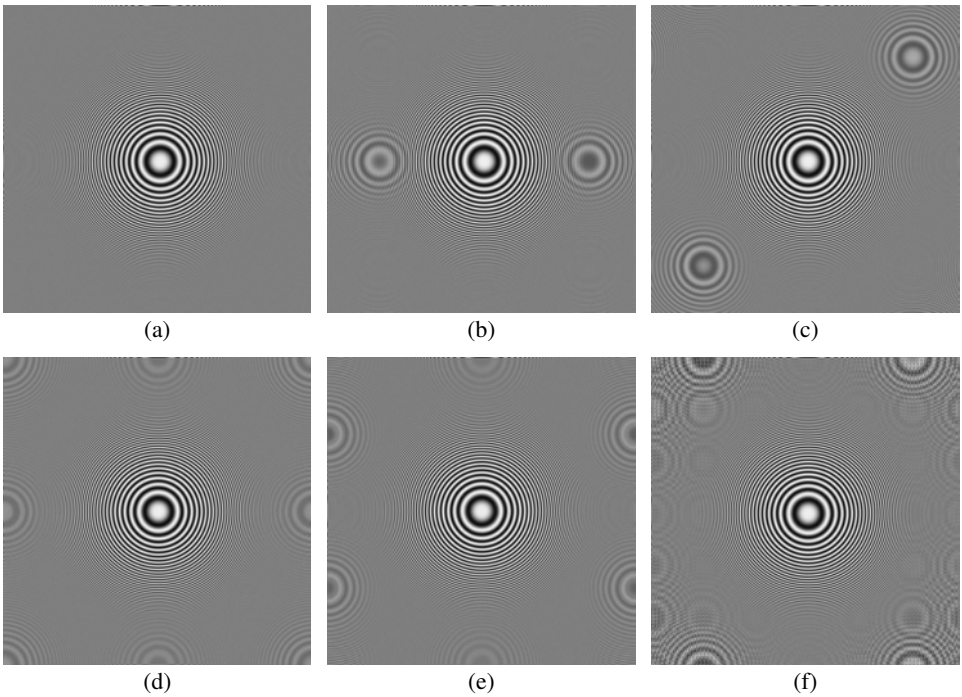


FIGURE 8.11 (See color insert.)

Results obtained using the 'CZP' image: (a) original image, (b-f) demosaicked images corresponding to (b) vertical stripe pattern, (c) diagonal stripe pattern, (d) Bayer pattern, (e) Lukac pattern, and (f) pseudorandom pattern.

terms of objective quality (the PSNR values) and in Figures 8.11, 8.12 and 8.13 for visual quality on the CZP image, the lighthouse image, and the Bahamas image.

Using PSNR as a quality criterion, we see that not all the CFAs give the same performance. As can be expected, the Bayer CFA gives better results than the diagonal stripes and the horizontal or vertical stripes. However, Lukac's CFA and the pseudorandom pattern give superior results than the widely used Bayer CFA. Two major properties intervene in the quality of a CFA: the frequencies of the chrominance and their orientations. Chrominance should actually be located the farthest from the center of the frequency spectrum to reduce the spectral overlap between luminance and modulated chrominances, while the vertical and horizontal directions are the most sensitive orientations for natural images [17].

As discussed earlier in this chapter, the Fourier spectrum illustrates the frequencies of chrominance and thus the potential amount of aliasing for the Bayer, Lukac, and stripe CFAs. These are all periodic patterns. However, that does not hold for the pseudorandom pattern. The Fourier spectrum actually shows information at every multiple of $1/n$ (with $n \times n$ being the size of the basic pattern) and these frequencies are present at a global scale, but not at a local scale. In order to know precisely which frequencies effectively alias, one should rather look at the demosaiced CZP images. This achromatic image is made of an isotropic sine function whose frequency is linearly modulated with spatial position. Consequently, we can visualize the *local* frequencies and know which ones cause spectral overlapping, as they will appear as false colors.

FIGURE 8.12

Results obtained using the 'lighthouse' image: (a) original image, (b-f) demosaicked images corresponding to (b) vertical stripe, (c) diagonal stripe, (d) Bayer, (e) Lukac, and (f) pseudorandom patterns.

Considering the frequency positions of the chrominance, the stripe CFAs are the worst, since they modulate at $1/3$ of the Nyquist frequency. Bayer and Lukac CFAs modulate at half of the Nyquist criterion. They are thus optimal concerning the frequency criteria. However, it is noteworthy that Lukac's CFA totally preserves the 0° orientation (or 90°, following the orientation of the pattern), whereas the Bayer CFA modulates in both directions. In Reference [17], the authors show that false colors arise mainly from these directions with the Bayer CFA. Lukac's CFA is thus superior considering this second criterion, and it gains 1dB in PSNR values for the B color plane. The pseudorandom pattern preserves both the 0° and 90° directions. Not all of the chrominance carriers are at half the Nyquist frequency, but those which are not have less energy. Hence, the pseudorandom CFA gives visually pleasant images, and it satisfies both the frequency and orientation criteria.

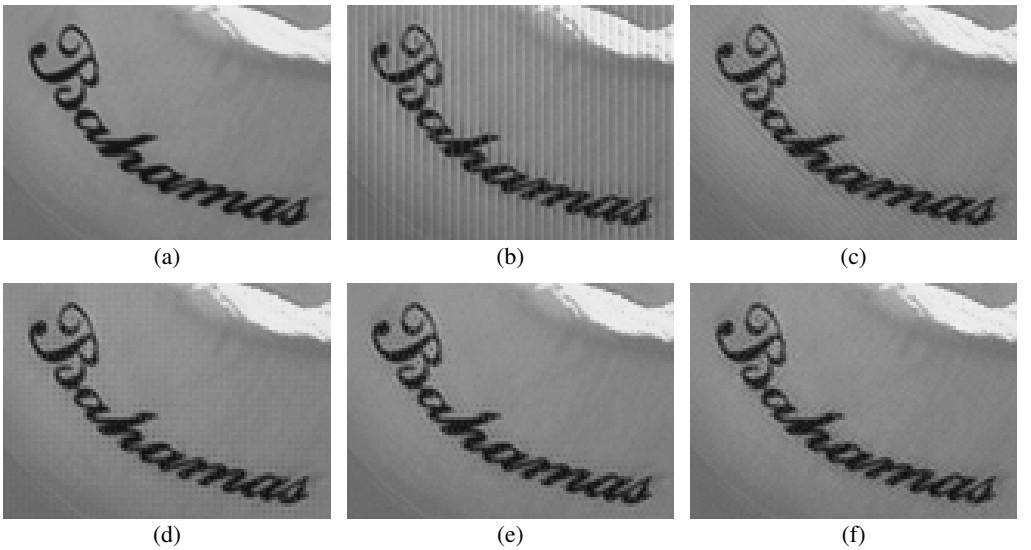


FIGURE 8.13

Results obtained using the 'bahamas' image: (a) original image, (b-f) demosaicked images corresponding to (b) vertical stripe, (c) diagonal stripe, (d) Bayer, (e) Lukac, and (f) pseudorandom patterns.

Zipper noise, a dual artifact to false colors, appears with these linear methods. It is caused by chrominance being interpreted as luminance. Changing the CFA pattern does not suppress this artifact, but changes its shape (Figure 8.13). Hence, zipper noise could be addressed by using an adaptive method exploiting the intra-plane correlation. Note that the stripe CFAs are extremely sensitive to zipper noise in regions of highly saturated colors.

8.4 Nonlinear and Adaptive Methods

Many demosaicking methods are nonlinear, as they exploit a nonlinear, adaptive and/or iterative process. It is impossible to describe all of these methods in this chapter, and the reader is referred to Reference [28].

The first nonlinear method proposed was based on pattern analysis [29]. The method first defines features like contours, corners, and bands based on the neighborhood of the pixel. Following these features, the interpolation differs to avoid the interpolation across

TABLE 8.2
PSNR values for several methods on the Bayer CFA.

Channel	LMMSE	Ref. [17]	Ref. [34]
R	38.53 (±2.67)	38.81 (±2.50)	38.78 (±2.59)
G	41.22 (±2.47)	42.82 (±2.50)	42.12 (±2.79)
B	37.25 (±2.59)	38.62 (±2.69)	38.68 (±2.62)

a contour, a corner, or a band. Inspired by this, many methods were later published that use an estimate of the image gradient to detect contours. Demosaicking is then performed along the contours rather than across them [30], [31], [32], [33]. We do not review all of these methods here, but rather focus on two nonlinear methods that use the spectral representation of the CFA image [17], [34] and give good objective results (Table 8.2).

8.4.1 Accurate Luminance Method

The zipper noise discussed in the previous section arises from the crosstalk of chrominance on luminance, i.e., when high frequencies of chrominance are interpreted as luminance. In Reference [34], a weighted interpolation of chrominance is performed, which takes edges into account. This allows retrieval of a sharp chrominance signal along the edges. Consequently, the chrominance recovers frequencies that go beyond the spectral boundary between luminance and chrominance, and hence the zipper noise is reduced.

For instance, at an R pixel with coordinates (x,y):

$$\psi_R(x,y) = \sum_{(i,j)\in D} w_{ij}\tilde{\psi}_R(x-i,y-j) \tag{8.34}$$

where $D = \{(-1,0),(0,1),(1,0),(0,-1)\}$ are the four closest neighbors, and where $\tilde{\Psi}_R$ is a rough estimate of ψ_R based on a linear interpolation of R components and a frequency selection of luminance. The weights w_{ij} depend on the gradient of the luminance and of the color channel R. The obtained ψ_R values are used to update the chrominance estimate at G pixels. Then another iteration of gradient-based estimation is performed. Note that this method resembles closely that of Reference [35], in which the green color plane represents luminance. Here the choice of luminance at G pixels is judicious because ψ_G vanishes at G pixels and luminance is thus least aliased. This method is able to correctly estimate luminance and chrominance within the luminance band. It suppresses zipper noise but sometimes fails in the chrominance band when luminance overflows. It is noteworthy that Lian's method has a complexity close to that of Reference [16] while having better visual quality and higher PSNR values.

8.4.2 Frequency Domain Method

This adaptive method suppresses false colors and artifacts. False colors are due to crosstalk of luminance on chrominance. The chrominance is then falsely estimated because it contains some achromatic components. In order to have a better estimate of chrominance, Dubois [17] exploits the redundancy of the chrominance components in the Fourier plane. We can make an analogy to the *spread spectrum* method in signal transmission, which consists of improving robustness by utilizing more frequencies than needed. In the case of a color mosaic, the chrominance components are modulated at different frequencies. If a chrominance frequency overlaps with a luminance frequency, it is possible that the same chrominance component — but at a different spectral location — will not overlap with a luminance frequency. This allows estimating a chrominance value without residual luminance, and thus potentially suppresses false colors. However, zipper noise is not reduced since chrominance is not interpolated while taking edges into account.

The choice of chrominance is driven by the amounts of energy E_x and E_y at the intersection between luminance and chrominance in horizontal and vertical directions:

$$\psi_2(x,y) = w_x \psi_2^h(x,y) + w_y \psi_2^v(x,y) \tag{8.35}$$

with $w_x = E_X/(E_X + E_Y)$ and $w_Y = 1 - w_X$.

This method correctly estimates chrominance within the chrominance band, eliminating the residual information of luminance. However, it fails at reducing zipper noise.

8.5 Conclusion

In this chapter we reviewed some linear methods for reconstructing full-color images from a mosaic of a single chromatic sample per spatial location. Using a model of chromatic sampling, we show that the representation in luminance and chrominance components also holds for CFA images. Subsampling does not affect the luminance component but rather the chrominance, which is modulated to high frequencies.

The LMMSE method described in this chapter allows objective comparisons of the different CFA arrangements published in the literature. It appears that the widely used Bayer CFA may not be the optimal one. Both the CFA proposed by Lukac and the pseudorandom pattern proposed here give better results in terms of minimizing false colors. For the Bayer CFA, however, sophisticated methods need to be applied in order to reduce these artifacts.

The best trade-off for the system comprising of the CFA and demosaicking modules will perhaps be the alliance between the best CFA that intrinsically reduces false color effects at data acquisition, and a cost-effective demosaicking method that will control the zipper noise effect. Such solutions are proposed in References [35] and [36], which both present a general demosaicking algorithm.

References

[1] K. Parulski and K.E. Spaulding, *Digital Color Image Handbook*, ch. Color image processing for digital cameras, G. Sharma (ed.), CRC Press, Boca Raton, FL, 2002, pp. 727–757.

[2] R. Lukac and K.N. Plataniotis, *Color Image Processing: Methods and Applications*, ch. Single-sensor camera image processing, R. Lukac and K.N. Plataniotis (eds.), Boca Raton, FL: CRC Press / Taylor & Francis, 2006, pp. 363–392.

[3] I. Newton, *Optics: Or, a Treatise of the Reflections, Refractions, Inflections and Colours of Light*. London, UK, 4th ed., 1730.

[4] T. Young, "On the theory of light and colours," *Philosophical Transactions*, vol. 92, pp. 12–48, July 1802.

[5] H. von Helmholtz, *Handbuch der Physiologischen Optik*. Leipzig: Leopold Voss, 1862.

[6] J. Nathans, D. Thomas, and D.S. Hogness, "Molecular genetics of human color vision: The genes encoding blue, green, and red pigments," *Science*, vol. 232, no. 4747, pp. 193–202, April 1986.

[7] D.A. Baylor, B.J. Nunn, and J.L. Schnapf, "Spectral sensitivity of the cones of the monkey Macaca fascicularis," *Journal of Physiology*, vol. 390, no. 1, pp. 145–160, September 1987.

[8] G. Wyszecki and W. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*. New York, New York, John Wiley & Sons, 1982.

[9] B.E. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 1976.

[10] P. Dillon, "Color imaging array," U.S. Patent 4 047 203, September 1977.

[11] R. Lukac and K.N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1260–1267, November 2005.

[12] R. Lyon and P. Hubel, "Eyeing the camera: Into the next century," in *Proceedings of the Color Imaging Conference*, Scottsdale, AZ, USA, November 2002, pp. 349–355.

[13] A. Roorda and D.R. Williams, "The arrangement of the three cone classes in the living human eye," *Nature*, vol. 397, no. 11, pp. 520–522, February 1999.

[14] E. Hering, *Zur Lehre vom Lichtsinn*. Vienna, Austria: 1878.

[15] L.M. Hurvich and D. Jameson, "An opponent-process theory of color vision," *Psychological Review*, vol. 64, no. 6, pp. 384–404, November 1957.

[16] D. Alleysson, S. Süsstrunk, and J. Hérault, "Linear color demosaicing inspired by the human visual system," *IEEE Transactions on Image Processing*, vol. 14, no. 4, pp. 439–449, April 2005.

[17] E. Dubois, "Frequency-domain methods for demosaicking of Bayer-sampled color images," *IEEE Signal Processing Letters*, vol. 12, no. 12, pp. 847–850, December 2005.

[18] E. Dubois, "Filter design for adaptive frequency-domain Bayer demosaicking," in *Proceedings of the IEEE International Conference on Image Processing*, Atlanta, GA, USA, October 2006, pp. 2705–2708.

[19] D.R. Cok, "Signal processing method and apparatus for sampled image signals," U.S. Patent 4 630 307, December 1986.

[20] D.R. Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal," U.S. Patent 4 642 678, February 1987.

[21] H.D. Crane, J.D. Peter, and E. Martinez-Uriegas, "Method and apparatus for decoding spatiochromatically multiplexed color image using predetermined coefficients," U.S. Patent 5 901 242, May 1999.

[22] D. Alleysson, S. Süsstrunk, and J. Hérault, "Color demosaicing by estimating luminance and opponent chromatic signals in the Fourier domain," in *Proceedings of the 10th IS&T/SID Color Imaging Conference*, Scottsdale, AZ, USA, November 2002, pp. 331–336.

[23] H.J. Trussell and R.E. Hartwig, "Mathematics for demosaicking," *IEEE Transactions on Image Processing*, vol. 11, no. 4, pp. 485–492, April 2002.

[24] D. Taubman, "Generalized Wiener reconstruction of images from colour sensor data using a scale invariant prior," in *IEEE International Conference on Image Processing*, Vancouver, BC, Canada, September 2000, vol. III, pp. 801–804.

[25] B. Chaix de Lavarène, D. Alleysson, and J. Hérault, "Practical implementation of LMMSE demosaicing using luminance and chrominance spaces," *Computer Vision and Image Understanding: Special Issue on Color Image Processing*, vol. 107, no. 1-2, pp. 3–13, July/August 2007.

[26] Y. Hel-Or, "The impulse responses of block shift-invariant systems and their use for demosaicing algorithms," in *Proceedings of the IEEE International Conference on Image Processing*, Genoa, Italy, September 2005, vol. 2, pp. 1006–1009.

[27] J. Portilla, D. Otaduy, and C. Dorronsoro, "Low-complexity linear demosaicing sing joint spatial-chromatic image statistics," in *Proceedings of the IEEE International Conference on Image Processing*, Genoa, Italy, September 2005, vol. I, pp. 61–64.

[28] B.K. Gunturk, J. Glotzbach, Y. Altunbazak, R.W. Schafer, and R.M. Mersereau, "Demosaicking: Color filter array interpolation in single-chip digital cameras," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 44–54, January 2005.

[29] D.R. Cok, "Signal processing method and apparatus for sampled image signals," U.S. Patent 4 630 307, December 1986.

[30] R.H. Hibbard, "Apparatus and method for adaptively interpolating a full color image utilizing luminance gradients," U.S. Patent 5 382 976, January 1995.

[31] C.A. Laroche and M.A. Prescott, "Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients," U.S. Patent 5 373 322, December 1994.

[32] J.F. Hamilton and J.E. Adams, "Adaptive color plane interpolation in single sensor color electronic camera," U.S. Patent 5 629 734, May 1997.

[33] R. Kimmel, "Demosaicing: Image reconstruction from color samples," in *IEEE Transactions on Image Processing*, vol. 8, no. 9, pp. 1221–1228, September 1999.

[34] N. Lian, L. Chang, and Y.P. Tan, "Improved color filter array demosaicking by accurate luminance estimation," in *IEEE International Conference on Image Processing*, Genoa, Italy, September 2005, vol. 1, pp. 41–44.

[35] R. Lukac and K.N. Plataniotis, "Universal demosaicking for imaging pipelines with an RGB color filter array," *Pattern Recognition*, vol. 38, no. 11, pp. 2208–2212, November 2005.

[36] B. Chaix de Lavarène, D. Alleysson, and J. Hérault, "Efficient demosaicing through recursive filtering," in *Proceedings of the IEEE International Conference on Image Processing*, San Antonio, TX, USA, September 2007, vol. II, pp. 189–192.

9

Color Filter Array Image Analysis for Joint Demosaicking and Denoising

Keigo Hirakawa

9.1 Introduction

Noise is among the worst artifacts that affect the perceptual quality of the output from a digital camera (see Chapter 1). While cost-effective and popular, single-sensor solutions to camera architectures are not adept at noise suppression. In this scheme, data are typically obtained via a spatial subsampling procedure implemented as a color filter array (CFA), a physical construction whereby each pixel location measures the intensity of the light corresponding to only a single color [1], [2], [3], [4], [5]. Aside from undersampling, observations made under noisy conditions typically deteriorate the estimates of the full-color image in the reconstruction process commonly referred to as *demosaicking* or *CFA interpolation* in the literature [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. A typical CFA scheme involves the canonical color triples (i.e., red, green, blue), and the most prevalent arrangement called Bayer pattern is shown in Figure 9.1b.

As the general trend of increased image resolution continues due to prevalence of multimedia, the importance of interpolation is de-emphasized while the concerns for computational efficiency, noise, and color fidelity play an increasingly prominent role in the decision making of a digital camera architect. For instance, the interpolation artifacts become less noticeable as the size of the pixel shrinks with respect to the image features, while the

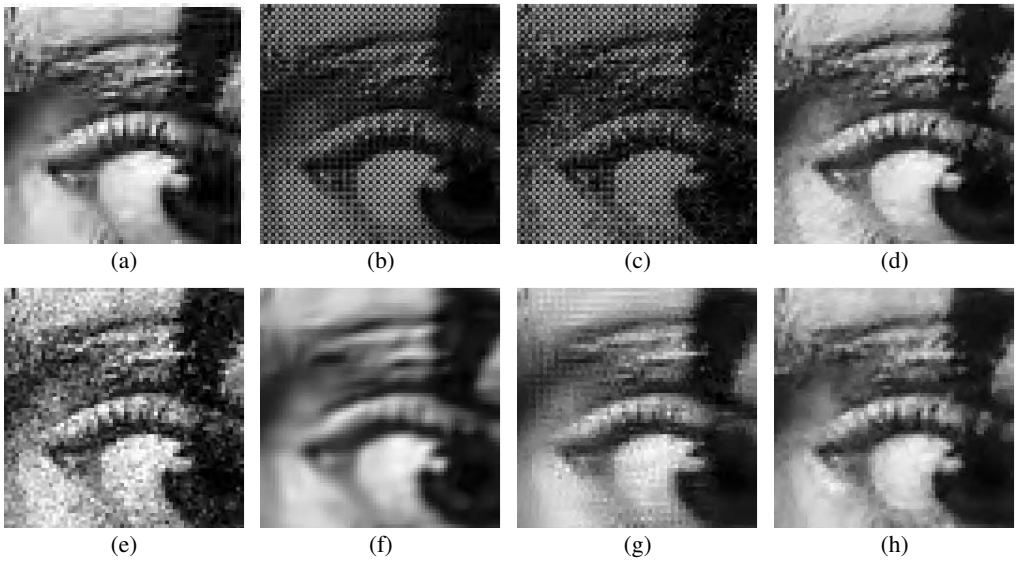


FIGURE 9.1 (See color insert.)

Zoomed portion of the *Clown* image: (a) original color image, (b) color version of ideal CFA image, (c) color version of noisy CFA image, (d) demosaicking the ideal CFA image, (e) demosaicking the noisy CFA image, (f) demosaicking the noisy CFA image followed by denoising, (g) denoising the noisy CFA image followed by demosaicking, and (h) joint denoising and demosaicking of the noisy CFA image.

decreased dimensionality of the pixel sensors on the complementary metal oxide semiconductor (CMOS) and charge coupled device (CCD) sensors make the pixels more susceptible to noise. Photon-limited influences are also evident in low-light photography, ranging from a specialty camera for precision measurement to indoor consumer photography.

Sensor data, which can be interpreted as subsampled or incomplete image data, undergo a series of image processing procedures in order to produce a digital photograph. Refer to Chapters 1 and 3 for details. However, these same steps may amplify noise introduced during image acquisition. Specifically, the demosaicking step is a major source of conflict between the image processing pipeline and image sensor noise characterization because the interpolation methods give high priority to preserving the sharpness of edges and textures. In the presence of noise, noise patterns may form false edge structures, and therefore the distortions at the output are typically correlated with the signal in a complicated manner that makes noise modelling mathematically intractable. Thus, it is natural to conceive of a rigorous tradeoff between demosaicking and image denoising.

For better illustration, Figure 9.1a shows a typical color image. Suppose we simulate the noisy sensor observation by subsampling this image according to a CFA pattern (Figure 9.1b) and corrupting with noise (Figure 9.1c). While state-of-the-art demosaicking methods such as the ones in [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] do an impressive job in estimating the full-color image given ideal sensor data (Figure 9.1d), the interpolation may also amplify the noise in the sensor measurements, as demonstrated in Figure 9.1e. The state-of-the-art denoising methods applied to Figure 9.1f yield unsatisfactory results (Figure 9.1g), suggesting a lack of coherent strategy to address interpolation

and noise issues jointly. For comparison, the output from a joint demosaicking and denoising method [17] is shown in Figure 9.1h, clearly demonstrating the advantages.

In this chapter, the problem of estimating the complete noise-free image signal of interest given a set of incomplete observations of pixel components that are corrupted by noise is approached statistically from a point of view of Bayesian statistics, that is modelling of the various quantities involved in terms of priors and likelihood. The three design regimes that will be considered here can be thought of as simultaneous interpolation and image denoising, though this chapter has a wider scope in the sense that modelling the image signal, missing data, and the noise process explicitly yield insight into the interplay between the noise and the signal of interest. The chapter is not intended to comprise detailed step-by-step instructions of how to estimate a complete noise-free image; rather we present a theoretical basis for generalizing the image signal models to the noisy subsampled case, and propose major building blocks for manipulating such data. The author feels that leading the discussion in this manner is most effective, as it allows flexibility in the choice of models.

There are a number of advantages to the proposed estimation schemes over the obvious alternative, which is the serial concatenation of the independently designed interpolation and image denoising algorithms. For example, the inherent image signal model assumptions underlying the interpolation procedure may differ from those of the image denoising. This discrepancy is not only contradictory and thus inefficient, but also triggers mathematically intractable interactions between mismatched models. Specifically, interpolating distorted image data may impose correlation structures or bias to the noise and image signal in an unintended way. Furthermore, a typical image denoising algorithm assumes a statistical model for natural images, not that of the output of interpolated image data. While grayscale and color image denoising techniques have been suggested [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], removing noise after demosaicking, however, is impractical. Likewise, although many demosaicking algorithms developed in the recent years yield impressive results in the absence of sensor noise, the performance is less than desirable in the presence of noise.

In this chapter, we investigate the problem of estimating a complete color image from the noisy undersampled signal using spectral and wavelet analysis of the noisy sensor data. In Section 9.2, we characterize the noise corresponding to CMOS and CCD sensors and evaluate it with respect to human visual system sensitivities and current image denoising techniques. Section 9.3 identifies the structure in the loss of information due to sampling and noise by examining the sensor data in the Fourier domain, and motivates a unified approach to interpolation and denoising. To exploit the local aliasing structures, Section 9.4 refines the spectral analysis of sensor data using time-frequency analysis. Conditioned on the signal image model, we propose three frameworks for estimating the complete noise-free image via the manipulation of noisy subsampled data. In Section 9.5, we discuss the design of a spatially-adaptive linear filter whose stop-band annihilates color artifacts and whose pass-band suppresses noise. Section 9.6 demonstrates the modelling of noisy subsampled color images in the wavelets domain using a statistical missing data formulation. As outlined in Section 9.7, however, it is possible to estimate the wavelet coefficients corresponding to the desiderata from the wavelet coefficients of the sensor data. This section presents example output images obtained using the techniques presented in this chapter. Finally, concluding remarks are listed in Section 9.8.

9.1.1 A Comment About Model Assumptions

The wavelet-based statistical models for image signals play a dominant part in the image denoising literature. In this paradigm, wavelet coefficients corresponding to image signals exhibit a heavy-tailed distribution behavior, motivating the use of Laplacian distribution, Student's t-distribution, and Gaussian mixtures, to name a few. These heavy-tailed priors can be written as a continuous mixture of Gaussian with the general form,

$$x|q \sim \mathcal{N}(\mu_x, \sigma_x^2/q),$$

where μ_x and σ_x^2 are the mean and variance parameters of a random variable x, and $q \neq 0$ is an augmented random variable with its own distribution specific to the choice of heavy-tail. Thus, x is conditionally normal; conditioned on q, its posterior distribution can largely be manipulated with second-order statistics. Alternatives to wavelet-based models include image patches [32], principal components [33], and anisotropic diffusion equations [28]. Many of them make use of the sum of (sometimes spatially-adaptive) outer-products of vectorized pixel neighborhoods, which is the deterministic-counterpart to the pixel-domain second-order statistics.

The intentions of this chapter, as stipulated previously, are to provide tools for analyzing and manipulating subsampled data in a way that is relevant to the CFA image. Rather than reinvent signal models for subsampled image data, we choose to work with statistical or deterministic models for a *complete* image data. In doing so, we inherit a rich literature in image modelling that has been shown to work well for image denoising, interpolation, segmentation, compression, and restoration. Furthermore, the discussion that follows is intentionally decoupled from a *particular choice* of image signal model. Instead, *conditioned* on the complete image model, the primary focus of the discussions will be on making the necessary changes amenable to the direct manipulation of the CFA image.

Specifically, the theoretical frameworks for analyzing subsampled data below are developed in terms of second-order statistics of complete image data. By taking the expectation over the conditionals in the posterior ($E[x] = E\big[E[x|q]\big]$ in the example above, where $x|q$ in the inner expectation is normal) one can generalize the estimator derived for the multivariate normal to the heavy-tailed distribution, as in the case of Bayesian estimators. Alternatively, replacing the second-order statistics with the sum of outer-products would yield the deterministic extension of the CFA image processing. In any case, the technical frameworks presented below are nonrestrictive and compatible with a wide range of assumed models, allowing for the flexibility in selecting a model best suited for the computational and image quality requirements of the application.

9.1.2 Terminologies and Notational Conventions

Because there are several technical terms used in this chapter that sound similar but have different meanings, we would like to clarify their definitions. The term *color filter* refers to a physical device placed over photosensitive elements called pixel sensors. It yields a color coding by cutting out electromagnetic radiations of specified wavelengths. This is not to be confused with a *filter*, or convolution filtering realized by taking a linear combination of nearby pixel or sensor values. Likewise, given a two-dimensional signal, terminologies

such as frequency and spectrum are to be interpreted in the context of two-dimensional Fourier transforms and not in the sense of colorimetry.

In this chapter, all image signals are assumed to be discrete (or post-sampling). For notational simplicity, plain characters (e.g., x) represent a singleton, whereas bold-face characters (e.g., $\boldsymbol{x}$) represent a vector or a matrix. An arrow over a character symbolizes a vectorization; that is, $\vec{x}$ is a re-arrangement of $x(\cdot)$ into a vector form. Other conventions are summarized below for bookkeeping, but their formal definitions will be made explicit in the sequel:

$\boldsymbol{n} \in \mathbb{Z}^2$	pixel/sample location index
$\boldsymbol{x} : \mathbb{Z}^2 \to \mathbb{R}^3$	signal-of-interest, ideal (noise-free) color image; $\boldsymbol{x} = [x_1, x_2, x_3]^T$ are the RGB triples
$\boldsymbol{\epsilon} : \mathbb{Z}^2 \to \mathbb{R}^3$	noise for $\boldsymbol{x}$
$\boldsymbol{c} : \mathbb{Z}^2 \to \{0,1\}^3$	color filter coding indicator
$\ell : \mathbb{Z}^2 \to \mathbb{R}$	monochromatic or approximate luminance image, $\ell = \frac{1}{4}x_1 + \frac{1}{2}x_2 + \frac{1}{4}x_3$
$\alpha : \mathbb{Z}^2 \to \mathbb{R}$	color difference or approximate chrominance image, $\alpha = x_1 - x_2$
$\beta : \mathbb{Z}^2 \to \mathbb{R}$	color difference or approximate chrominance image, $\beta = x_3 - x_2$
$y : \mathbb{Z}^2 \to \mathbb{R}$	ideal (noise-free) sensor data or CFA image, $y(\boldsymbol{n}) = \boldsymbol{c}^T(\boldsymbol{n})\boldsymbol{x}(\boldsymbol{n})$
$\varepsilon : \mathbb{Z}^2 \to \mathbb{R}$	noise for y
$z : \mathbb{Z}^2 \to \mathbb{R}$	noisy sensor data, $z = y + \varepsilon$
$\boldsymbol{g} : \mathbb{Z}^2 \times \mathbb{Z}^2 \to \mathbb{R}^3$	spatially-adaptive filter coefficients
$h_0, h_1, f_0, f_1 : \mathbb{Z} \to \mathbb{R}$	one-dimensional impulse responses to convolution filters used in filterbank

In the above, the elements in the vector $\boldsymbol{x}(\boldsymbol{n}) = [x_1(\boldsymbol{n}), x_2(\boldsymbol{n}), x_3(\boldsymbol{n})]^T$ are interpreted as the red, green, blue pixel component values, respectively, though the results established in this chapter are equally applicable in other color coding schemes. The luminance-chrominance representation of a color image, $[\ell(\boldsymbol{n}), \alpha(\boldsymbol{n}), \beta(\boldsymbol{n})]$, is an invertible linear transformation of $\boldsymbol{x}(\boldsymbol{n})$. The symbols $x : \mathbb{Z} \to \mathbb{R}$ and $\varepsilon : \mathbb{Z} \to \mathbb{R}$ are also occasionally used for a generic (nondescriptive) signal and noise, respectively. Singleton functions $x(\boldsymbol{n})$ and $\varepsilon(\boldsymbol{n})$ are used interchangeably with $\boldsymbol{x}(\boldsymbol{n})$ and $\boldsymbol{\epsilon}(\boldsymbol{n})$ to generalize results to the multivariate case, respectively.

In addition, given a two-dimensional function $x : \mathbb{Z}^2 \to \mathbb{R}$, its Fourier transform is denoted by $\hat{x}(\boldsymbol{\omega})$, where, in the two-dimensional case, $\boldsymbol{\omega} = [\omega_0, \omega_1]^T \in \mathbb{R}^2$ is the modulo-2π frequency index. Similarly, let $\boldsymbol{i} \in \{0, 1, \ldots, I\}^2$ be the subband index for the $(I+1)^2$-level (separable) two-dimensional filterbank decomposition, where a smaller index value corresponds to low-frequency channel. Then $w_{\boldsymbol{i}}^x(\boldsymbol{n})$ is the filterbank (or wavelet packets) coefficient at the $\boldsymbol{i}$-th subband, $\boldsymbol{n}$-th spatial location corresponding to the signal $x(\boldsymbol{n})$.

9.2 Noise Model

In order to design an effective image denoising system, it is important to characterize the noise in an image sensor. The CMOS photodiode active pixel sensor typically uses a photodiode and three transistors, all major sources of noise [34]. The CCD sensors rely on the electron-hole pair that is generated when a photon strikes silicon [35]. While a detailed investigation of the noise source is beyond the scope of this chapter, studies suggest that $z : \mathbb{Z}^2 \to \mathbb{R}$, the number of photons encountered during an integration period (duration between resets), is a Poisson process $\mathscr{P}_y$:

$$p\big(z(\boldsymbol{n})\big|y(\boldsymbol{n})\big) = \frac{e^{-y(\boldsymbol{n})}y(\boldsymbol{n})^{z(\boldsymbol{n})}}{z(\boldsymbol{n})!},$$

where $\boldsymbol{n} \in \mathbb{Z}^2$ is the pixel location index, and $y(\boldsymbol{n})$ is the expected photon count per integration period at location $\boldsymbol{n}$, which is linear with respect to the intensity of the light. Note $E\big[z(\boldsymbol{n})\big|y(\boldsymbol{n})\big] = y(\boldsymbol{n})$ and $E\Big[z^2(\boldsymbol{n}) - E\big[z(\boldsymbol{n})\big|y(\boldsymbol{n})\big]^2\Big|y(\boldsymbol{n})\Big] = y(\boldsymbol{n})$. Then, as the integration period increases, $p(z(\boldsymbol{n})|y(\boldsymbol{n}))$ converges weakly to $\mathscr{N}\big(y(\boldsymbol{n}), y(\boldsymbol{n})\big)$, or

$$z(\boldsymbol{n}) \approx y(\boldsymbol{n}) + \sqrt{y(\boldsymbol{n})}\varepsilon(\boldsymbol{n}), \tag{9.1}$$

where $\varepsilon \overset{\text{i.i.d.}}{\sim} \mathscr{N}(0,1)$ is independent of y. This approximation is justifiable via a straightforward application of central limit theorem to the binomial distribution. The noise term, $\sqrt{y(\boldsymbol{n})}\varepsilon(\boldsymbol{n})$ is commonly referred to as the *shot noise*.

In practice, the photodiode charge (e.g., photodetector readout signal) is assumed proportional to $z(\boldsymbol{n})$, thus we interpret $y(\boldsymbol{n})$ and $z(\boldsymbol{n})$ as the ideal and noisy sensor data at pixel location $\boldsymbol{n}$, respectively. For a typical consumer-grade digital camera, the approximation in Equation 9.1 is reasonable. The significance of Equation 9.1 is that the signal-to-noise ratio improves for a large value of $y(\boldsymbol{n})$ (e.g., outdoor photography), while for a small value of $y(\boldsymbol{n})$ (e.g., indoor photography) the noise is severe. To make matters worse, human visual response to the light $y(\boldsymbol{n})$ is often modeled as $\sqrt[3]{y(\boldsymbol{n})}$, suggesting a heightened sensitivity to the deviation in the dark regions of the image. To see this, the perceived noise magnitude is proportional to:

$$\sqrt[3]{z(\boldsymbol{n})} - \sqrt[3]{y(\boldsymbol{n})} = \sqrt[3]{y(\boldsymbol{n}) + \sqrt{y(\boldsymbol{n})}\varepsilon(\boldsymbol{n})} - \sqrt[3]{y(\boldsymbol{n})},$$

which is a monotonically decreasing function with respect to $y(\boldsymbol{n})$ for a fixed value of $\varepsilon(\boldsymbol{n})$.

There have been some hardware solutions to the sensor noise problems. For example, the cyan-magenta-yellow (CMY) CFA pattern performs better in a noisy environment, as the quantum efficiency is more favorable for CMY as compared to RGB. That is, a CMY-based CFA allows more photons to penetrate through to the photosensitive element because the pigments used in it are considerably thinner than those of the RGB-based CFA. The disadvantage is that the photo-sensitivity wavelengths of the cyan, magenta, and yellow overlap considerably, and therefore the color space conversion from CMY to the RGB color space is an unstable operation. Today, the CMY-based CFAs are more readily used in video

cameras, since the frame-rate restricts the length of the integration period. Other circuit-based noise-reduction techniques include correlated double sampling. In this scheme, the pixel sensors are each sampled twice, first measuring the reset/amplifier noise alone, and second measuring the photon counts and the reset/amplifier noise combined. The difference of the two is presumed noise-free.

In reality, efforts to address signal-dependent noise in Equation 9.1 lag behind those of image interpolation and image denoising for additive white Gaussian noise (AWGN). A standard technique for working with signal-dependent noise is to apply an invertible nonlinear operator $\gamma(\cdot)$ on z such that signal and noise are (approximately) decoupled:

$$\gamma(z)|\gamma(y) \sim \mathcal{N}\left(\gamma(y), \sigma^2\right)$$

for some constant σ^2. Homomorphic filtering is one such operator designed with monotonically-increasing nonlinear pointwise function $\gamma : \mathbb{R} \to \mathbb{R}$, [36], [37]. The Haar-Fisz transform $\gamma : \mathbb{Z}^2 \times \mathbb{R} \to \mathbb{Z}^2 \times \mathbb{R}$ is a multiscale method that asymptotically decorrelates signal and noise [38], [39]. In any case, a signal estimation technique (assuming AWGN) is used to estimate $\gamma(y)$ given $\gamma(z)$, and the inverse transform $\gamma^{-1}(\cdot)$ yields an estimate of y. The advantage of this approach is the modularity of the design of $\gamma(\cdot)$ and the estimator. The disadvantage is that the signal model assumed for y may not hold for $\gamma(y)$ and the optimality of the estimator (e.g., minimum mean squared error estimator) in the new domain does not translate to optimality in the rangespace of y, especially when $\gamma(\cdot)$ significantly deviates from linearity.

An alternative to decorrelation is to approximate the noise standard deviation, $\sqrt{y(\boldsymbol{n})}$. The AWGN noise model is effectively a zero-th order Taylor expansion of the Poisson process; an affine noise model is the first order Taylor expansion of Equation 9.1 used in References [32] and [40]. In practice, these approximations yield acceptable performance because the CMOS sensors operate on a relatively limited dynamic range, giving validity to the Taylor assumption (when the expansion is centered about the midpoint of the operating range). The human visual system can also tolerate a greater degree of error in the brighter regions of the image, allowing for more accurate noise characterization for small values of y (at the cost of poorer characterization for higher rangespace of y). Alternatively, empirical methods that address signal-dependent noise take a two-step approach [21]. First, a crude estimate of the noise variance at each pixel location $\boldsymbol{n}$ is found; second, conditioned on this noise variance estimate, we assume that the signal is corrupted by signal-independent noise. A *piecewise* AWGN model achieves a similar approximation.

Methods that work with the posterior distribution of the coefficients of interests, such as Markov chain Monte Carlo and importance sampling, either have a slow convergence rate or require a large number of observations [41]. Emerging frameworks in Bayesian analysis for Poisson noise yield an asymptotic representation of the Poisson process in the wavelets domain, but the manipulation of data in this class of representation is extremely complicated [42].

For all the reasons above, it is clear that the estimation of the mean y given the Poisson process z is not a well-understood problem; and existing methods use variations of AWGN models to address the Poisson noise. Hence, while acknowledging inadequacies, we restrict

our attention to the AWGN problem,

$$z(\boldsymbol{n}) = y(\boldsymbol{n}) + \varepsilon(\boldsymbol{n}), \tag{9.2}$$

where $\varepsilon \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_\varepsilon^2)$.

9.3 Spectral Analysis of CFA Image

In this section, we take a closer look at the sampling scheme and the structure of aliasing induced by the Bayer color filter array illustrated in Figure 9.1b, [11], [17]. The estimation of missing pixel components given observed pixel components is generally an ill-posed problem. By assuming that the image signals are highly structured, however, we effectively assume that the signal-of-interest lives in a lower-dimensional subspace that can be represented by the subspace spanned by the color filter array. Thus, although the *loss of data* at the hardware interface is inevitable, the *loss of information* due to sampling may be limited. We will show that the Fourier analysis and aliasing serve as a measure of loss of information, and that they motivate joint modelling and manipulation of subsampled data and noise (which will subsequently be fine-tuned using locally adaptive schemes in Sections 9.5 to 9.7).

In a color image, such as one shown in Figure 9.1a, the image pixel $\boldsymbol{x}(\boldsymbol{n}) = [x_1(\boldsymbol{n}), x_2(\boldsymbol{n}), x_3(\boldsymbol{n})]^T$ at the position $\boldsymbol{n} \in \mathbb{Z}^2$ denotes a vectorial value, typically expressed in terms of RGB coordinates. Figure 9.2a is a grayscale version of Figure 9.1a. Visual inspection of the original color image and its corresponding red, green, and blue channels depicted in Figure 9.2b to Figure 9.2d, respectively, reveals that the decomposed color channels may contain redundant information with respect to edge and textural formation, reflecting the fact that the changes in color at the object boundary are secondary to the changes in intensity. It follows from the (de-)correlation of color content at high frequencies and is well accepted among the color image scientists that the difference images (e.g., red-green, blue-green) exhibit rapid spectral decay relative to monochromatic image signals (e.g., gray, red, green), and are therefore slowly-varying over spatial domain. See Figure 9.2e and Figure 9.2f. Such heuristic intuitions are further backed by human physiology — the contrast sensitivity function for the luminance channel in human vision is typically modelled with a much higher pass-band than that of the chrominance channels.

An alternative to spectral modelling strategy based on color-ratio has been studied [43], [44], [45], [46]. Assuming that objects are piecewise constant color, then the ratios between color components within an object are constant, even though the intensities of pixels may vary over space. In practice, however, the numerical stability of ratios is difficult to achieve, and the spatial variation of the intensity levels is not captured explicitly by this model. For these reasons, while acknowledging the merits of the color-ratio modelling strategy, the discussions in this chapter will be confined to the difference image modelling.

Let $\boldsymbol{c}(\boldsymbol{n}) = [c_1(\boldsymbol{n}), c_2(\boldsymbol{n}), c_3(\boldsymbol{n})]^T \in \{[1,0,0]^T, [0,1,0]^T, [0,0,1]^T\}$ be a CFA coding such that the noise-free sensor data can be written as an inner product, $y(\boldsymbol{n}) = \boldsymbol{c}^T(\boldsymbol{n})\boldsymbol{x}(\boldsymbol{n})$. Given

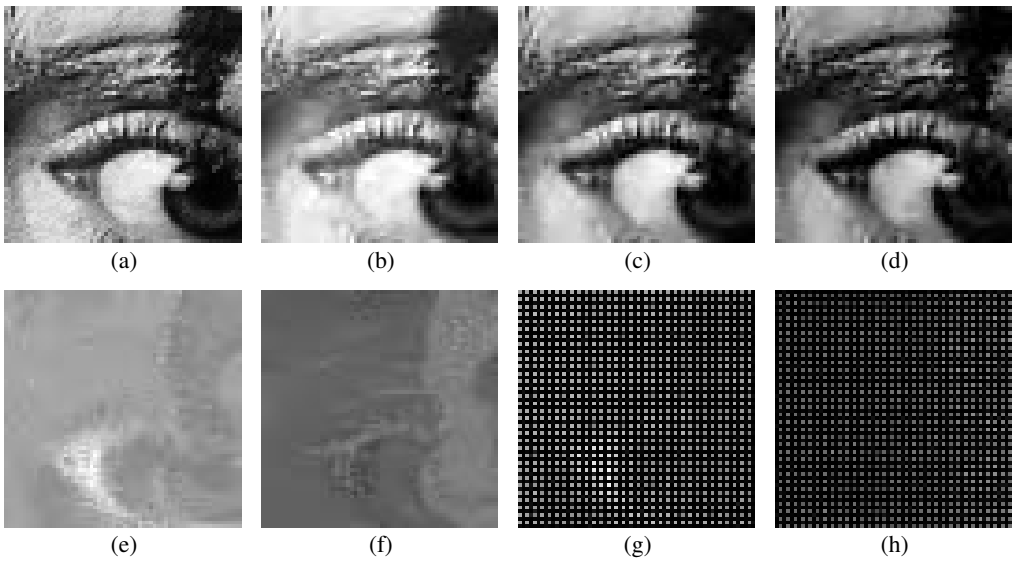


FIGURE 9.2

Zoomed portion of the *Clown* image: (a) gray-scale version of original color image, (b) decomposed red channel, (c) decomposed green channel, (d) decomposed blue channel, (e) difference image $x_1 - x_2$, (f) difference image $x_3 - x_2$, (g) subsampled version of $x_1 - x_2$, and (h) subsampled version of $x_3 - x_2$.

that it is a convex combination, we may then decompose $y(\boldsymbol{n})$ in the following manner:

$$\begin{aligned} y(\boldsymbol{n}) &= c_1(\boldsymbol{n})x_1(\boldsymbol{n}) + c_2(\boldsymbol{n})x_2(\boldsymbol{n}) + c_3(\boldsymbol{n})x_3(\boldsymbol{n}) \\ &= c_1(\boldsymbol{n})x_1(\boldsymbol{n}) + (1 - c_1(\boldsymbol{n}) - c_3(\boldsymbol{n}))x_2(\boldsymbol{n}) + c_3(\boldsymbol{n})x_3(\boldsymbol{n}) \\ &= c_1(\boldsymbol{n})(x_1(\boldsymbol{n}) - x_2(\boldsymbol{n})) + c_3(\boldsymbol{n})(x_3(\boldsymbol{n}) - x_2(\boldsymbol{n})) + x_2(\boldsymbol{n}) \\ &= c_1(\boldsymbol{n})\alpha(\boldsymbol{n}) + c_3(\boldsymbol{n})\beta(\boldsymbol{n}) + x_2(\boldsymbol{n}), \end{aligned} \tag{9.3}$$

where the difference images $\alpha(\boldsymbol{n}) = x_1(\boldsymbol{n}) - x_2(\boldsymbol{n})$ and $\beta(\boldsymbol{n}) = x_3(\boldsymbol{n}) - x_2(\boldsymbol{n})$ are crude approximations for the chrominance channels. In other words, the convex combination above can be thought of as the summation of $x_2(\boldsymbol{n})$ with the subsampled difference images, $c_1(\boldsymbol{n})\alpha(\boldsymbol{n})$ and $c_3(\boldsymbol{n})\beta(\boldsymbol{n})$; it is shown pictorially in Figure 9.2c, Figure 9.2g and Figure 9.2h, as their sum is equal to the sensor data in Figure 9.1b. It follows from the composition of the dyadic decimation and interpolation operators induced by the Bayer sampling pattern that $\hat{y}(\boldsymbol{\omega})$, the Fourier transform of sensor data $y(\boldsymbol{n})$, is a sum of $\hat{x_2}(\boldsymbol{\omega})$ and the spectral copies of $\hat{\alpha}(\boldsymbol{\omega})$ and $\hat{\beta}(\boldsymbol{\omega})$:

$$\begin{aligned} \hat{y}(\boldsymbol{\omega}) &= \hat{x_2}(\boldsymbol{\omega}) + \frac{1}{4}\Big((\hat{\alpha}+\hat{\beta})(\boldsymbol{\omega}) + (\hat{\alpha}-\hat{\beta})(\boldsymbol{\omega} - [\pi,0]^T) \\ &\quad + (\hat{\alpha}-\hat{\beta})(\boldsymbol{\omega} - [0,\pi]^T) + (\hat{\alpha}+\hat{\beta})(\boldsymbol{\omega} - [\pi,\pi]^T)\Big) \\ &= \hat{\ell}(\boldsymbol{\omega}) + \frac{1}{4}\Big((\hat{\alpha}-\hat{\beta})(\boldsymbol{\omega} - [\pi,0]^T) \\ &\quad + (\hat{\alpha}-\hat{\beta})(\boldsymbol{\omega} - [0,\pi]^T) + (\hat{\alpha}+\hat{\beta})(\boldsymbol{\omega} - [\pi,\pi]^T)\Big), \end{aligned} \tag{9.4}$$

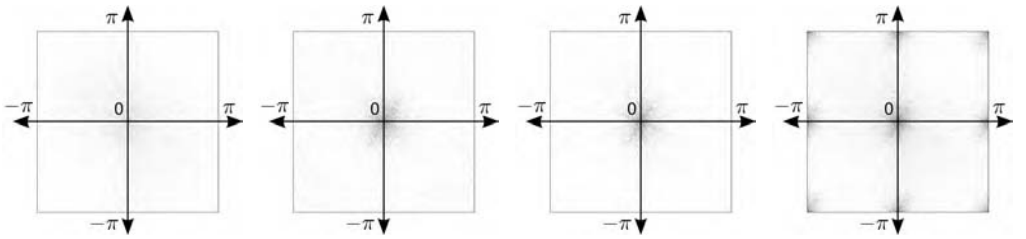


FIGURE 9.3

Log-magnitude two-dimensional spectra of: (a) $\hat{\ell}$, (b) $\hat{\alpha}$, (c) $\hat{\beta}$, and (d) $\hat{y}$. The spectra were obtained using the *Clown* image. The figure is color-coded to show contribution from each channel in figure (d): green for $\hat{\ell}$, red for $\hat{\alpha}$, blue for $\hat{\beta}$.

where, without loss of generality, the origin is fixed as $\boldsymbol{c}(0,0) = [1,0,0]^T$, and

$$\hat{\ell} = \hat{x}_2(\boldsymbol{\omega}) + \frac{1}{4}\hat{\alpha}(\boldsymbol{\omega}) + \frac{1}{4}\hat{\beta}(\boldsymbol{\omega}) = \frac{1}{4}\hat{x}_1(\boldsymbol{\omega}) + \frac{1}{2}\hat{x}_2(\boldsymbol{\omega}) + \frac{1}{4}\hat{x}_3(\boldsymbol{\omega}), \tag{9.5}$$

is a crude approximation to the luminance channel.

The representation of sensor data (Equation 9.4) in terms of luminance ℓ and difference images α and β is convenient because α and β are typically sparse in the Fourier domain. To see this, consider Figure 9.3, in which the log-magnitude spectra of a typical color image is shown. The high-frequency components, a well-accepted indicator for edges, object boundaries, and textures, are easily found in Figure 9.3a. In contrast, the spectra in Figure 9.3b and Figure 9.3b reveal that α and β are low-pass, which supports our earlier claim about the slowly-varying nature of the signals in Figure 9.2e and Figure 9.2f. It is typically easier to estimate a lower bandwidth signal from its sparsely subsampled versions (see Figure 9.2g and Figure 9.2h), since it is less subject to aliasing. The key observation that can be made in Equation 9.4, therefore, is that we expect a Fourier domain representation of sensor data similar to what is illustrated in Figure 9.3d — the spectral copies of $\hat{\alpha} - \hat{\beta}$ centered around $[\pi, 0]^T$ and $[0, \pi]^T$ overlap with the baseband $\hat{\ell}$, while $\hat{\alpha} + \hat{\beta}$ centered around $[\pi, \pi]^T$ remain aliasing-free.

Note that there exists no straightforward global strategy such that we recover unaliased $\hat{\ell}$ because both spectral copies centered around $[\pi, 0]^T$ and $[0, \pi]^T$ are aliased with the baseband $\hat{\ell}$. Dubois et al., however, emphasized that the local image features of the baseband, $\hat{\ell}$, exhibit a strong directional bias, and therefore either $(\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - [\pi, 0]^T)$ or $(\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - [0, \pi]^T)$ is locally recoverable from the sensor data [47]. This observation motivates nonlinear processing that is locally adaptive — in fact, most existing demosaicking methods can be reexamined from this perspective. Specifically, Figure 9.4 illustrates the presumed local aliasing pattern. The locally horizontal images suffer from aliasing between $\hat{\ell}$ and $(\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - [\pi, 0]^T)$ while we expect that $(\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - [0, \pi]^T)$ remains relatively intact. Conversely, locally vertical images suffer from aliasing between $\hat{\ell}$ and $(\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - [0, \pi]^T)$ while $(\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - [\pi, 0]^T)$ is clean. On a sidenote, locally diagonal image features, which are often ignored by the demosaicking algorithm designs, do not interfere with $(\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - [\pi, 0]^T)$ and $(\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - [0, \pi]^T)$, making the reconstruction of diagonal features a trivial task.

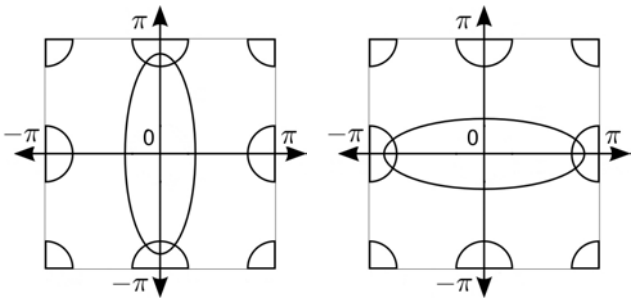


FIGURE 9.4

Presumed aliasing structure in local spectra, conditioned local image features of the surrounding. Images correspond to: (a) $\hat{y}$ given horizontal features, and (b) $\hat{y}$ given vertical features. Compare with Figure 9.3d.

Finally, let $z(\boldsymbol{n})$ be the noisy sensor data,

$$z(\boldsymbol{n}) = y(\boldsymbol{n}) + \varepsilon(\boldsymbol{n}) = c_1(\boldsymbol{n})\alpha(\boldsymbol{n}) + c_3(\boldsymbol{n})\beta(\boldsymbol{n}) + x_2(\boldsymbol{n}) + \varepsilon(\boldsymbol{n}), \tag{9.6}$$

where $\varepsilon \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_\varepsilon^2)$. Recall that Fourier transform is a unitary transformation — a spatially white noise in space domain remains uncorrelated in the frequency representation. It follows that the Fourier transform of a noisy observation is

$$\begin{aligned}\hat{z}(\boldsymbol{\omega}) = \hat{\ell}(\boldsymbol{\omega}) + \frac{1}{4}\Big(&(\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - [\pi, 0]^T) \\ &+ (\hat{\alpha} - \hat{\beta})(\boldsymbol{\omega} - (0, \pi)^T) + (\hat{\alpha} + \hat{\beta})(\boldsymbol{\omega} - [\pi, \pi]^T)\Big) + \hat{\varepsilon}(\boldsymbol{\omega}).\end{aligned}$$

In other words, the sensor data is the baseband luminance image $\hat{\ell}$ distorted by the noise $\hat{\varepsilon}$ and aliasing due to spectral copies of $\hat{\alpha}$ and $\hat{\beta}$, where $\hat{\varepsilon}$, $\hat{\alpha}$, and $\hat{\beta}$ are conditionally normal. A unified strategy to demosaicking and denoising, therefore, is to design an estimator that suppresses noise and attenuates aliased components simultaneously. We will see how this can be accomplished via a spatially-adaptive linear filter whose stop-band contains the spectral copies of the difference images and pass-band suppresses noise (Section 9.5).

9.4 Wavelet Analysis of CFA Image

In the previous section, we established the inadequacy of taking the global approach to CFA image processing. In this section, we develop a time-frequency analysis framework to exploit the local aliasing structures [17]. Specifically, image signals are highly nonstationary/inhomogeneous and thus an orthogonal filterbank (or wavelet packet) expansion for sparsely sampled signal would prove useful.

For simplicity, consider first a one-dimensional signal $x : \mathbb{Z} \to \mathbb{R}$. A one-level filterbank structure defined by filters $\{h_0, h_1, f_0, f_1\}$ is shown in Figure 9.5. It is a linear transformation composed of convolution filters and decimators. The channel containing the

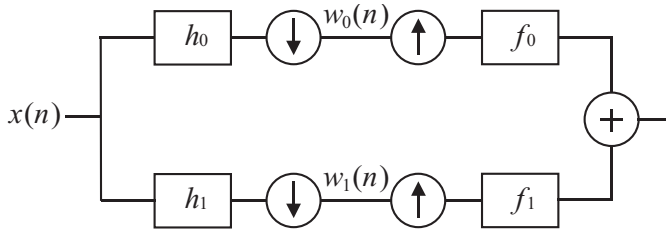


FIGURE 9.5

One-level filterbank structure.

low-frequency components is often called *approximation* (denoted $w_0^x(n)$), and the other containing the high-frequency components is referred to as the *detail* (denoted $w_1^x(n)$). The decomposition can be nested recursively to gain more precision in frequency. The approximation and detail coefficients from one-level decomposition can be analyzed in the Fourier domain as:

$$\hat{w}_i^x(\omega) = \frac{1}{2}\Big(\hat{h}_i\left(\tfrac{\omega}{2}\right)\hat{x}\left(\tfrac{\omega}{2}\right) + \hat{h}_i\left(\tfrac{\omega}{2}-\pi\right)\hat{x}\left(\tfrac{\omega}{2}-\pi\right)\Big),$$

where $i \in \{0,1\}$. With a careful choice of filters $\{h_0, h_1, f_0, f_1\}$, the original signal, $x(n)$ can be recovered exactly from the filterbank coefficients $w_0^x(n)$ and $w_1^x(n)$. To see this, consider the reconstruction of one-level filterbank, as in Figure 9.5. The transfer function of the system (or the reconstructed signal $x^{\text{rec}}(n)$) has the following form in the frequency domain:

$$\begin{aligned}\hat{x}^{\text{rec}}(\omega) &= \hat{f}_0(\omega)\hat{w}_0^x(2\omega) + \hat{f}_1(\omega)\hat{w}_1^x(2\omega)\\ &= \frac{1}{2}\Big(\hat{f}_0(\omega)\hat{h}_0(\omega) + \hat{f}_1(\omega)\hat{h}_1(\omega)\Big)\hat{x}(\omega)\\ &\quad + \frac{1}{2}\Big(\hat{f}_0(\omega)\hat{h}_0(\omega-\pi) + \hat{f}_1(\omega)\hat{h}_1(\omega-\pi)\Big)\hat{x}(\omega-\pi).\end{aligned}$$

In other words, the output is a linear combination of the filtered versions of the signal $\hat{x}(\omega)$ and a frequency-modulated signal $\hat{x}(\omega-\pi)$. The structure in Figure 9.5 is called a perfect reconstruction filterbank if

$$\begin{aligned}\hat{f}_0(\omega)\hat{h}_0(\omega) + \hat{f}_1(\omega)\hat{h}_1(\omega) &= 2\\ \hat{f}_0(\omega)\hat{h}_0(\omega-\pi) + \hat{f}_1(\omega)\hat{h}_1(\omega-\pi) &= 0.\end{aligned}$$

The filters corresponding to $\hat{x}(\omega)$ constitute a constant, whereas the filters corresponding to the aliased version are effectively a zero.

A large body of literature exists on designing a set of filters $\{h_0, h_1, f_0, f_1\}$ that comprise a perfect reconstruction filterbank [48]. For example, wavelet packets belong to a class of filterbanks arising from the factorizing filters satisfying the Nyquist condition (Smith-Barnwell [48]). In this case, the following are met by construction:

$$\begin{aligned}\hat{h}_1(\omega) &= -e^{-j\omega m}\hat{h}_0(-\omega-\pi)\\ \hat{f}_0(\omega) &= \hat{h}_1(\omega-\pi)\\ \hat{f}_1(\omega) &= -\hat{h}_0(\omega-\pi).\end{aligned} \tag{9.7}$$

In other words, h_1 is a *time-shifted, time-reversed,* and *frequency-modulated* version of h_0; and f_0 and f_1 are *time-reversed* versions of h_0 and h_1, respectively. Derivation of these filters is beyond of the scope of this chapter, and interested readers are referred to Reference [48] for details.

Define modulated signal and subsampled signal of $x(n)$, respectively, as

$$\begin{aligned} x_m(n) &= (-1)^n x(n) \\ x_s(n) &= \frac{1}{2}\big(x(n)+x_m(n)\big) = \begin{cases} x(n) & \text{for even } n \\ 0 & \text{for odd } n. \end{cases} \end{aligned}$$

To derive an explicit filterbank representation of $x_s(n)$, we are interested in characterizing the relationship between filterbank coefficients of $x(n)$ and $x_m(n)$. Let $w_0^{x_m}(n)$ and $w_1^{x_m}(n)$ be the approximation and detail coefficients of the one-level filterbank decomposition of $(-1)^n x(n)$. Then substituting into Equation 9.7 we obtain

$$\begin{aligned} \hat{w}_0^{x_m}(\omega) &= \frac{1}{2}\Big(\hat{h}_0\left(\tfrac{\omega}{2}\right)\hat{x}\left(\tfrac{\omega}{2}-\pi\right)+\hat{h}_0\left(\tfrac{\omega}{2}-\pi\right)\hat{x}\left(\tfrac{\omega}{2}\right)\Big) \\ &= \frac{1}{2}\Big(e^{-jm\frac{\omega}{2}}\hat{h}_1\left(-\tfrac{\omega}{2}-\pi\right)\hat{x}\left(\tfrac{\omega}{2}-\pi\right)+e^{-jm\left(\frac{\omega}{2}-\pi\right)}\hat{h}_1\left(-\tfrac{\omega}{2}\right)\hat{x}\left(\tfrac{\omega}{2}\right)\Big) \\ &= \frac{e^{-jm\frac{\omega}{2}}}{2}\Big(\hat{h}_1^*\left(\tfrac{\omega}{2}-\pi\right)\hat{x}\left(\tfrac{\omega}{2}-\pi\right)-\hat{h}_1^*\left(\tfrac{\omega}{2}\right)\hat{x}\left(\tfrac{\omega}{2}\right)\Big) \end{aligned}$$

$$\begin{aligned} \hat{w}_1^{x_m}(\omega) &= \frac{1}{2}\Big(\hat{h}_1\left(\tfrac{\omega}{2}\right)\hat{x}\left(\tfrac{\omega}{2}-\pi\right)+\hat{h}_1\left(\tfrac{\omega}{2}-\pi\right)\hat{x}\left(\tfrac{\omega}{2}\right)\Big) \\ &= \frac{1}{2}\Big(-e^{-jm\frac{\omega}{2}}\hat{h}_0\left(-\tfrac{\omega}{2}-\pi\right)\hat{x}\left(\tfrac{\omega}{2}-\pi\right)-e^{-jm\left(\frac{\omega}{2}-\pi\right)}\hat{h}_0\left(-\tfrac{\omega}{2}\right)\hat{x}\left(\tfrac{\omega}{2}\right)\Big) \\ &= \frac{e^{-jm\frac{\omega}{2}}}{2}\Big(-\hat{h}_0^*\left(\tfrac{\omega}{2}-\pi\right)\hat{x}\left(\tfrac{\omega}{2}-\pi\right)+\hat{h}_0^*\left(\tfrac{\omega}{2}\right)\hat{x}\left(\tfrac{\omega}{2}\right)\Big), \end{aligned}$$

where m is an odd integer, and $*$ denotes the complex conjugation. A subtle but important detail of the equations above is that if the approximation and detail coefficients of $x(n)$ were computed using $h_0(-n-m)$ and $h_1(-n-m)$ instead of $h_0(n)$ and $h_1(n)$, these coefficients behave exactly like the *detail* ($w_1^{x_m}(n)$) and *approximation* ($w_0^{x_m}(n)$) coefficients for $(-1)^n x(n)$, respectively (note the *reversed* ordering of detail and approximation). It is straightforward to verify that if $\{h_0(n), h_1(n)\}$ comprise perfect reconstruction filterbank, then $\{h_0(-n-m), h_1(-n-m)\}$ constitute a legitimate perfect reconstruction filterbank as well (we will refer to the latter as the time-reversed filterbank). Reversal of coefficients is illustrated in Figure 9.6 — the systems in Figure 9.6a and Figure 9.6b are equivalent.

Restricting our attention to the Haar decomposition for the rest of discussion and fixing $m = 1$, we have that $h_0(n) = h_0(-n-1)$ and $h_1(n) = -h_1(-n-1)$ and the approximation coefficient of $(-1)^n x(n)$ is exactly equal to the detail coefficient of $x(n)$ by construction, and vice-versa — i.e., $w_0^{x_m}(n) = w_1^x(n)$ and $w_1^{x_m}(n) = w_0^x(n)$. It follows that the multi-level filterbank decomposition of $(-1)^n x(n)$ is equivalent to the time-reversed filterbank decomposition of $x(n)$, but with the reversed ordering of low-to-high frequency coefficients. This reversed-order filterbank can be used to derive the filterbank representation of $x_s(n)$.

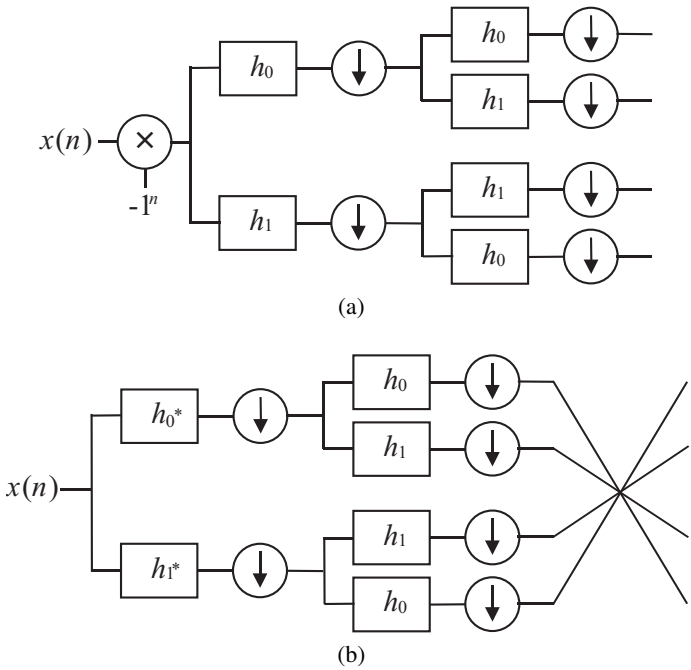


FIGURE 9.6

Two equivalent filterbanks for $x_m(n) = (-1)^n x(n)$: (a) filterbank transform of x_m, (b) reversed-order filterbank transform of x. Here, * indicates time-reversed filter coefficients.

Specifically, let $w_0^{x_s}(n)$ and $w_1^{x_s}(n)$ be the approximation and detail coefficients of the one-level filterbank decomposition of $x_s(n)$. Then

$$w_0^{x_s}(n) = w_0^{1/2(x+x_m)}(n) = \frac{1}{2}\left(w_0^x(n) + w_0^{x_m}(n)\right) = \frac{1}{2}\left(w_0^x(n) + w_1^x(n)\right)$$
$$w_1^{x_s}(n) = w_1^{1/2(x+x_m)}(n) = \frac{1}{2}\left(w_1^x(n) + w_1^{x_m}(n)\right) = \frac{1}{2}\left(w_1^x(n) + w_0^x(n)\right) = w_0^{x_s}(n).$$

Now, update the definition of w_i^x to mean the i-th subband of $(I+1)$-level filterbank decomposition. Then by recursion, we have a general form

$$w_i^{x_s}(n) = \frac{1}{2}\left(w_i^x(n) + w_{I-i}^x(n)\right). \tag{9.8}$$

Also see Figure 9.7. Equation 9.8 should not come as a surprise, as it is analogous to the Fourier domain aliasing where the high frequency component is summed to the low. Similar analysis for x_s can be performed for nonHaar decompositions, but omitted here for simplicity.

Extending to two-dimensional signals, let us show the decomposition of CFA image in the separable wavelet packet domain. Let $w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}), w_{\boldsymbol{i}}^{\alpha}(\boldsymbol{n}), w_{\boldsymbol{i}}^{\beta}(\boldsymbol{n})$ be the filterbank coefficients corresponding to $\ell(\boldsymbol{n}), \alpha(\boldsymbol{n}), \beta(\boldsymbol{n})$, respectively, where $\boldsymbol{i} = [i_0, i_1]^T \in \{0,1,\ldots,I\}^2$ indexes the horizontal and the vertical filterbank channels, respectively. As before, assume without loss of generality that $\boldsymbol{c}(0,0) = [1,0,0]^T$. In order to apply the filterbank analysis to the

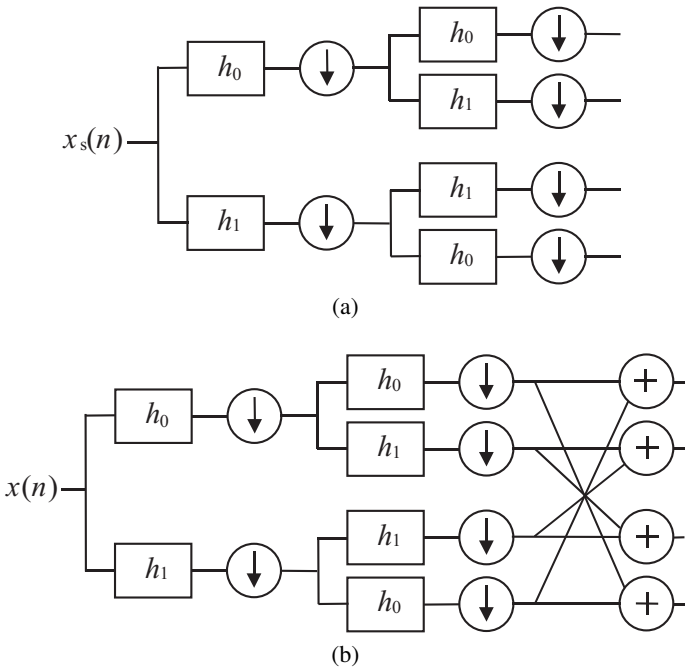


FIGURE 9.7

Two equivalent filterbanks for $x_s = \frac{1}{2}(x + x_m)$; up to multiplicative constant 2: (a) filterbank transform of x_s, and (b) ordinary and reversed-order filterbank transform of x. Here, we assume the Haar decomposition.

sensor data, we re-write $y(\boldsymbol{n})$ in the following manner:

$$\begin{aligned} y(\boldsymbol{n}) &= x_2(\boldsymbol{n}) + c_1(\boldsymbol{n})\alpha(\boldsymbol{n}) + c_3(\boldsymbol{n})\beta(\boldsymbol{n}) \\ &= x_2(\boldsymbol{n}) + \left(1 + (-1)^{n_0} + (-1)^{n_1} + (-1)^{n0+n1}\right)\frac{\alpha(\boldsymbol{n})}{4} \\ &\quad + \left(1 + (-1)^{n_0+1} + (-1)^{n_1+1} + (-1)^{n0+n1}\right)\frac{\beta(\boldsymbol{n})}{4}, \end{aligned}$$

and its corresponding filterbank representation:

$$\begin{aligned} w_{\boldsymbol{i}}^{y}(\boldsymbol{n}) &= w_{\boldsymbol{i}}^{x_2}(\boldsymbol{n}) + \frac{1}{4}\left(w_{\boldsymbol{i}}^{\alpha}(\boldsymbol{n}) + w_{(i_0,I-i_1)}^{\alpha}(\boldsymbol{n}) + w_{(I-i_0,i_1)}^{\alpha}(\boldsymbol{n}) + w_{(I-i_0,I-i_1)}^{\alpha}(\boldsymbol{n})\right) \\ &\quad + \frac{1}{4}\left(w_{\boldsymbol{i}}^{\beta}(\boldsymbol{n}) - w_{(i_0,I-i_1)}^{\beta}(\boldsymbol{n}) - w_{(I-i_0,i_1)}^{\beta}(\boldsymbol{n}) + w_{(I-i_0,I-i_1)}^{\beta}(\boldsymbol{n})\right) \\ &= w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) + \frac{1}{4}\left(w_{(i_0,I-i_1)}^{\alpha}(\boldsymbol{n}) + w_{(I-i_0,i_1)}^{\alpha}(\boldsymbol{n}) + w_{(I-i_0,I-i_1)}^{\alpha}(\boldsymbol{n})\right) \\ &\quad \frac{1}{4}\left(-w_{(i_0,I-i_1)}^{\beta}(\boldsymbol{n}) - w_{(I-i_0,i_1)}^{\beta}(\boldsymbol{n}) + w_{(I-i_0,I-i_1)}^{\beta}(\boldsymbol{n})\right), \end{aligned}$$

where the minus signs in some w^{β} terms occur due to translation in space, and $w^{\ell}(\boldsymbol{n})$ is the filterbank coefficients of the signal in Equation 9.5. The globally bandlimitedness of difference images, as argued in the previous section, allows us to conclude that $w_{\boldsymbol{i}}^{\alpha}(\boldsymbol{n}) \approx 0$ and $w_{\boldsymbol{i}}^{\beta}(\boldsymbol{n}) \approx 0, \forall i_0 > \hat{I}$ or $i_1 > \hat{I}$ for some $\hat{I}$. The above simplifies to a form that reveals the

energy compaction structure within CFA image:

$$w_{\boldsymbol{i}}^{y}(\boldsymbol{n}) \approx \begin{cases} w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) + \left(w^{\alpha}_{(I-i_0,i_1)}(\boldsymbol{n}) - w^{\beta}_{(I-i_0,i_1)}(\boldsymbol{n})\right)/4 & \text{if } I-i_0 < \hat{I},\, i_1 < \hat{I} \\ w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) + \left(w^{\alpha}_{(i_0,I-i_1)}(\boldsymbol{n}) - w^{\beta}_{(i_0,I-i_1)}(\boldsymbol{n})\right)/4 & \text{if } i_0 < \hat{I},\, I-i_1 < \hat{I} \\ w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) + \left(w^{\alpha}_{(I-i_0,I-i_1)}(\boldsymbol{n}) + w^{\beta}_{(I-i_0,I-i_1)}(\boldsymbol{n})\right)/4 & \text{if } I-i_0 < \hat{I},\, I-i_1 < \hat{I} \\ w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) & \text{otherwise} \end{cases} \tag{9.9}$$

Recall Equation 9.2 and that the filterbank transforms with appropriate choices of filters constitute a unitary transform. Thus, $w_{\boldsymbol{i}}^{z}(\boldsymbol{n}) = w_{\boldsymbol{i}}^{y}(\boldsymbol{n}) + w_{\boldsymbol{i}}^{\varepsilon}(\boldsymbol{n})$, providing

$$w_{\boldsymbol{i}}^{z}(\boldsymbol{n}) \approx \begin{cases} w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) + \left(w^{\alpha}_{(I-i_0,i_1)}(\boldsymbol{n}) - w^{\beta}_{(I-i_0,i_1)}(\boldsymbol{n})\right)/4 + w_{\boldsymbol{i}}^{\varepsilon}(\boldsymbol{n}) & \text{if } I-i_0 < \hat{I},\, i_1 < \hat{I} \\ w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) + \left(w^{\alpha}_{(i_0,I-i_1)}(\boldsymbol{n}) - w^{\beta}_{(i_0,I-i_1)}(\boldsymbol{n})\right)/4 + w_{\boldsymbol{i}}^{\varepsilon}(\boldsymbol{n}) & \text{if } i_0 < \hat{I},\, I-i_1 < \hat{I} \\ w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) + \left(w^{\alpha}_{(I-i_0,I-i_1)}(\boldsymbol{n}) + w^{\beta}_{(I-i_0,I-i_1)}(\boldsymbol{n})\right)/4 + w_{\boldsymbol{i}}^{\varepsilon}(\boldsymbol{n}) & \text{if } I-i_0 < \hat{I},\, I-i_1 < \hat{I} \\ w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) + w_{\boldsymbol{i}}^{\varepsilon}(\boldsymbol{n}) & \text{otherwise,} \end{cases} \tag{9.10}$$

where $w_{\boldsymbol{i}}^{\varepsilon} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{\varepsilon}^2)$ is a filterbank transform of $\varepsilon(\boldsymbol{n})$. In other words, the filterbank transformation of noisy sensor data w^z is the baseband luminance coefficient w^{ℓ} distorted by the noise w^{ε} and aliasing due to reversed-order filterbank coefficients w^{α} and w^{β}, where w^{ℓ}, w^{α}, and w^{β} are (conditionally) normal. A unified strategy to demosaicking and denoising, therefore, is to design an estimator that estimates w^{ℓ}, w^{α}, and w^{β} from the mixture of w^{ℓ}, w^{α}, w^{β}. and w^{ε}. We will see how this can be accomplished in Section 9.7.

Lastly, we remind the readers that Equation 9.10 can be generalized to any filterbanks that satisfy Equation 9.7 using time-reversed filter coefficients for h_0 and h_1. However, Haar wavelets are used exclusively in this chapter to simplify the notation.

9.5 Constrained Filtering

In this section, we motivate an approach to joint demosaicking and denoising using well-understood DSP machineries [40]. Recall Equations 9.3 and 9.4. We are interested in estimating $\boldsymbol{x}(\boldsymbol{n})$ given $z(\cdot)$. It is worth noting that even if $z(\boldsymbol{n})$ for some $\boldsymbol{n}$ corresponds to an observation of a red pixel $x_1(\boldsymbol{n})$, for example, $z(\boldsymbol{n})$ does not suffice as an estimate of $x_1(\boldsymbol{n})$ (unlike the pure demosaicking problems) because it is contaminated by noise.

We begin by highlighting monochromatic image denoising methods that operate by taking a linear combination of neighboring pixels. These methods include bilateral filters [28], principal components [31], and total least squares based methods [32], where the linear weights adapt to the local image features. Transform-based shrinkage and threshold methods can also be re-interpreted as spatially-varying linear estimators, because there exists a linear combination of neighboring pixels that is equivalent to shrinkage of transform coefficients. In the Bayesian estimation framework, the linearity of estimation is (conditionally) true for (a mixture of) normally distributed transform coefficients. In any case, the

locally adaptive linear estimator, x^{est}, takes the general form:

$$x^{\text{est}}(\boldsymbol{n}) = \sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} g(\boldsymbol{n},\boldsymbol{m})z(\boldsymbol{n}-\boldsymbol{m}),$$

where $z(\boldsymbol{n})$ is the noisy version of $x(\boldsymbol{n})$, $g(\boldsymbol{n},\boldsymbol{m})$ is the spatially-adaptive linear weights, and the summation is over $\boldsymbol{\eta}(\boldsymbol{n})$, a local neighborhood of pixels centered around $\boldsymbol{n}$. Typically, we choose $g(\boldsymbol{n},\boldsymbol{m})$ such that it solves the least-squares minimization problem (though not necessarily [32]),

$$\min_{g} E\left\|x(\boldsymbol{n}) - x^{\text{est}}(\boldsymbol{n})\right\|^2. \tag{9.11}$$

In this section, we will show how the estimator in the above form can be modified such that the linear weights can be used to simultaneously interpolate and denoise CFA data [40]. Let $\boldsymbol{x}^{\text{est}}(\boldsymbol{n})$ be an estimate of ideal color image $\boldsymbol{x}(\boldsymbol{n})$ by taking a linear combination of noisy sensor data $z(\boldsymbol{n})$. That is,

$$\boldsymbol{x}^{\text{est}}(\boldsymbol{n}) = \sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} \boldsymbol{g}(\boldsymbol{n},\boldsymbol{m})z(\boldsymbol{n}-\boldsymbol{m}), \tag{9.12}$$

where $\boldsymbol{g}(\boldsymbol{n},\boldsymbol{m}) \in \mathbb{R}^3$ is a spatially-adaptive linear weight.

Let $g_k(\boldsymbol{n},\boldsymbol{m})$ correspond to the linear weight for estimating x_k. In the following discussion, we focus on the estimation of $x_2(\boldsymbol{n})$ via the design of $g_2(\boldsymbol{n},\cdot)$ because Equation 9.3 already assumes $x_2(\boldsymbol{n})$ as its baseband. The results achieved here are generalized to the estimation of $x_1(\boldsymbol{n})$ and $x_3(\boldsymbol{n})$ at the end of this section. Substituting Equation 9.6 into Equation 9.12,

$$\begin{aligned} x_2^{\text{est}}(\boldsymbol{n}) &= \sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} g_2(\boldsymbol{n},\boldsymbol{m})z(\boldsymbol{n}-\boldsymbol{m}) \\ &= \sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} \Big(g_2(\boldsymbol{n},\boldsymbol{m})\big(x_2(\boldsymbol{n}-\boldsymbol{m}) + \varepsilon(\boldsymbol{n}-\boldsymbol{m})\big) \\ &\quad + g_2(\boldsymbol{n},\boldsymbol{m})\big(c_1(\boldsymbol{n}-\boldsymbol{m})\alpha(\boldsymbol{n}-\boldsymbol{m}) + c_3(\boldsymbol{n}-\boldsymbol{m})\beta(\boldsymbol{n}-\boldsymbol{m})\big)\Big). \end{aligned} \tag{9.13}$$

The first term, $\sum_{\boldsymbol{m}} g_2(\boldsymbol{n},\boldsymbol{m})[x_2(\boldsymbol{n}-\boldsymbol{m}) + \varepsilon(\boldsymbol{n}-\boldsymbol{m})]$ represents an ordinary monochromatic image denoising. That is, $g_2(\cdot,\cdot)$ operates on the noisy version of $x_2(\cdot)$. The extra term involving $\alpha(\cdot)$ and $\beta(\cdot)$ also motivates the need for further restricting $g_2(\cdot,\cdot)$ such that the latter term is attenuated.

To accomplish this task, recall that $c_1(\boldsymbol{n})\alpha(\boldsymbol{n})$ and $c_3(\boldsymbol{n})\beta(\boldsymbol{n})$ occupy frequency regions around $\boldsymbol{\omega} = \{(0,0),(0,\pi),(\pi,0),(\pi,\pi)\}$. Let us consider a class of linear filters with stopbands near $\{(0,0),(0,\pi),(\pi,0),(\pi,\pi)\}$ (i.e., band-pass). In particular, if the filter coefficients corresponding to red and blue samples in CFA sum to zero, respectively, then $\forall \boldsymbol{n}$,

$$\begin{aligned} \sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} g_2(\boldsymbol{n},\boldsymbol{m})c_1(\boldsymbol{n}-\boldsymbol{m}) &= 0 \\ \sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} g_2(\boldsymbol{n},\boldsymbol{m})c_3(\boldsymbol{n}-\boldsymbol{m}) &= 0, \end{aligned} \tag{9.14}$$

and with a finite spatial support on $g_2(\cdot,\boldsymbol{m})$, we can safely assume that the frequency components in the vicinity of $\{(0,0),(0,\pi),(\pi,0),(\pi,\pi)\}$ are attenuated as well (because $\hat{g}_2$ is a linear combination of cosines in the Fourier domain).

If the restriction in Equation 9.14 holds true, then the estimator in Equation 9.13 reduces to a monochromatic image denoising problem — that is, $x_2^{\text{est}} \approx \sum_{\boldsymbol{m}} g_2(\boldsymbol{n},\boldsymbol{m})[x_2(\boldsymbol{n}-\boldsymbol{m}) + \varepsilon(\boldsymbol{n}-\boldsymbol{m})]$. Therefore, the underlying strategy for deriving a joint demosaicking and denoising operator is to solve a constrained linear estimation problem. In other words, instead of Equation 9.11, solve

$$J = \min\ E\left\| x_2(\boldsymbol{n}) - \sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} g_2(\boldsymbol{n},\boldsymbol{m})[x_2(\boldsymbol{n}-\boldsymbol{m}) + \varepsilon(\boldsymbol{n}-\boldsymbol{m})]\right\|^2. \tag{9.15}$$
$$\text{subject to} \sum_{\boldsymbol{m}} g_2(\boldsymbol{n},\boldsymbol{m})c_1(\boldsymbol{n}-\boldsymbol{m}) = \sum_{\boldsymbol{m}} g_2(\boldsymbol{n},\boldsymbol{m})c_3(\boldsymbol{n}-\boldsymbol{m}) = 0$$

Conveniently, this optimization problem allows us to *pretend* as though we are designing a monochromatic image denoising method. However, the constraints on the filter coefficients ensure that J remains a good approximation to the residual of the actual problem, $\|x_2(\boldsymbol{n}) - x_2^{\text{est}}(\boldsymbol{n})\|^2$. Note that Equation 9.14 does not imply $\sum_{\boldsymbol{m}} g_2(\boldsymbol{n},\boldsymbol{m})c_2(\boldsymbol{n}-\boldsymbol{m}) = 0$. Instead, the contributions from x_1 and x_3 to the estimation of x_2 are limited to the frequency components in the band-pass region, whereas the contributions from x_2 are unrestricted.

In many cases, the existing image denoising techniques naturally extend to simultaneously solving the demosaicking and denoising problems. Let $\vec{x}$ (and similarly $\vec{\varepsilon},\vec{z},\vec{g}$) be a re-arrangement of $\{x(\boldsymbol{n}-\boldsymbol{m})|\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})\}$ into a vector form. Then least-squares solution to Equation 9.11 often involves an inner product of the form $x^{\text{est}}(\boldsymbol{n}) = \vec{g}_{\text{LS}}^T(\vec{x}+\vec{\varepsilon})$, where

$$\vec{g}_{\text{LS}} = E\left[(\vec{x}+\vec{\varepsilon})(\vec{x}+\vec{\varepsilon})^T\right]^{-1} E\left[(\vec{x}+\vec{\varepsilon})^T x(\boldsymbol{n})\right]. \tag{9.16}$$

The inner product occurs often in Bayesian estimators, when the prior on the data are (conditionally) normally distributed (e.g., Laplace, Student's t, Gaussian mixture). If this prior on x is defined in the linear transform domain (such as on the wavelet coefficients), then the equivalent second-order statistics in the pixel domain are simply a linear transformation of the statistics in the transform domain.

Let $M = |\boldsymbol{\eta}(\boldsymbol{n})|$ be the size of the neighborhood, $\boldsymbol{\eta}(\boldsymbol{n})$. The band-pass constraint in Equation 9.14 may be imposed by asserting that $\vec{g}\in\mathbb{R}^M$ lives in a lower-dimensional subspace, $\text{span}\{\vec{v}\in\mathbb{R}^M|\vec{v}^T\vec{c}_1 = \vec{v}^T\vec{c}_3 = 0\}$, or $\vec{g} = \boldsymbol{G}\boldsymbol{s}$, where $\boldsymbol{G}\in\mathbb{R}^{M\times M-2}$ is an orthogonal matrix whose column vectors span this subspace. Then the constrained LS problem in Equation 9.15 can be rewritten as

$$J = \min_{\vec{g}=\boldsymbol{G}\boldsymbol{s}} E\left\|\vec{x}_2 - \vec{g}^T[\vec{x}_2+\vec{\varepsilon}]\right\|^2 = \min_{\boldsymbol{s}} E\left\|\vec{x}_2 - \boldsymbol{s}^T\boldsymbol{G}^T[\vec{x}_2+\vec{\varepsilon}]\right\|^2. \tag{9.17}$$

It is easy to verify that the solution to the above has the form $x_2^{\text{est}}(\boldsymbol{n}) = \vec{g}_{\text{CLS}}^T\vec{z}$, where

$$\vec{g}_{\text{CLS}} = \boldsymbol{G}\left[\boldsymbol{G}^T E\left[(\vec{x}_2+\vec{\varepsilon})(\vec{x}_2+\vec{\varepsilon})^T\right]\boldsymbol{G}\right]^{-1} E\left[\boldsymbol{G}(\vec{x}_2+\vec{\varepsilon})^T x_2(\boldsymbol{n})\right]. \tag{9.18}$$

Note that Equations 9.17 and 9.18 are minor alterations to Equations 9.11 and 9.16 using the same second-order statistics, respectively, and thus it is a straightforward exercise to

leverage existing monochromatic image denoising methods to a joint demosaicking and denoising scheme.

In order to design spatially adaptive filters similar to Equation 9.18 for estimating x_1 and x_3, we see that Equation 9.3 can be written alternatively as

$$\begin{aligned} y(\boldsymbol{n}) &= c_2(\boldsymbol{n})[x_2(\boldsymbol{n}) - x_1(\boldsymbol{n})] + c_3(\boldsymbol{n})[x_3(\boldsymbol{n}) - x_1(\boldsymbol{n})] + x_1(\boldsymbol{n}) \\ &= c_1(\boldsymbol{n})[x_1(\boldsymbol{n}) - x_3(\boldsymbol{n})] + c_2(\boldsymbol{n})[x_2(\boldsymbol{n}) - x_3(\boldsymbol{n})] + x_3(\boldsymbol{n}). \end{aligned}$$

It follows that the appropriate constraints on filter coefficients $g_1(\cdot,\cdot)$ and $g_3(\cdot,\cdot)$ are

$$\sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} g_1(\boldsymbol{n},\boldsymbol{m})c_2(\boldsymbol{n}-\boldsymbol{m}) = 0, \qquad \sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} g_1(\boldsymbol{n},\boldsymbol{m})c_3(\boldsymbol{n}-\boldsymbol{m}) = 0$$

$$\sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} g_3(\boldsymbol{n},\boldsymbol{m})c_1(\boldsymbol{n}-\boldsymbol{m}) = 0, \qquad \sum_{\boldsymbol{m}\in\boldsymbol{\eta}(\boldsymbol{n})} g_3(\boldsymbol{n},\boldsymbol{m})c_2(\boldsymbol{n}-\boldsymbol{m}) = 0.$$

9.6 Missing Data

The statistical modelling of image signals in a linear transform domain is primarily motivated by the correlation structures that exist within the transform coefficients of image signals. These models, which require a complete observation of image data, are not easily generalizable to the digital camera context, as the observation of color image data is incomplete at the sensor interface. That is, processing with missing or incomplete pixels is difficult because a linear transformation takes a linear combination of the pixel values, and thus *all* of the noisy transform coefficients are unobserved. Yet, it is still convenient or desirable to apply the sophisticated statistical modelling techniques even when none of the transform coefficients are observable.

This section explicitly addresses the issue of combining the treatment of missing data and the wavelet-based modeling [49]. Bayesian hierarchical modelling is used to capture the second-order statistics in the transform domain. We assume a general model form and couple the EM algorithm framework with the Bayesian models to estimate the hyper- and nuisance parameters via the marginal likelihood; that is, we adopt the empirical partial Bayes approach. Within this framework, problems with missing pixels or pixel components, and hence unobservable wavelet coefficients, are handled simultaneously with image denoising.

In order to extend the complete image modelling strategy to incomplete data, let $\boldsymbol{w}_i^{\boldsymbol{x}}(\boldsymbol{n}) = [w_i^{x_1}(\boldsymbol{n}), w_i^{x_2}(\boldsymbol{n}), w_i^{x_3}(\boldsymbol{n})]^T$ correspond to wavelet coefficients corresponding to x_1, x_2, x_3 at i-th level, and assume that

$$\boldsymbol{w}_i^{\boldsymbol{x}}(\boldsymbol{n}) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \boldsymbol{\Sigma}_i).$$

Then the distribution of the neighboring pixel values is also jointly normal, as linear transformation of a multivariate normal vector is also normal. Because the wavelet transform is unitary,

$$\boldsymbol{w}^{\boldsymbol{x}+\boldsymbol{\epsilon}}(\boldsymbol{n}) | \boldsymbol{w}^{\boldsymbol{x}}(\boldsymbol{n}) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\boldsymbol{w}^{\boldsymbol{x}}(\boldsymbol{n}), \sigma_{\boldsymbol{\epsilon}}^2 \boldsymbol{I}).$$

To summarize, $\boldsymbol{\theta} = \{\boldsymbol{\Sigma}_i, \sigma_\epsilon^2\}$ are the hyper- and nuisance parameters, respectively.

If the $\boldsymbol{\theta}$ is known, the regression of the missing pixels on the known clean pixel-component measurements $y(\boldsymbol{n})$, $E[\boldsymbol{x}(\boldsymbol{n})|y(\boldsymbol{n}),\boldsymbol{\theta}]$, serves as a demosaicking method based on the LS estimator and has a straightforward implementation. Conditioned on the incomplete and noisy measurement of pixel-components $z(\boldsymbol{n})$, $E[\boldsymbol{x}(\boldsymbol{n})|z(\boldsymbol{n}),\boldsymbol{\theta}]$, is an interpolated and denoised image signal, where

$$\boldsymbol{x}^{\text{est}}(\boldsymbol{n}) = E[\boldsymbol{x}(\boldsymbol{n})|z(\boldsymbol{n}),\boldsymbol{\theta}] = E\left[E[\boldsymbol{x}(\boldsymbol{n})|y(\boldsymbol{n}),z(\boldsymbol{n}),\boldsymbol{\theta}]\Big|z(\boldsymbol{n}),\boldsymbol{\theta}\right].$$

The nested expectation operator has an intuitive interpretation: the inner expectation, $E[\boldsymbol{x}(\boldsymbol{n})|y(\boldsymbol{n}),z(\boldsymbol{n}),\boldsymbol{\theta}] = E[\boldsymbol{x}(\boldsymbol{n})|y(\boldsymbol{n}),\boldsymbol{\theta}]$, is an interpolator, and the outer expectation is the denoiser. Conversely, the same formula can equivalently be written as:

$$\boldsymbol{x}^{\text{est}}(\boldsymbol{n}) = E[\boldsymbol{x}(\boldsymbol{n})|z(\boldsymbol{n}),\boldsymbol{\theta}] = E\left[E[\boldsymbol{x}(\boldsymbol{n})|(\boldsymbol{x}+\boldsymbol{\epsilon})(\boldsymbol{n}),z(\boldsymbol{n}),\boldsymbol{\theta}]\Big|z(\boldsymbol{n}),\boldsymbol{\theta}\right],$$

where the inner expectation operator, $E[\boldsymbol{x}(\boldsymbol{n})|(\boldsymbol{x}+\boldsymbol{\epsilon})(\boldsymbol{n}),z(\boldsymbol{n}),\boldsymbol{\theta}] = E[\boldsymbol{x}(\boldsymbol{n})|(\boldsymbol{x}+\boldsymbol{\epsilon})(\boldsymbol{n}),\boldsymbol{\theta}]$ is a denoiser, and the outer expectation is the interpolator. Conditioned on $\boldsymbol{\theta}$, therefore, a design of simultaneous demosaicking and denoising method is straightforward.

The posterior mean estimate, $\boldsymbol{x}^{\text{est}}$, is sensitive to the choice of parameters $\boldsymbol{\theta}$; and given only a subset of the noisy pixel components $z(\boldsymbol{n})$, we are left with estimating $\boldsymbol{\theta}$ from the data when the wavelet coefficients are not observable. In particular, we solve for the $\boldsymbol{\theta}$ that maximizes the marginal log-likelihood $\log p(z|\boldsymbol{\theta})$, and estimate $\boldsymbol{x}$ as its posterior mean conditioned on $\hat{\boldsymbol{\theta}}$ (where $\hat{\boldsymbol{\theta}}$ is obtained from the maximal likelihood estimate above). The direct maximization of $\log p(z|\boldsymbol{\theta})$ is very difficult because of the missing pixel values. The EM algorithm circumvents this problem by iteratively maximizing the much easier *augmented-data* log-likelihood, $\log p(\boldsymbol{x},\boldsymbol{\epsilon}|\boldsymbol{\theta})$, where $\{\boldsymbol{x},\boldsymbol{\epsilon}\}$ are the augmented data.

Given the $[t]$-th iterate hyper- and nuisance parameter estimate, $\boldsymbol{\theta}^{[t]} = \{\boldsymbol{\Sigma}_i^{[t]}, \sigma_\epsilon^{2[t]}\}$, the $[t+1]$-st iteration of the EM algorithm first calls for

$$Q\left(\boldsymbol{\theta};\boldsymbol{\theta}^{[t]}\right) = E\left[\log p(\boldsymbol{x},\boldsymbol{\epsilon}|\boldsymbol{\theta})\,\Big|z,\boldsymbol{\theta}^{[t]}\right].$$

A celebrated result of EM algorithm [50] states that

$$\log p(z|\boldsymbol{\theta}) - \log p(z|\boldsymbol{\theta}^{[t]}) \geq Q\left(\boldsymbol{\theta};\boldsymbol{\theta}^{[t]}\right) - Q\left(\boldsymbol{\theta}^{[t]};\boldsymbol{\theta}^{[t]}\right),$$

where $\log p(z|\boldsymbol{\theta})$ is the log-likelihood of $\boldsymbol{\theta}$ based on the actual observed data, $z(\boldsymbol{n})$. Thus, the choice of $\boldsymbol{\theta}$ that maximizes $Q(\boldsymbol{\theta};\boldsymbol{\theta}^{[t]})$, that is, the next iterate $\boldsymbol{\theta}^{[t+1]}$, increases our objective function:

$$\log p\left(z\Big|\boldsymbol{\theta}^{[t+1]}\right) \geq \log p\left(z\Big|\boldsymbol{\theta}^{[t]}\right).$$

Consequently, maximizing $Q(\boldsymbol{\theta};\boldsymbol{\theta}^{[t]})$ is the same as maximizing $\log p(z|\boldsymbol{\theta})$, but with augmented-data sufficient statistics. Given $[t]$-th iterate hyper- and nuisance parameters $\boldsymbol{\theta}^{[t]}$, the explicit formula for $Q(\boldsymbol{\theta};\boldsymbol{\theta}^{[t]})$ is in the closed form:

$$\begin{aligned} Q\left(\boldsymbol{\theta};\boldsymbol{\theta}^{[t]}\right) &= E\left[\log p(\boldsymbol{x},\boldsymbol{\epsilon}|\boldsymbol{\theta})\,\Big|z,\boldsymbol{\theta}^{[t]}\right] \\ &= \sum_{i,\boldsymbol{n}} E\left[\log p(\boldsymbol{w}_i^x(\boldsymbol{n})|\boldsymbol{\Sigma}_i) + \log p(\boldsymbol{w}_i^\epsilon(\boldsymbol{n})|\sigma_\epsilon^2)\,\Big|z,\boldsymbol{\theta}^{[t]}\right]. \end{aligned} \tag{9.19}$$

It is then easy to verify that the maximizer of $Q(\boldsymbol{\theta};\boldsymbol{\theta}^{[t]})$ is the weighted least squares estimate [50]:

$$\begin{aligned}\boldsymbol{\Sigma}_{\boldsymbol{i}}^{[t+1]} &= \frac{1}{N_{\boldsymbol{i}}}\sum_{\boldsymbol{n}} E\left[\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{x}}(\boldsymbol{n})\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{x}T}(\boldsymbol{n})\,\middle|\,z,\boldsymbol{\theta}^{[t]}\right] \\ \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}^{[t+1]} &= \frac{1}{3\sum_{\boldsymbol{i}} N_{\boldsymbol{i}}}\sum_{\boldsymbol{i},\boldsymbol{n}} E\left[\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{\epsilon}T}(\boldsymbol{n})\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{\epsilon}}(\boldsymbol{n})\,\middle|\,z,\boldsymbol{\theta}^{[t]}\right],\end{aligned} \tag{9.20}$$

where $N_{\boldsymbol{i}}$ is the number of wavelets samples in the $\boldsymbol{i}$-th subband. In each iteration, the computation of the sufficient statistics in Equation 9.19 is often called expectation- or E-step, whereas the process of carrying out Equation 9.20 to find $\boldsymbol{\theta}^{[t+1]}$ is referred to as maximization- or M-step. Carrying out the math to find $E[\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{x}}(\boldsymbol{n})\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{x}T}(\boldsymbol{n})|z,\boldsymbol{\theta}^{[t]}]$ and $E[\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{\epsilon}T}(\boldsymbol{n})\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{\epsilon}}(\boldsymbol{n})|z,\boldsymbol{\theta}^{[t]}]$ in E-step is rather cumbersome, and the derivation is omitted in this chapter. Interested readers are encouraged to refer to References [49] and [50] for more details.

As was the case in the previous section, it is worth noting that wavelet coefficients are often modelled with heavy-tailed distributions (e.g., Laplace, Student's t, Gaussian mixture). Distributions belonging to an exponential family can be rewritten as a scalar mixture of Gaussian random variables, and thus are conditionally Gaussian. The EM algorithm developed above is therefore generalized to a heavy-tailed distribution via the integration over the mixture variable in the posterior sense.

9.7 Filterbank Coefficient Estimation

Computational efficiency and elegance of shrinkage or thresholding estimators and theoretical properties amenable to spatial inhomogeneities have contributed to the immense popularity of wavelet-based methods for image denoising. However, typical denoising techniques assume complete grayscale or color image observation, and hence must be applied *after* demosaicking. In the previous section we showed that it is possible to model noisy color images in the wavelet domain directly by taking advantage of the statistical framework of missing data. However, the computational burden of doing so is severe, and the energy compaction arguments put forth in Section 9.4 suggest an alternative approach by choosing to work with wavelet coefficients of the noisy subsampled data directly. In this section, we propose necessary changes to a *complete image* wavelet coefficient model such that it is amenable to the direct manipulation of $w_{\boldsymbol{i}}^{y}(\boldsymbol{n})$, [17].

Given that the difference images are sufficiently low-pass, simplification in Equation 9.9 reveals that there is a surprising degree of similarity between $w_{\boldsymbol{i}}^{y}(\boldsymbol{n})$ and $w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n})$. Specifically, $w_{\boldsymbol{i}}^{y}(\boldsymbol{n}) \approx w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n})$ for the majority of subbands — the exceptions are the subbands that are normally considered high-frequency, which now contain a strong mixture of the low-frequency (or scaling) coefficients from the difference images, α and β. Operating under the premises that the filterbank transform decomposes image signals such that subbands are approximately uncorrelated from each other, the posterior mean estimate of $w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n})$ takes the

form

$$\{w_{\boldsymbol{i}}^{\ell}\}^{\text{est}}(\boldsymbol{n}) = E\left[w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n})\middle|w_{\boldsymbol{i}}^{z}\right] \approx E\left[w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n})\middle|w_{\boldsymbol{i}}^{\ell+\varepsilon}\right]$$

for all subbands that meet the $w_{\boldsymbol{i}}^{y}(\boldsymbol{n}) \approx w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n})$ approximation. Since the wavelet shrinkage function $f:\mathbb{R}\to\mathbb{R}$, $f(w_{\boldsymbol{i}}^{\ell+\varepsilon}) = E(w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n})|w_{\boldsymbol{i}}^{\ell+\varepsilon})$ is a well studied problem in the literature, we can leverage existing image denoising methods to the CFA image context. In a simple special case where $w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}) \sim \mathcal{N}(0,\sigma_{\ell,\boldsymbol{i}}^2)$, the L^2 estimator is

$$f(w_{\boldsymbol{i}}^{z}) = \frac{\sigma_{\ell,\boldsymbol{i}}^2}{\sigma_{\ell,\boldsymbol{i}}^2+\sigma_{\varepsilon}^2} w_{\boldsymbol{i}}^{z}(\boldsymbol{n}).$$

However, in the subbands that contain a mixture of $w_{\boldsymbol{i}}^{\ell}$, $w_{\boldsymbol{i}}^{\alpha}$, $w_{\boldsymbol{i}}^{\beta}$, and $w_{\boldsymbol{i}}^{\varepsilon}$, we must proceed with caution. Let $w_{\boldsymbol{i}}^{\alpha}(\boldsymbol{n}) \sim \mathcal{N}(0,\sigma_{\alpha,\boldsymbol{i}}^2)$, $w_{\boldsymbol{i}}^{\beta}(\boldsymbol{n}) \sim \mathcal{N}(0,\sigma_{\beta,\boldsymbol{i}}^2)$. Consider the case such that $i_0 > I-\hat{I}$ and $i_1 < \hat{I}$, and define $\boldsymbol{j} = (i_0, I-i_1), \boldsymbol{k} = (I-i_0, I-i_1)$. Then $w_{\boldsymbol{i}}^{z}(\boldsymbol{n}), w_{\boldsymbol{j}}^{z}(\boldsymbol{n}), w_{\boldsymbol{k}}^{z}(\boldsymbol{n})$ are highly correlated due to their common components in their mixture, $w_{\boldsymbol{i}'}^{\alpha}$ and $w_{\boldsymbol{i}'}^{\beta}$, where $\boldsymbol{i}' = (I-i_0, i_1)$. Thus the L^2 estimates for $w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n}), w_{\boldsymbol{j}}^{\ell}(\boldsymbol{n}), w_{\boldsymbol{k}}^{\ell}(\boldsymbol{n})$ are

$$\begin{bmatrix}\{w_{\boldsymbol{i}}^{\ell}\}^{\text{est}}(\boldsymbol{n})\\ \{w_{\boldsymbol{j}}^{\ell}\}^{\text{est}}(\boldsymbol{n})\\ \{w_{\boldsymbol{k}}^{\ell}\}^{\text{est}}(\boldsymbol{n})\end{bmatrix} = E\left(\begin{bmatrix}w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{\ell}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{\ell}\end{bmatrix}\middle|\begin{bmatrix}w_{\boldsymbol{i}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{z}\end{bmatrix}\right)$$

$$= E\left(\begin{bmatrix}w_{\boldsymbol{i}}^{\ell}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{\ell}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{\ell}\end{bmatrix}\begin{bmatrix}w_{\boldsymbol{i}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{z}\end{bmatrix}^{T}\right) E\left(\begin{bmatrix}w_{\boldsymbol{i}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{z}\end{bmatrix}\begin{bmatrix}w_{\boldsymbol{i}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{z}\end{bmatrix}^{T}\right)^{-1}\begin{bmatrix}w_{\boldsymbol{i}}^{z}\\ w_{\boldsymbol{j}}^{z}\\ w_{\boldsymbol{k}}^{z}\end{bmatrix}$$

$$= \begin{bmatrix}\sigma_{\ell,\boldsymbol{i}}^2 & & \\ & \sigma_{\ell,\boldsymbol{j}}^2 & \\ & & \sigma_{\ell,\boldsymbol{k}}^2\end{bmatrix}\begin{bmatrix}\sigma_{\ell,\boldsymbol{i}}^2+\sigma_{\varepsilon}^2+\frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2-\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16}\\ \frac{\sigma_{\alpha,\boldsymbol{i}'}^2-\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \sigma_{\ell,\boldsymbol{j}}^2+\sigma_{\varepsilon}^2+\frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2-\sigma_{\beta,\boldsymbol{i}'}^2}{16}\\ \frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2-\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \sigma_{\ell,\boldsymbol{k}}^2+\sigma_{\varepsilon}^2+\frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16}\end{bmatrix}^{-1}\begin{bmatrix}w_{\boldsymbol{i}}^{z}\\ w_{\boldsymbol{j}}^{z}\\ w_{\boldsymbol{k}}^{z}\end{bmatrix}.$$

Similarly,

$$\begin{bmatrix}\{w_{\boldsymbol{i}'}^{\alpha}\}^{\text{est}}(\boldsymbol{n})\\ \{w_{\boldsymbol{i}'}^{\beta}\}^{\text{est}}(\boldsymbol{n})\end{bmatrix} = E\left(\begin{bmatrix}w_{\boldsymbol{i}'}^{\alpha}(\boldsymbol{n})\\ w_{\boldsymbol{i}'}^{\beta}(\boldsymbol{n})\end{bmatrix}\middle|\begin{bmatrix}w_{\boldsymbol{i}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{z}\end{bmatrix}\right)$$

$$= E\left(\begin{bmatrix}w_{\boldsymbol{i}'}^{\alpha}(\boldsymbol{n})\\ w_{\boldsymbol{i}'}^{\beta}(\boldsymbol{n})\end{bmatrix}\begin{bmatrix}w_{\boldsymbol{i}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{z}\end{bmatrix}^{T}\right) E\left(\begin{bmatrix}w_{\boldsymbol{i}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{z}\end{bmatrix}\begin{bmatrix}w_{\boldsymbol{i}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{j}}^{z}(\boldsymbol{n})\\ w_{\boldsymbol{k}(\boldsymbol{n})}^{z}\end{bmatrix}^{T}\right)^{-1}\begin{bmatrix}w_{\boldsymbol{i}}^{z}\\ w_{\boldsymbol{j}}^{z}\\ w_{\boldsymbol{k}}^{z}\end{bmatrix}$$

$$= \begin{bmatrix}\frac{\sigma_{\alpha,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2}{16}\\ -\frac{\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\beta,\boldsymbol{i}'}^2}{16} & -\frac{\sigma_{\beta,\boldsymbol{i}'}^2}{16}\end{bmatrix}\begin{bmatrix}\sigma_{\ell,\boldsymbol{i}}^2+\sigma_{\varepsilon}^2+\frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2-\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16}\\ \frac{\sigma_{\alpha,\boldsymbol{i}'}^2-\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \sigma_{\ell,\boldsymbol{j}}^2+\sigma_{\varepsilon}^2+\frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2-\sigma_{\beta,\boldsymbol{i}'}^2}{16}\\ \frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \frac{\sigma_{\alpha,\boldsymbol{i}'}^2-\sigma_{\beta,\boldsymbol{i}'}^2}{16} & \sigma_{\ell,\boldsymbol{k}}^2+\sigma_{\varepsilon}^2+\frac{\sigma_{\alpha,\boldsymbol{i}'}^2+\sigma_{\beta,\boldsymbol{i}'}^2}{16}\end{bmatrix}^{-1}\begin{bmatrix}w_{\boldsymbol{i}}^{z}\\ w_{\boldsymbol{j}}^{z}\\ w_{\boldsymbol{k}}^{z}\end{bmatrix}.$$

Once $\{w_{\boldsymbol{i}}^{\ell}\}^{\text{est}}, \{w_{\boldsymbol{i}}^{\alpha}\}^{\text{est}}, \{w_{\boldsymbol{i}}^{\beta}\}^{\text{est}}$ are computed $\forall \boldsymbol{i}, \boldsymbol{n}$ as above, then $\boldsymbol{x}^{\text{est}}(\boldsymbol{n})$ is calculated by taking the inverse filterbank transform of $\{w_{\boldsymbol{i}}^{\ell}\}^{\text{est}}, \{w_{\boldsymbol{i}}^{\alpha}\}^{\text{est}}, \{w_{\boldsymbol{i}}^{\beta}\}^{\text{est}}$ to find the estimates of $\ell(\boldsymbol{n}), \alpha(\boldsymbol{n}), \beta(\boldsymbol{n})$, which in turn is used to solve $\boldsymbol{x}^{\text{est}}$.

Practically, it should be noted that the actual implementation of this method should include cycle-spinning, a standard technique in filterbank and wavelet literature whereby a linear space-variant system can be transformed into linear space-invariant system via averaging over all possible spacial shifts. As with the previous sections, we note that the estimator naturally extends to multivariate normal or heavy-tailed distributions.

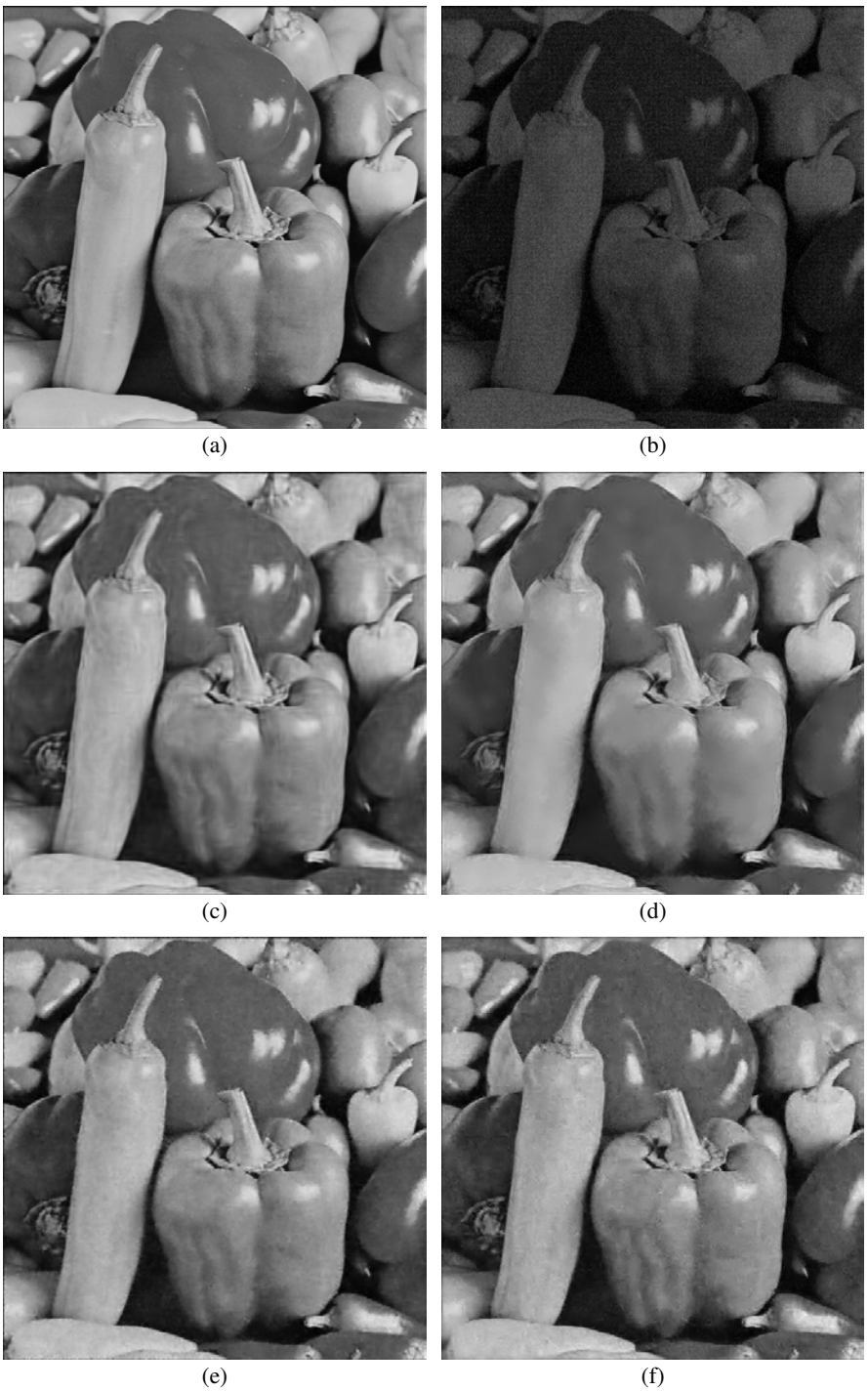
9.8 Conclusion

Given the inadequacies and model inconsistencies of treating the image denoising and demosaicking problem independently, we focused on the analysis and the techniques for processing (see Figure 9.8 and Figure 9.9) subsampled data. In particular, the Fourier and filterbank (wavelet-packet) analyses reveal a systematic aliasing structure in CFA images, where the observed data consists of a mixture of baseband luminance signal, spectrally-shifted difference images, and noise. The same analysis motivates a unified strategy to address demosaicking and denoising estimation problems by interpreting the sensor data as luminance image distorted by noise with some degrees of structure.

Conditioned on the *complete observation* image model of the digital camera designer's choosing, we proposed three design regimes for estimating the complete noise-free image signal of interest given a set of incomplete observations of pixel components that are corrupted by noise. First, well-understood DSP machineries were employed to design a spatially-adaptive linear filter whose stop-band contains the spectral copies of the difference images, and the pass-band suppresses noise. Second, coupling of the EM algorithm framework with the Bayesian models to estimate the hyper- and nuisance parameters via the marginal likelihood, and in turn, adopting the empirical partial Bayes approach for estimating the ideal color image data allowed us to apply heavy-tailed priors to the unobservable wavelet coefficients. Third, exploiting the reversed-order filterbank structure, a regression of luminance and difference image filterbank coefficients on the CFA image filterbank coefficients were simplified. The above estimation techniques were derived using second-order statistics for complete observation models.

Acknowledgments

The author would like to thank his wonderful collaborators, Dr. Thomas W. Parks in the Department of Electrical and Computer Science at Cornell University, Dr. Xiao-Li Meng in the Department of Statistics at Harvard University, and Dr. Patrick J. Wolfe in the School of Engineering and Applied Sciences at Harvard University, whose invaluable contributions



(a) (b) (c) (d) (e) (f)

FIGURE 9.8

Reconstruction of the *Peppers* image given a simulated noisy sensor data: (a) noise-free original color image, (b) color version of simulated noisy sensor data, (c) estimated with demosaicking method in Reference [7] and denoising method in [23], (d) estimated with the approach in Section 9.5, (e) estimated with the approach in Section 9.6, and (f) estimated with the approach in Section 9.7.

FIGURE 9.9

Reconstruction of the *Lena* image given a simulated noisy sensor data: (a) noise-free original color image, (b) color version of simulated noisy sensor data, (c) estimated with demosaicking method in Reference [7] and denoising method in [23], (d) estimated with the approach in Section 9.5, (e) estimated with the approach in Section 9.6, and (f) estimated with the approach in Section 9.7.

to the works in References [17], [40], [42], [49] are reflected in this chapter. His gratitude extends also to Dr. Bahadir Gunturk at Louisiana State University and Dr. Javier Portilla at Universidad de Granada for making their simulation codes available; and to Daniel Rudoy, Ayan Chakrabarti, and Prabahan Basu at Harvard University for their constructive criticisms.

References

[1] B.E. Bayer, "Color imaging array." U.S. Patent 3 971 065, July 1976.

[2] R. Lukac and K.N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1260–1267, November 2005.

[3] K. Parulski and K.E. Spaulding, *Digital Color Image Handbook*, ch. Color image processing for digital cameras, G. Sharma (ed.), Boca Raton, FL: CRC Press, 2002, pp. 727–757.

[4] S. Yamanaka, "Solid state camera." U.S. Patent 4 054 906, November 1977.

[5] M. Parmar and S.J. Reeves, "A perceptually based design methodology for color filter arrays," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Canada, May 2004, vol. 3, pp. 473–476.

[6] K. Hirakawa and T.W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 360–369, March 2005.

[7] B.K. Gunturk, Y. Altunbasak, and R.M. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 997–1013, September 2002.

[8] X. Wu and N. Zhang, "Primary-consistant soft-decision color demosaicking for digital cameras," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1263–1274, September 2004.

[9] B.K. Gunturk, J. Glotzbach, Y. Altunbasak, R.W. Schafer, and R.M. Mersereau, "Demosaicking: Color filter array interpolation in single chip digital cameras," *IEEE Signal Processing Magazine; Special Issue on Color Image Processing*, vol. 22, no. 1, pp. 44–54, January 2005.

[10] R. Lukac and K.N. Plataniotis, "Universal demosaicking for imaging pipelines with an RGB color filter array," *Pattern Recognition*, vol. 38, no. 11, pp. 2208–2212, November 2005.

[11] D. Alleysson, S. Süsstrunk, and J. Herault, "Linear demosaicing inspired by the human visual system," *IEEE Transactions on Image Processing*, vol. 14, no. 4, pp. 439–449, April 2005.

[12] L. Chang and Y.P. Tang, "Effective use of spatial and spectral correlations for color filter array demosaicking," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 355–365, May 2004.

[13] R. Kakarala and Z. Baharav, "Adaptive demosaicing with the principal vector method," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 4, pp. 932–937, November 2002.

[14] W. Lu and Y.P. Tan, "Color filter array demosaicking: new method and performance measures," *IEEE Transactions on Image Processing*, vol. 12, no. 10, pp. 1194–1210, October 2003.

[15] D.D. Muresan and T.W. Parks, "Demosaicing using optimal recovery," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 267–278, February 2005.

[16] R. Ramanath and W.E. Snyder, "Adaptive demosaicking," *Journal of Electronic Imaging*, vol. 12, no. 4, pp. 633–642, October 2003.

[17] K. Hirakawa, X.L. Meng, and P.J. Wolfe, "A framework for wavelet-based analysis and processing of color filter array images with applications to denoising and demosaicking," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Honolulu, HI, USA, April 2007, vol. 1, pp. 597–600.

[18] M.S. Crouse, R.D. Nowak, and R.G. Baraniuk, "Bayesian tree-structured image modeling using wavelet-domain hidden Markov models," *IEEE Transactions on Image Processing*, vol. 46, no. 7, pp. 1056–1068, July 1998.

[19] D.L. Donoho and I.M. Johnstone, "Ideal spatial adaptation via wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, September 1994.

[20] M. Jansen and A. Bultheel, "Empirical Bayes approach to improve wavelet thresholding for image noise reduction," *Journal of American Statistical Association*, vol. 96, no. 454, pp. 629–639, June 2001.

[21] I.M. Johnstone and B.W. Silverman, "Wavelet threshold estimators for data with correlated noise," *Journal of Royal Statistical Society - Series B*, vol. 59, no. 2, pp. 319–351, 1997.

[22] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli, "Image denoising using scale mixture of Gaussians in the wavelet domain," Tech. Rep. TR2002-831, Comput. Sci. Dept., Courant Inst. Math. Sci., New York Univ., 2002.

[23] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli, "Image denoising using scale mixture of Gaussians in the wavelet domain," *IEEE Transactions on Image Processing*, vol. 12, no. 11, pp. 1338–1351, November 2003.

[24] A. Pizurica, W. Philips, I. Lemahieu, and M. Acheroy, "A joint inter and intrascale statistical model for Bayesian wavelet based image denoising," *IEEE Transactions on Image Processing*, vol. 11, no. 5, pp. 545–557, May 2002.

[25] L. Sendur and I.W. Selesnick, "Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency," *IEEE Transactions on Signal Processing*, vol. 50, no. 11, pp. 2744–2756, November 2002.

[26] L. Sendur and I.W. Selesnick, "Bivariate shrinkage with local variance estimation," *IEEE Signal Processing Letters*, vol. 9, no. 12, pp. 438–441, December 2002.

[27] J.L. Starck, D.L. Donoho, and E. Cande, "Very high quality image restoration," in *Proceedings of the SPIE Conference on Wavelet and Applications in Signal and Image Processing*, San Diego, CA, USA, July 2001, vol. 4478, pp. 9–19.

[28] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the IEEE International Conference on Computer Vision*, Bombay, India, January 1998, pp. 839–846.

[29] G. Hua and M. T. Orchard, "A new interpretation of translation invariant denoising," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Montreal, Canada, May 2004, vol. 3, pp. 189–192.

[30] X. Li and M.T. Orchard, "Spatially adaptive image denoising under overcomplete expansion," in *Proceedings of the IEEE International Conference on Image Processing*, Vancouver, BC, Canada, September 2000, vol. 3, pp. 300–303.

[31] D.D. Muresan and T.W. Parks, "Adaptive principal components and image denoising," in *Proceedings of the IEEE International Conference on Image Processing*, Barcelona, Spain, September 2003, vol. 1, pp. 101–104.

[32] K. Hirakawa and T.W. Parks, "Image denoising for signal-dependent noise," *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2730–2742, September 2006.

[33] D.D. Muresan and T.W. Parks, "Adaptively quadratic (AQua) image interpolation," *IEEE Transactions on Image Processing*, vol. 13, no. 5, pp. 690–698, May 2004.

[34] H. Tian, B. Fowler, and A.E. Gamal, "Analysis of temporal noise in CMOS photodiode active pixel sensor," *IEEE Journal on Solid State Circuits*, vol. 36, no. 1, pp. 92–101, January 2001.

[35] G.E. Healey and R. Kondepudy, "Radiometric CCD camera calibration and noise estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, pp. 267–276, March 1994.

[36] R. Ding and A.N. Venetsanopoulos, "Generalized homomorphic and adaptive order statistic filters for the removal of impulsive and signal-dependent noise," *IEEE Transactions on Circuits Systems*, vol. 34, no. 8, pp. 948–955, August 1987.

[37] A.V. Oppenheim, R.W. Schafer, and J.R. Buck, *Discrete-Time Signal Processing*. Upper Saddle River, NJ: Prentice-Hall, 2nd Edition, 1999.

[38] P. Fryzlewicz and G.P. Nason, "A Haar-Fisz algorithm for Poisson intensity estimation," *Journal of Computational and Graphical Statistics*, vol. 13, no. 3, pp. 621–638, September 2004.

[39] P. Fryzlewicz and G.P. Nason, "Smoothing the wavelet periodogram using the Haar-Fisz transform," *Scientific Charge*, 2001.

[40] K. Hirakawa and T.W. Parks, "Joint demosaicing and denoising," *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2146–2157, August 2006.

[41] M. Raphan and E.P. Simoncelli, *Advances in Neural Information Processing System*, ch. Learning to be Bayesian without Supervision. Cambridge, MA: MIT Press, vol. 19, 2007.

[42] K. Hirakawa, "Signal-dependent noise characterization in wavelets domain," in *Proceedings of the SPIE Conference on Optics & Photonics*, San Diego, CA, USA, August 2007.

[43] D.R. Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal," U.S. Patent 4 642 678, February 1987.

[44] R. Kimmel, "Demosaicing: Image reconstruction from color CCD samples," *IEEE Transactions on Image Processing*, vol. 8, no. 9, pp. 1221–1228, September 1999.

[45] R. Lukac, K.N. Plataniotis, D. Hatzinakos, and M. Aleksic, "A novel cost effective demosaicing approach," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 256–261, February 2004.

[46] R. Lukac and K.N. Plataniotis, "Normalized color-ratio modeling for CFA interpolation," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 737–745, May 2004.

[47] E. Dubois, "Filter design for adaptive frequency-domain Bayer demosaicking," in *Proceedings of the IEEE International Conference on Image Processing*, Atlanta, GA, USA, October 2006, pp. 2705–2708.

[48] M.J.T. Smith and T.P. Barnwell, "A procedure for designing exact reconstruction filter banks for tree structured subband coders," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, San Diego, CA, USA, March 1984, pp. 27:1.1–1.4.

[49] K. Hirakawa and X.L. Meng, "An empirical Bayes EM-wavelet unification for simultaneous denoising, interpolation, and/or demosaicing," in *Proceedings of the IEEE International Conference on Image Processing*, Atlanta, GA, USA, October 2006, pp. 1453–1456.

[50] G.J. McLachelan and T. Krishnan, *The EM Algorithm and Extentions*. New York: John Wiley & Sons, 1997.

10

Automatic White Balancing in Digital Photography

Edmund Y. Lam and George S. K. Fung

10.1 Introduction

Color constancy is one of the most amazing features of the human visual system. When we look at objects under different illuminations, their colors stay relatively constant. This helps humans to identify objects conveniently. While the precise physiological mechanism is not fully known, it has been postulated that the eyes are responsible for capturing different wavelengths of the light reflected by an object, and the brain attempts to "discount" the contribution of the illumination so that the color perception matches more closely with the object reflectance, and therefore is mostly constant under different illuminations [1].

A similar behavior is highly desirable in digital still and video cameras. This is achieved via white balancing which is an image processing step employed in a digital camera imaging pipeline (detailed description of the camera imaging pipeline can be found in Chapters 1 and 3) to adjust the coloration of images captured under different illuminations [2], [3].

This is because the ambient light has a significant effect on the color stimulus. If the color temperature of a light source is low, the object being captured will appear reddish. An example is the domestic tungsten lamp, whose color temperature is around 3000 Kelvins (K). On the other hand, with a high color temperature light source, the object will appear bluish. This includes the typical daylight, with color temperature above 6000 K [4].

Various manual and automatic methods exist for white balancing. For the former, the camera manufacturer often has predefined settings for typical lighting conditions such as sunlight, cloudy, fluorescent, or incandescent. The user only needs to make the selection, and the camera will compute the adjustment automatically. Higher-end cameras, such as prosumer (professional-consumer) and single-lens reflex (SLR) digital cameras, would even allow the user to define his or her own white balance reference. Most amateur users, however, prefer automatic white balancing. The camera then needs to be able to dynamically detect the color temperature of the ambient light and compensate for its effects, or determine from the image content the necessary color correction due to the illumination. The automatic white balancing (AWB) algorithm employed in the camera imaging pipeline is thus critical to the color appearance of digital pictures. This chapter is devoted to a study of such algorithms commonly used in digital photography.

We organize this chapter as follows. In Section 10.2, we first briefly review the human visual system and the theory of color, which are necessary background materials for our discussion on AWB strategies in cameras. Certain terminologies would also be introduced that are commonly used in discussing color. This is followed by looking into the physical principles of color formation on an electronic sensor. The challenges that exist in digital photography are described in Section 10.3. Then, in Section 10.4, we describe a few representative AWB algorithms. Our goal is not to be encyclopedic, which is rather impossible considering the wide array of methods in existence and the proprietary nature of some of these schemes, but to be illustrative of the main principles behind the major approaches. In Section 10.5, experimental results of some of these representative algorithms are presented to evaluate and compare the efficacy of various techniques. Some concluding remarks are given in Section 10.6.

10.2 Human Visual System and Color Theory

It is instructive for us to begin the discussion on color with the physics of light and the physiology of the human visual system. The primary reason is that humans are typically the end-user and the judge of the images in our camera systems. A secondary reason is that the eye is itself a complex and beautifully made organ that acts as an *image capturing device* for our brain. In many cases, we model our camera system design on the natural design of our eyes.

Visible light occupies a small section of the electromagnetic spectrum, which we call the visible spectrum. In the seventeenth century, Sir Isaac Newton (1642–1727) was the first to demonstrate that when a white beam entered a prism, due to the law of refraction of light the exit beam would consist of shades of different colors. Further experimentation and

measurement show that different colors correspond to electromagnetic waves of different wavelengths, usually denoted with the symbol λ and measured in nanometers (nm). Visible spectrum roughly spans from $\lambda = 400\,\text{nm}$ to $\lambda = 700\,\text{nm}$.

Our eyes interpret different colors based on the wavelength of the electromagnetic wave. For example, at $\lambda \approx 400\,\text{nm}$ we have the sensation of blue; at $\lambda \approx 550\,\text{nm}$ we have the sensation of green; at $\lambda \approx 700\,\text{nm}$ we have the sensation of red. For a range of $\lambda < 400\,\text{nm}$, the region is called ultra-violet (UV), while for a range of $\lambda > 700\,\text{nm}$, we refer to it as infra-red (IR). UV and IR are gaining importance in imaging, particularly in medical imaging and remote sensing respectively, but our focus with digital camera systems is on the visible spectrum. Hence we will focus on the range approximately from $400\,\text{nm} < \lambda < 700\,\text{nm}$ hereafter.

10.2.1 Illumination

Imaging begins with the source of light called *illumination*. Virtually all illuminations consist of light with multiple wavelengths. (In fact, we have to go to extraordinary lengths to create lasers that are of a single wavelength or a very narrow bandwidth. Imaging under such circumstances is rare for digital photography and is mostly for scientific research purposes, and thus we do not take them into account here.) Each illumination then is described with a curve showing the strength of the electromagnetic radiations at different values of λ. If we normalize the curves of various illuminations, the result is a very useful description of the *spectral power distribution* of illumination as a function of wavelength. We can then compare the representative spectral power distributions of various common light sources. Note that we usually only describe the general characteristics of the illuminations; an actual measurement for sunlight, for example, would depend on the location, altitude, and atmospheric and weather conditions during the measurement.

As an example, Figure 10.1[1] shows the spectral power distribution of various common illumination sources. Figure 10.1a is the curve for typical sunlight, which is continuous (although not uniform) over the visible spectrum. Tungsten light, as shown in Figure 10.1b, also appears to be rather smooth. Later, we will see why sunlight tends to be perceived as bright yellowish-white, while tungsten usually gives us a sensation of yellowish hue. In contrast, the fluorescent lamp consists of sharp spikes in the spectral power distribution, as shown in Figure 10.1c. Finally, we see that with a light-emitting diode (LED), the spectral power distribution is also smooth, but is often limited to a narrow range of the visible spectrum as depicted in Figure 10.1d. LEDs with spectral power concentrating at the higher wavelength region are more common, and as a result we have mostly red LEDs.

In this chapter we denote the spectral power distribution of an illumination as $I(\lambda)$. This is an important quantity as we relate this with other color production factors to be explained next. In general, there are numerous possible curves $I(\lambda)$. It is also possible to express $I(\lambda)$

[1]Data for some of the spectral power distributions and spectral reflectance can be obtained online from the Munsell Color Science Laboratory, Chester F. Carlson Center for Imaging Science at Rochester Institute of Technology.

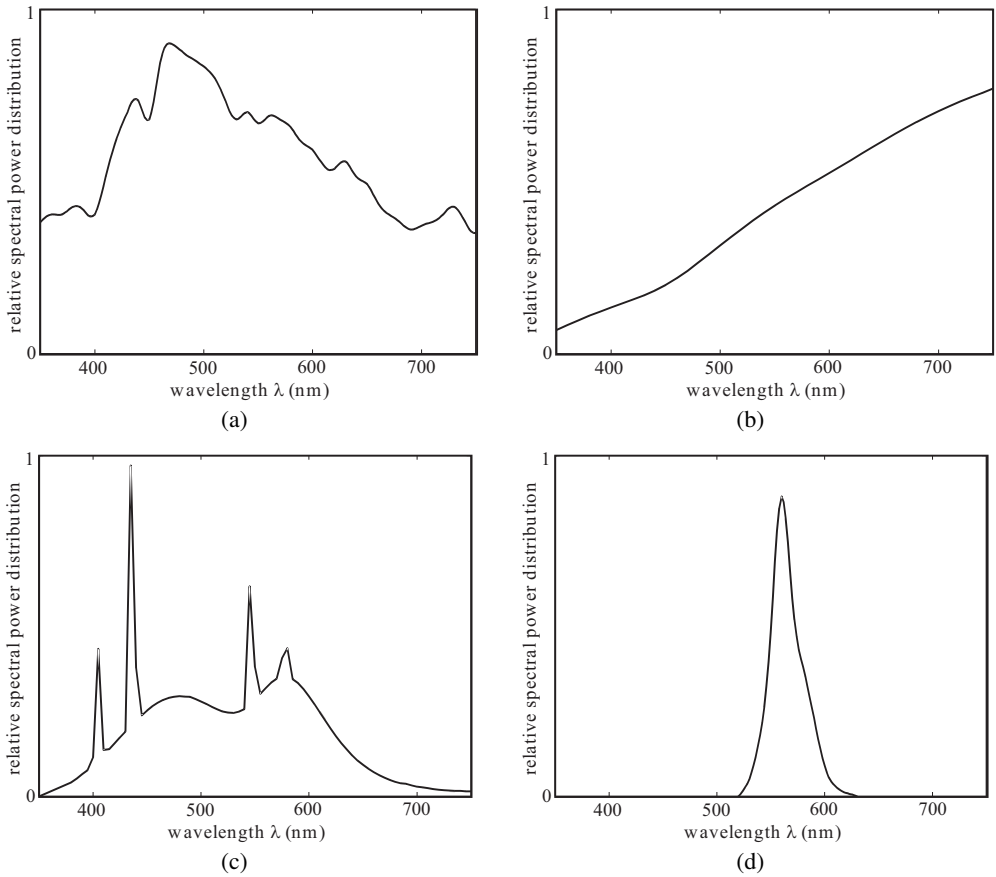


FIGURE 10.1

Spectral power distribution of various common types of illuminations: (a) sunlight, (b) tungsten light, (c) fluorescent light, and (d) light-emitting diode (LED).

as a linear combination of known basis functions $I_j(\lambda)$ with

$$I(\lambda) = \sum_{j=1}^{m} \alpha_j I_j(\lambda), \tag{10.1}$$

where, for example, three basis functions (corresponding to $m = 3$) are sufficient to represent standard daylights [5]. This property can be used in the design of AWB algorithms.

10.2.2 Object

Illumination is one of the three main factors contributing to the sensation of color in our brains. The second major factor is the object. When the electromagnetic radiations from the illumination reach an object, they are partially absorbed, and partially reflected or transmitted (for transparent objects). For different items, the proportion of the reflection or transmission varies with wavelengths, but this is an inherent property of the object irrespective of the illumination that takes place. We can therefore characterize an object's *spectral reflectance* or *spectral transmittance* as a function of wavelength for comparison.

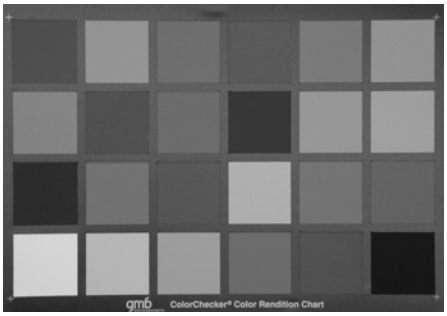


FIGURE 10.2 (See color insert.)

GretagMacbeth color rendition chart.

To illustrate, we plot the spectral reflectance corresponding to several typical object colors. These colors represent patches taken from the GretagMacbeth Color Checker (Figure 10.2) which is often used to test digital camera performances. The spectral reflectance plots are shown in Figure 10.3. For each plot, the y-axis denotes the fraction of light that is being reflected from the object. Figure 10.3a to Figure 10.3d show the spectral reflectance of a red, light blue, yellow, and gray patches of color, respectively. As expected, a red patch absorbs most of the greenish-blue frequencies and reflects most of the higher-wavelength light that gives the sensation of red. However, it should be noted that some residual lower-wavelength frequencies are also reflected, only that the amount is much smaller. We observe a similar behavior for the light blue and yellow patches as well. For the gray patch, the spectral reflectance is roughly a constant for the different wavelengths, causing the resulting gray sensation to be neutral in color.

We denote the spectral reflectance of an object with $R(\lambda)$. Similarly, we can also define the spectral transmittance of an object with $T(\lambda)$. There are also attempts to decompose the spectral reflectance into a summation of known basis functions $R_j(\lambda)$ such that

$$R(\lambda) = \sum_{j=1}^{m} \beta_j R_j(\lambda). \tag{10.2}$$

It has been shown that three basis functions (corresponding to $m = 3$) can accurately represent 433 Munsell-chips reflectance functions [6], and seven basis functions (corresponding to $m = 7$) are sufficient for a large number of natural objects [7].

10.2.3 Color Stimulus

Illumination and object reflectance or transmittance together determine the *color stimulus*. The spectral power distribution of the illumination governs how much energy is incident on the object at every wavelength. For a reflective object, the spectral reflectance dictates what fraction of that radiation is reflected and will arrive at the eye or the sensor, again at every wavelength. Similarly, for a transmissive object, the spectral transmittance determines the fraction of the radiation being transmitted through the object. Therefore, the spectral power distribution of an object is the *product* of the spectral power of the illumination and the spectral reflectance of the object. This is also called the color stimulus.

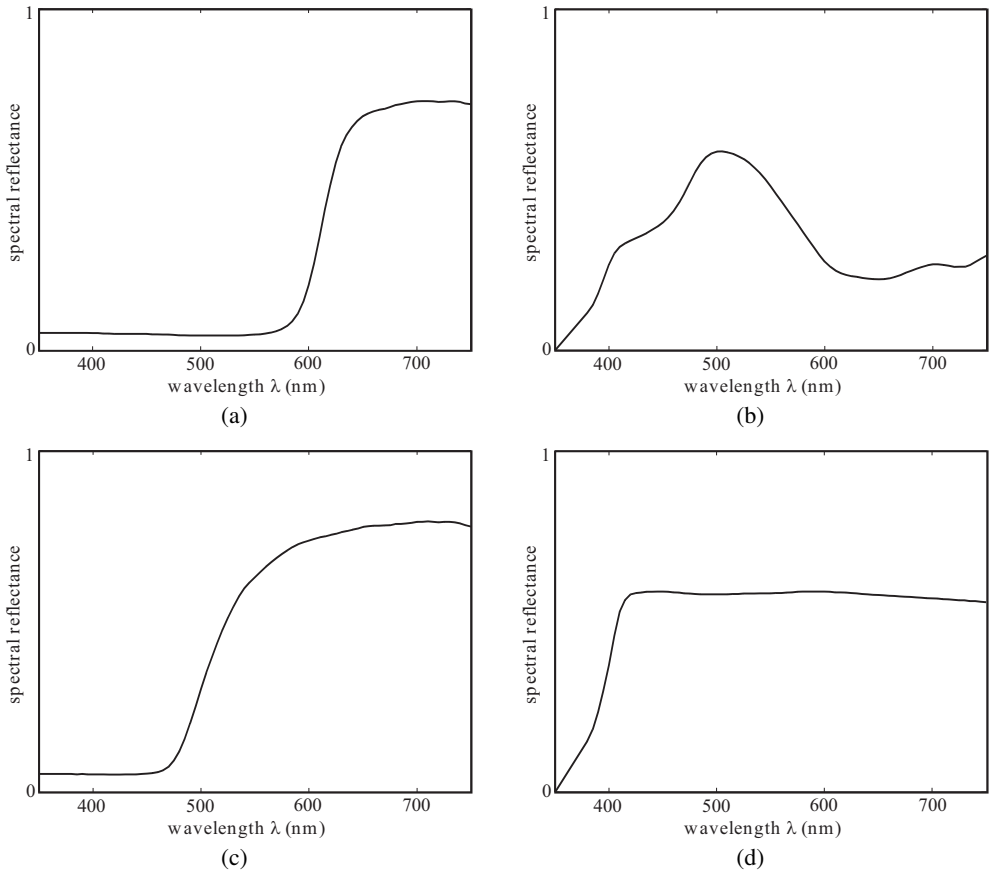


FIGURE 10.3

Spectral reflectance of various color patches: (a) red patch, (b) light blue patch, (c) yellow patch, (d) gray patch.

Mathematically, we denote the color stimulus with $S(\lambda)$. This is related to the illumination $I(\lambda)$ and object reflectance $R(\lambda)$ by [8]

$$S(\lambda) = I(\lambda)R(\lambda). \tag{10.3}$$

If we use the basis decomposition in Equations 10.1 and 10.2, this becomes

$$S(\lambda) = \left(\sum_{j=1}^{m} \alpha_j I_j(\lambda)\right)\left(\sum_{k=1}^{n} \beta_k R_k(\lambda)\right) \tag{10.4}$$

$$= \sum_{j=1}^{m}\sum_{k=1}^{n} \alpha_j \beta_k I_j(\lambda) R_k(\lambda). \tag{10.5}$$

Although the above equations appear deceptively simple, they underscore an important fact in color science that we must emphasize here. The color stimulus depends on both the illumination and the object. Therefore, given any object, we can theoretically manipulate the illumination so that it produces any desired color stimulus! We will see shortly that the

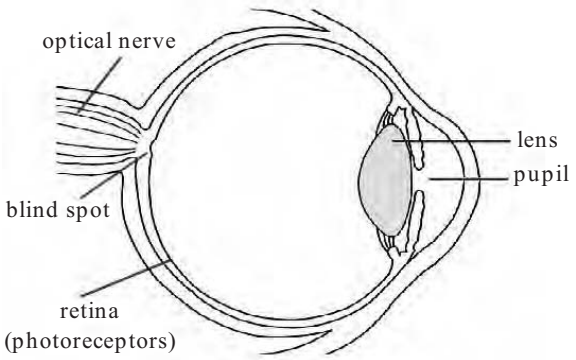


FIGURE 10.4

An anatomy of a human eye.

color stimulus in turn contributes to our perception of color. One noteworthy corollary is that color is not an inherent feature of an object. A red object, for example, can be perceived as blue by a clever design of the illumination.

In other words, the illumination is as important as the object reflectance. This fact is very important for our camera system design. When we take a picture of the same object first under sunlight and then under fluorescent light, for example, the color stimuli vary significantly due to differences in illumination. We must adapt our camera to interpret the color stimuli differently, or otherwise the color of the photographs would look very different. This explains why AWB is so critical and challenging. However, before we can proceed on discussing AWB algorithms, we need to explore how our eyes interpret the color stimuli first. This brings us to the third major factor contributing to the color sensation: the physiology of our eyes.

10.2.4 Human Visual System

The color stimulus is a function of wavelength. If we had to faithfully reproduce the color stimulus, there is a lot of information to be stored. A key fact from color science is that there is no need to reproduce the entire spectral distribution in color reproduction. As a matter of fact, we often need only three values to specify a color, despite the obvious loss in information and possibility of ambiguity. The reason lies in our human visual system.

A basic anatomy of a human eye is shown in Figure 10.4. Visible light enters through the pupil and is refracted by the lens to create an image on the retina. The optical nerve transfers the image on the retina to the brain for interpretation. The retina is able to form an image because of tiny sensors called photoreceptors. For a normal person, there are two types of photoreceptors, cones and rods, which function very differently.

The rods are responsible for scotopic, or dim-light, vision. We have somewhere between 75 and 150 million rods in each of our eyes, and they are distributed all over the retina. Their chief aim is to give us an overall picture of the field of view of our eyes, rather than for color vision. When we enter a room that is rather dark, we may still be able to see objects even though they tend to be colorless. This is because under such illumination, only the rods are able to give us the images.

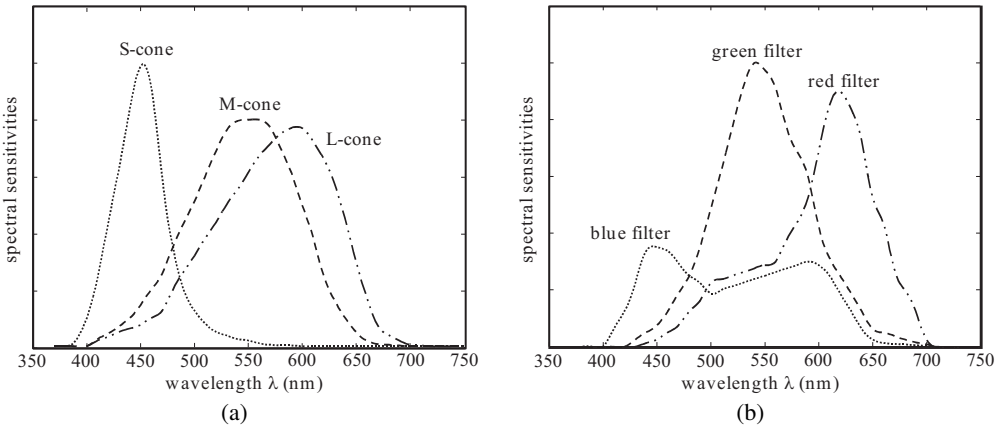


FIGURE 10.5

Spectral sensitivities of: (a) the three types of cones in a human eye, and (b) a typical digital camera.

The cones, on the other hand, are highly sensitive to color. There are far fewer cones in our eyes; an average person has about six to seven million only. They are also localized at a place called the fovea, rather than distributed all over the retina. They help us resolve fine details in images, and are responsible for photopic, or bright-light, vision. More importantly, there are three types of cones:

- *L-cones* which have peak sensitivity towards the long wavelength section of the visible spectrum,
- *M-cones* which have peak sensitivity towards the middle wavelength section of the visible spectrum, and
- *S-cones* which have peak sensitivity towards the short wavelength section of the visible spectrum.

These three types of cones together give us the sensation of color vision. When one or more of the cone types are defective, those people are said to possess what we collectively refer to as color deficiency or color blindness. Approximately one in twelve men has this condition to varying degrees, and this is more common in men than in women.

Figure 10.5a shows the spectral sensitivities of the three types of cones of the human eye. In subsequent discussions, we use $l(\lambda)$, $m(\lambda)$, and $s(\lambda)$ to denote the spectral sensitivity responses of the L-, M-, and S-cones respectively. For the sake of comparison, the curves have been normalized to equal area. It is interesting to observe that they do not cover disjoint sections of the visible spectrum, nor do they cover it entirely. In fact, the responses of L-cones and M-cones overlap significantly, and all three curves show low response to stimulus below around 400 nm and above around 650 nm. As we will see in the next section, camera designs mimic the responses of our human eyes. The sensor in the camera consists of three filters, typically red, green, and blue filters. The spectral sensitivities of a typical camera are shown in Figure 10.5b. We can observe that the peaks of these filters correspond to the peaks of the L-, M-, and S-cones of our eyes.

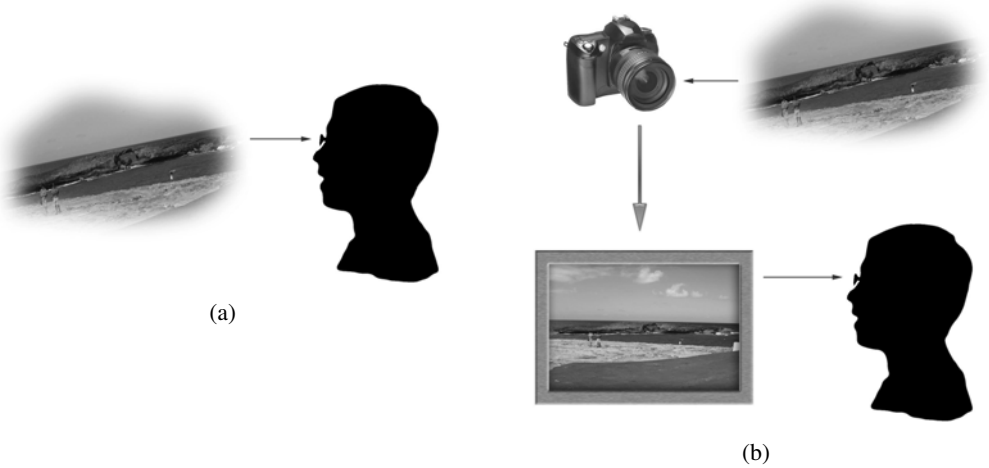


FIGURE 10.6

The aim of photography. Observing an object: (a) directly through a human eye, and (b) indirectly through a photograph.

When an object with stimulus $S(\lambda) = I(\lambda)R(\lambda)$ is observed, each of the three cones responds to the stimulus by summing up the reaction at all wavelengths. Therefore, three values are produced from the three cones, in accordance with the equations:

$$\begin{aligned} X &= \int_{400}^{700} l(\lambda)I(\lambda)R(\lambda)\,\mathrm{d}\lambda \\ Y &= \int_{400}^{700} m(\lambda)I(\lambda)R(\lambda)\,\mathrm{d}\lambda. \\ Z &= \int_{400}^{700} s(\lambda)I(\lambda)R(\lambda)\,\mathrm{d}\lambda \end{aligned} \tag{10.6}$$

The triplet (X,Y,Z) is called trichromatic response. Despite its simplicity to describe color, it is estimated that humans are capable of resolving about 10 million color sensations!

An important consequence of trichromatic response is that in the digital camera, we only require three numbers at each pixel to capture the color information. We do not need to record the color stimulus at all wavelengths! In fact, this gives rise to a useful phenomenon: Even if we consider the spectral sensitivities to be known, for any given triplet (X,Y,Z) there could be an infinite number of possibilities for the color stimulus according to Equation 10.6. Two stimuli that produce the same trichromatic response are called metamers. The possibility of metamers is key to color photography.

10.2.5 Color Matching

The goal of photography is a bit different from the way our eyes perceive the color of an object. Consider the two scenarios depicted in Figure 10.6. In Figure 10.6a, our eyes observe a certain color object. In Figure 10.6b, our camera captures the object, and in turn

produces an image on screen or in a hardcopy. Our eyes then observe that object. Note that through the capturing device, because the spectral sensitivities of the camera differ from our eyes and the ink in the hardcopy differs in reflectance from the object, the print is not of identical color to the original object. The goal of color photography is to make the image appear as similar to the object as possible.

As such, our aim is *color matching*. Consider digital images shown on a screen, such as using cathode-ray tube (CRT), liquid crystal display (LCD), or even organic light-emitting device (OLED). In these cases, each pixel consists of three color patches called primaries, which are usually red, green, and blue. Color is formed from a linear combination of intensities from these primaries. Each primary is associated with a certain color stimulus, which we denote as $P_r(\lambda)$, $P_g(\lambda)$, and $P_b(\lambda)$ for the three colors.

It is not difficult to realize that we only need to perform color matching on monochromatic light sources. Any real stimulus, caused by any real illumination reflected or transmitted through any object, can be decomposed as a linear combination of these single-wavelength light sources. Mathematically, we assume that our light source has the color stimulus

$$S(\lambda) = \delta(\lambda - \lambda_0), \tag{10.7}$$

which indicates that it has unit strength at wavelength λ_0 and zero elsewhere. We assign scalar weights l_0, m_0, and s_0, respectively, to the three primaries $P_r(\lambda)$, $P_g(\lambda)$, and $P_b(\lambda)$ in matching colors. Note further that the color matching must be performed with respect to an observer. It is common to define a standard observer, with a particular set of spectral sensitivities $l(\lambda)$, $m(\lambda)$, and $s(\lambda)$.

Equipped with all these parameters, we can now calculate the tristimulus value of directly observing the original stimulus with unit strength at wavelength λ_0 to be

$$\begin{aligned} X &= \int_{400}^{700} l(\lambda) S(\lambda)\, \mathrm{d}\lambda = l(\lambda_0) \\ Y &= \int_{400}^{700} m(\lambda) S(\lambda)\, \mathrm{d}\lambda = m(\lambda_0). \\ Z &= \int_{400}^{700} s(\lambda) S(\lambda)\, \mathrm{d}\lambda = s(\lambda_0) \end{aligned} \tag{10.8}$$

The tristimulus value of indirect observation, through the three primaries of our display device, would be

$$\begin{aligned} \hat{X} &= \int_{400}^{700} l(\lambda) \left[l_0 P_r(\lambda) + m_0 P_g(\lambda) + s_0 P_b(\lambda) \right] \mathrm{d}\lambda \\ \hat{Y} &= \int_{400}^{700} m(\lambda) \left[l_0 P_r(\lambda) + m_0 P_g(\lambda) + s_0 P_b(\lambda) \right] \mathrm{d}\lambda. \\ \hat{Z} &= \int_{400}^{700} s(\lambda) \left[l_0 P_r(\lambda) + m_0 P_g(\lambda) + s_0 P_b(\lambda) \right] \mathrm{d}\lambda \end{aligned} \tag{10.9}$$

To match the color, we require only that the tristimulus values match, i.e., $X = \hat{X}$, $Y = \hat{Y}$, and $Z = \hat{Z}$. Equating Equations 10.8 and 10.9, we have

$$\underbrace{\begin{bmatrix} \int P_r(\lambda)l(\lambda)\,d\lambda & \int P_g(\lambda)l(\lambda)\,d\lambda & \int P_b(\lambda)l(\lambda)\,d\lambda \\ \int P_r(\lambda)m(\lambda)\,d\lambda & \int P_g(\lambda)m(\lambda)\,d\lambda & \int P_b(\lambda)m(\lambda)\,d\lambda \\ \int P_r(\lambda)s(\lambda)\,d\lambda & \int P_g(\lambda)s(\lambda)\,d\lambda & \int P_b(\lambda)s(\lambda)\,d\lambda \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} l_0 \\ m_0 \\ s_0 \end{bmatrix}}_{\mathbf{v_c}} = \underbrace{\begin{bmatrix} l(\lambda_0) \\ m(\lambda_0) \\ s(\lambda_0) \end{bmatrix}}_{\mathbf{v}}. \tag{10.10}$$

This equation is fundamental to color science. Several remarks can be made for the matrix equation above:

1. The vector $\mathbf{v_c}$ is the only unknown in the above equation. With three equations and three unknowns, the solution is unique provided that the matrix P is not singular.

2. The process can be repeated for different values of λ_0. If we view l_0 above not as a scalar but as a value of the curve $x(\lambda)$ at $\lambda = \lambda_0$, by repeating the process at different wavelengths we can generate the entire curve of $x(\lambda)$. Similarly, we generate $y(\lambda)$ from m_0 and $z(\lambda)$ from s_0. These curves are called color-matching functions.

3. The color-matching functions depend on the primaries ($R(\lambda)$, $G(\lambda)$, and $B(\lambda)$) and the observer ($l(\lambda)$, $m(\lambda)$, and $s(\lambda)$). Changing either, or both, of these quantities would result in new color-matching functions.

4. The vector $\mathbf{v}$ is fixed for the same observer. In this case, the color-matching functions with different primaries are simply linear combinations of one another. Therefore, the two sets of color-matching can be described as $P_1\mathbf{v_{c1}} = \mathbf{v}$ and $P_2\mathbf{v_{c2}} = \mathbf{v}$, and thus

$$\mathbf{v_{c1}} = P_1^{-1}P_2\mathbf{v_{c2}}. \tag{10.11}$$

5. One particularly useful set of color-matching functions defined by the Commission Internationale de l'Éclairage (CIE), called the CIE Standard Colorimetric Observer color-matching functions, is shown in Figure 10.7. They are used in the calculation of the CIE tristimulus values X, Y, and Z, which quantify the trichromatic characteristics of color stimuli [8].

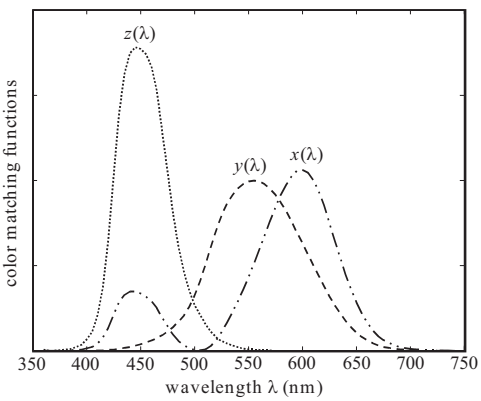


FIGURE 10.7

The CIE standard colorimetric observer color-matching functions.

10.3 Challenges in Automatic White Balancing

In the previous section, we have discussed how the color stimulus is equally dependent on the illumination and the object reflectance. Mathematically, we would thus expect the color stimulus of the same object to be different under different lighting conditions. However, our experience seems to the contrary: the same object appears to be of the same color even under different illuminations. This is known as *color constancy*. It is also known that the human visual system corrects for the prevailing scene illumination [9], [10]. However, for digital cameras, this is a challenging engineering problem.

Digital cameras nowadays use a single-image sensor, with a mosaic of color filters on top of each photodetector. For details refer to Chapters 1 and 5. These filters can be fabricated as a photoresist layer mixed with the red, green, or blue dyes [11], with spectral sensitivities such as those shown in Figure 10.5b.

When an object with stimulus $S(\lambda) = I(\lambda)R(\lambda)$ is observed, each filter responds to the stimulus by summing up the reaction at all wavelengths. Therefore, three values are produced from the three filters, in accordance with the equation

$$\begin{aligned} R_{\text{sensor}} &= \int_{400}^{700} r(\lambda)I(\lambda)R(\lambda)\,\mathrm{d}\lambda \\ G_{\text{sensor}} &= \int_{400}^{700} g(\lambda)I(\lambda)R(\lambda)\,\mathrm{d}\lambda, \\ B_{\text{sensor}} &= \int_{400}^{700} b(\lambda)I(\lambda)R(\lambda)\,\mathrm{d}\lambda \end{aligned} \tag{10.12}$$

where $r(\lambda)$, $g(\lambda)$, and $b(\lambda)$ refer to the spectral sensitivities of the sensors under the red, green, and blue filters respectively. This equation is essentially identical to Equation 10.6 except that we are now concerned with the spectral sensitivities of the camera sensors rather than our cone responses in our eyes. We can now state our AWB goal as follows: we seek to minimize the effect of $I(\lambda)$ and ensure that R_{sensor}, G_{sensor}, and B_{sensor} correlate with the object reflectance $R(\lambda)$ only [12].

The solution, however, involves dealing with an underdetermined set of equations. We can count the number of variables and equations as follows. For simplicity, assume for the moment that our sensors do not rely on demosaicking to recover the full RGB image. Hence, for an image of size $n \times n$, we have n^2 pixels and therefore $3n^2$ captured values. From these known values, we want to estimate parameters for the n^2 pixels together with the illuminant. Assume we discretize Equation 10.12 above so that

$$\begin{aligned} R_{\text{sensor}} &= \sum_{j=1}^{m} r(\lambda_j)I(\lambda_j)R(\lambda_j)\,\Delta\lambda \\ G_{\text{sensor}} &= \sum_{j=1}^{m} g(\lambda_j)I(\lambda_j)R(\lambda_j)\,\Delta\lambda. \\ B_{\text{sensor}} &= \sum_{j=1}^{m} b(\lambda_j)I(\lambda_j)R(\lambda_j)\,\Delta\lambda \end{aligned} \tag{10.13}$$

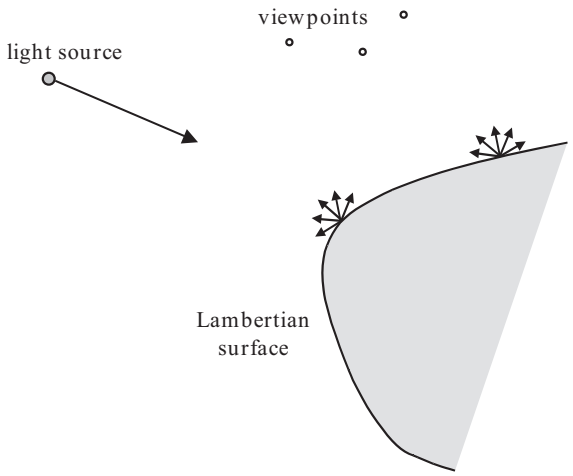


FIGURE 10.8

The Lambertian reflection model.

We are then using m sample points to represent the integral. Thus, for each pixel we want to derive $R(\lambda_j)$ for m values, hence there are a total of mn^2 unknowns. In addition, we have m unknowns for the illuminant. Thus, comparing $3n^2$ known values with $mn^2 + m = m(n^2 + 1)$ unknowns, it is clear that we do not have sufficient equations [12].

Moreover, we also need to note that Equations 10.12 and 10.13 above correspond to a simplified two-dimensional world in which all objects are flat, matte, Lambertian surfaces, and uniformly illuminated [12]. A Lambertian, or diffuse, surface assumes that light energy reaching a surface is reflected evenly in all directions [13], as shown in Figure 10.8. Thus, a planar patch appears to be of uniform brightness for all visible viewpoints. This occurs when the surface is rough enough relative to the wavelength of the light. Otherwise, considerations such as flare will substantially complicate the problem further.

To deal with the underconstrained nature of this problem, we often make additional assumptions about the world. Many of the AWB techniques to be mentioned in the following section rely on particular assumptions. For instance, the gray world method, as the name implies, considers that the average intensity of the scene is gray. The white patch method assumes there are always some white pixels in the image. Different assumptions thus lead to different implementations, and the efficacy of various AWB algorithms can be judged from how well the actual scenes satisfy the prior assumptions.

10.4 Automatic White Balancing Algorithms

For the above discussions, it can be seen that ideally AWB techniques require information about the camera being used, and possibly are based on assumptions about the statistical properties of the expected illumination and spectral reflectance [14]. In practice, many AWB algorithms follow a two-stage process:

1. *Illuminant estimation* — This may be done explicitly, often choosing from a known set of possible illuminants, or implicitly with assumptions about the effect of such illuminants.

2. *Image color correction* — This generates a new image as if it had been taken under a standard illuminant. The correction is often achieved through an independent gain regulation of the three color signals. This is known as the Von Kries hypothesis [15]. Commonly, it is achieved by adjusting the intensities of red and blue only, as AWB is concerned about the ratio of the three color signals.

Below we discuss several representative algorithms. We present them independently, but we should also note that combination techniques exist (e.g., Reference [14]) where multiple algorithms are run simultaneously, and a consensus decision is required afterwards to select the best results.

10.4.1 Gray World

The first method incorporates the gray world assumption, which argues that the average reflectance of a scene is achromatic. In other words, the mean of the red (R_{sensor}), green (G_{sensor}), and blue (B_{sensor}) channels in a given scene should be roughly equal. This method has its root in film photography, where for the negatives the average is biased towards dark regions of the scene, which tend to be neutral [4]. Algorithmically, as stated above we can adjust a gain factor to two of the channels so that both their means are now equal to the reference channel, which is often taken to be green.

We denote a full-color image of size $n \times n$ as $RGB_{\text{sensor}}(x,y)$, where x and y denote the indices of the pixel position. The individual red, green, and blue color components are then $R_{\text{sensor}}(x,y)$, $G_{\text{sensor}}(x,y)$, and $B_{\text{sensor}}(x,y)$, respectively. We compute

$$\begin{aligned} R_{\text{avg}} &= \frac{1}{n^2}\sum_{x=1}^{n}\sum_{y=1}^{n} R_{\text{sensor}}(x,y) \\ G_{\text{avg}} &= \frac{1}{n^2}\sum_{x=1}^{n}\sum_{y=1}^{n} G_{\text{sensor}}(x,y). \\ B_{\text{avg}} &= \frac{1}{n^2}\sum_{x=1}^{n}\sum_{y=1}^{n} B_{\text{sensor}}(x,y) \end{aligned} \tag{10.14}$$

If the three values are identical, the image already satisfies the gray world assumption and no further adjustment is necessary. In general, they may not be. We then compute the gain for the red and blue channels as $\hat{\alpha}$ and $\hat{\beta}$, where

$$\hat{\alpha} = \frac{G_{\text{avg}}}{R_{\text{avg}}} \qquad \text{and} \qquad \hat{\beta} = \frac{G_{\text{avg}}}{B_{\text{avg}}}. \tag{10.15}$$

The corrected image is formed with $\hat{R}_{\text{sensor}}(x,y)$, $\hat{G}_{\text{sensor}}(x,y)$, and $\hat{B}_{\text{sensor}}(x,y)$, where

$$\begin{aligned} \hat{R}_{\text{sensor}}(x,y) &= \hat{\alpha} R_{\text{sensor}}(x,y) \\ \hat{G}_{\text{sensor}}(x,y) &= G_{\text{sensor}}(x,y). \\ \hat{B}_{\text{sensor}}(x,y) &= \hat{\beta} B_{\text{sensor}}(x,y) \end{aligned} \tag{10.16}$$

Often, this image has sufficient intensity range for all the channels. In the event that the highest intensity of the three channels is significantly below the maximum allowable value, we can scale all three channels by the same amount so that the average intensity is still preserved. This gray world method is quite effective in practice, except in situations where a certain color may dominate, such as a blue hue for the sky, or when an object with a substantial amount of a certain color occupies the majority of the view.

There are a number of extensions to this method that can deal with such situations. One example is given in Reference [16]. In this method, one defines a region in the $R_{\text{avg}} - G_{\text{avg}}$ versus $B_{\text{avg}} - G_{\text{avg}}$ plane. If the computed $\{R_{\text{avg}}, G_{\text{avg}}, B_{\text{avg}}\}$ falls within the region, the scene is considered good enough and AWB adjustments using Equation 10.16 will not be performed.

10.4.2 White Patch

The second method is based on the Retinex theory[2] of visual color constancy, which argues that perceived white is associated with the maximum cone signals [18]. This is also known as the white world assumption [19]. This is because the brightest point in an image is often due to reflectance of a glossy surface, which tends to reflect the actual color of the light source [20]. The white balancing scheme then attempts to equalize the maximum value of the three channels to produce a white patch. To avoid disturbances to the calculation caused by a few bright pixels, one can treat clusters of pixels or lowpass the image [4]. To implement this, we compute

$$\begin{aligned} R_{\max} &= \max_{x,y} R_{\text{sensor}}(x,y) \\ G_{\max} &= \max_{x,y} G_{\text{sensor}}(x,y). \\ B_{\max} &= \max_{x,y} B_{\text{sensor}}(x,y) \end{aligned} \tag{10.17}$$

If $G_{\max}$ is too small we can scale the green intensities up first, otherwise we keep the green channel unchanged. We define the gain for the red and blue channels as $\tilde{\alpha}$ and $\tilde{\beta}$, where

$$\tilde{\alpha} = \frac{G_{\max}}{R_{\max}} \qquad \text{and} \qquad \tilde{\beta} = \frac{G_{\max}}{B_{\max}}. \tag{10.18}$$

The corrected image is formed with $\tilde{R}_{\text{sensor}}(x,y)$, $\tilde{G}_{\text{sensor}}(x,y)$, and $\tilde{B}_{\text{sensor}}(x,y)$, where

$$\begin{aligned} \tilde{R}_{\text{sensor}}(x,y) &= \tilde{\alpha} R_{\text{sensor}}(x,y) \\ \tilde{G}_{\text{sensor}}(x,y) &= G_{\text{sensor}}(x,y). \\ \tilde{B}_{\text{sensor}}(x,y) &= \tilde{\beta} B_{\text{sensor}}(x,y) \end{aligned} \tag{10.19}$$

Gray world and white patch methods have their respective strengths. It is conceivable that satisfying the conditions in both methods would result in even better images. But we first need to make the following remarks:

[2]Retinex, which comes from the words *retina* and *cortex*, was coined to suggest that both the eye and the brain are involved in visual color constancy [17].

- For most images, the two methods produce different results. In other words, the corrected image can rarely satisfy both the gray world assumption and the Retinex theory.
- Equations 10.16 and 10.19 are both linear adjustments to the pixel intensities. Furthermore, there is also a fixed point in the mappings: for pixels with zero intensity, the two mappings would not affect their values. Evidently, it is rarely possible to achieve the requirements of both gray world assumption and Retinex theory with a linear technique.

Instead, a simple adjustment with a quadratic mapping of intensities was described in Reference [21]. Let the change to the red channel be

$$\breve{R}_{\text{sensor}}(x,y) = \mu R^2_{\text{sensor}}(x,y) + \nu R_{\text{sensor}}(x,y), \tag{10.20}$$

where μ and ν are parameters to be found. The adjustment to the blue channel is computed analogously. To satisfy the gray world assumption, we require that

$$\sum_{x=1}^{n}\sum_{y=1}^{n} \breve{R}_{\text{sensor}}(x,y) = n^2 G_{\text{avg}}, \tag{10.21}$$

and therefore,

$$\mu \sum_{x=1}^{n}\sum_{y=1}^{n} R^2_{\text{sensor}}(x,y) + \nu \sum_{x=1}^{n}\sum_{y=1}^{n} R_{\text{sensor}}(x,y) = n^2 G_{\text{avg}}. \tag{10.22}$$

Simultaneously, to satisfy the Retinex assumption to produce a white patch, we need

$$\max_{x,y} \breve{R}_{\text{sensor}}(x,y) = G_{\max}, \tag{10.23}$$

and therefore, if we assume that $R_{\text{sensor}}(x,y)$ takes on integer values between 0 and 255, and that μ and ν are positive numbers,

$$\mu \max_{x,y} R^2_{\text{sensor}}(x,y) + \nu \max_{x,y} R_{\text{sensor}}(x,y) = G_{\max}. \tag{10.24}$$

Equations 10.22 and 10.24 together form two equations in two unknowns. We can represent them in a matrix form

$$\begin{bmatrix} \sum_{x=1}^{n}\sum_{y=1}^{n} R^2_{\text{sensor}}(x,y) & \sum_{x=1}^{n}\sum_{y=1}^{n} R_{\text{sensor}}(x,y) \\ \max_{x,y} R^2_{\text{sensor}}(x,y) & \max_{x,y} R_{\text{sensor}}(x,y) \end{bmatrix} \begin{bmatrix} \mu \\ \nu \end{bmatrix} = \begin{bmatrix} n^2 G_{\text{avg}} \\ G_{\max} \end{bmatrix}. \tag{10.25}$$

This can be solved analytically for μ and ν using Cramer's rule.

10.4.3 Iterative White Balancing

The gray world method and white patch method described above are global techniques in that all pixels are involved in the computation. A drawback is that both may be susceptible to statistical anomalies. For the former, the method will give incorrect results if the scene

is heavily biased towards certain color cast, such as an outdoor scene of an ocean and the sky is typically rich in blue. For the white patch method, if a few pixels in the image have very large red, green, or blue values, they end up dominating the calculations.

In contrast, we have algorithms that pre-select a subset of pixels fulfilling certain *a priori* criteria, and the necessary color correction is derived from these pixels, although the adjustment is performed on all pixels subsequently. We can, for instance, perform an iterative white balancing technique as follows by extracting certain white points. We first convert the RGB values to YUV, a color space commonly used in video signals such as the PAL format, given by the following formula:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R_{\text{sensor}} \\ G_{\text{sensor}} \\ B_{\text{sensor}} \end{bmatrix}. \tag{10.26}$$

An ideal white point is when $R_{\text{sensor}} = G_{\text{sensor}} = B_{\text{sensor}} = 255$, which when put to the equation above makes $Y = 255$ and $U = V = 0$. Relaxing this condition a bit, we extract the pixels as white points if they satisfy the condition [22]

$$\begin{aligned} Y &> \xi \\ |U| &< \rho, \\ |V| &< \tau \end{aligned} \tag{10.27}$$

or an alternative criterion defined as [23]:

$$Y - |U| - |V| > \zeta, \tag{10.28}$$

where ξ, ρ, τ, and ζ are some pre-defined constants. While such a local method can avoid the scene being dominated by statistical anomalies, there are also situations that this would fail such as when there is no white object in the scene.

Another refinement is to look at gray points, which form a superset of the white points and therefore are more abundant in a typical scene. Reference [24] proposes selecting these points by the formula

$$\frac{|U| + |V|}{Y} < \eta, \tag{10.29}$$

where η is a positive threshold value much less than 1. The rationale is that if the light source is biased, say, to have a stronger red component, we can represent the captured red component $\tilde{R}$ as

$$\tilde{R} = (1 + \kappa_R)R, \tag{10.30}$$

where R is the true red component if captured in a canonical light source, and κ_R denotes the percentage increase. This gives rise to a set of Y, U, and V where

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} (1+\kappa_R)R \\ G \\ B \end{bmatrix}. \tag{10.31}$$

Note that for a canonical illumination, we have $R = G = B$ for a gray point, and therefore

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} (1+\kappa_R)R \\ R \\ R \end{bmatrix} \tag{10.32}$$

$$= \begin{bmatrix} 1+0.299\kappa_R \\ -0.147\kappa_R \\ 0.615\kappa_R \end{bmatrix} R. \tag{10.33}$$

Putting the above in Equation 10.29, we get

$$\frac{|U|+|V|}{Y} = \frac{0.147\kappa_R + 0.615\kappa_R}{1+0.299\kappa_R} \tag{10.34}$$

$$= \frac{0.762\kappa_R}{1+0.299\kappa_R} < \nu. \tag{10.35}$$

This value is close to zero if κ_R is small. Similar results can be derived if the color cast is in green or blue.

After selecting these gray points, we compute their average U and V values as $\hat{U}_{\text{avg}}$ and $\hat{V}_{\text{avg}}$. An iterative procedure is then employed to adjust them both to zero. At the jth iteration, we compute

$$\phi_j = \max(|\hat{U}_{\text{avg}}|, |\hat{V}_{\text{avg}}|). \tag{10.36}$$

If this equals to $\hat{U}_{\text{avg}}$, implying that the color is biased towards blue, we adjust the gain of the blue channel. Otherwise, the color is biased towards red, and the gain of the red channel is adjusted. The amount of adjustment used in Reference [24] is empirical and determined by trial and error. This changes $\hat{U}_{\text{avg}}$ and $\hat{V}_{\text{avg}}$ for the next iteration, and Equation 10.36 is computed again until satisfactory results are obtained.

10.4.4 Illuminant Voting

The three methods discussed above all make assumptions about the effects of illumination and adjust the pixel intensities directly. In principle, such methods attempt to adjust the intensity values of an image so that they appear "normal," but there is no guarantee that the resulting image is indeed possible under any illuminant! In other words, we may have created an image that is not physically realizable with any lighting condition on the particular object. On the other hand, there are also various techniques that aim at recovering the illuminant explicitly from the observed images. One such example is the illuminant voting technique [25]. After identifying the illuminant, the correction to any alternative lighting condition will ensure that the resulting image is realizable.

This illuminant voting method is based on the idea of Hough transform. This is a well known technique in image processing, especially in pattern detection, and can be illustrated as follows. Suppose we would like to detect a straight line in an image. This line can be represented as

$$\rho = x\cos\theta + y\sin\theta \tag{10.37}$$

in the x–y plane, where ρ is the distance from the origin and θ is the angle of the line. The Hough transform of this line is then the point (ρ, θ) in a new parameter space. We

can think of the Hough transform as mapping a line to a point, but we can also think of it as mapping a point to a line, because a point in the original x–y plane can relate to a set of (ρ_j, θ_j), where j is the index for the element, as long as they satisfy Equation 10.37 above. In theory there is an infinite number of (ρ_j, θ_j), but in practice they are quantized and therefore there is only a finite number of elements. Thus, we can imagine each point casts one vote to each member of the element. When we have multiple points, the (ρ_j, θ_j) with the most number of votes denotes the strongest presence of a line. In implementation, we commonly pick θ_j first and then solve for ρ_j and eventually count the votes, before moving on to a new value of θ_j. Details of the Hough transform can be found in many image processing textbooks (e.g., Reference [26]).

In a similar manner, we rely on the observed data to vote for the most likely illuminant. This requires modelling of the illuminant and reflectance by using low order linear combinations, as described in Equations 10.1 and 10.2. Putting them to Equation 10.12, we have

$$\begin{bmatrix} R_{\text{sensor}} \\ G_{\text{sensor}} \\ B_{\text{sensor}} \end{bmatrix} = \begin{bmatrix} \int_{400}^{700} r(\lambda) \sum_{j=1}^{m} \sum_{k=1}^{n} \alpha_j \beta_k I_j(\lambda) R_k(\lambda)\, \mathrm{d}\lambda \\ \int_{400}^{700} g(\lambda) \sum_{j=1}^{m} \sum_{k=1}^{n} \alpha_j \beta_k I_j(\lambda) R_k(\lambda)\, \mathrm{d}\lambda \\ \int_{400}^{700} b(\lambda) \sum_{j=1}^{m} \sum_{k=1}^{n} \alpha_j \beta_k I_j(\lambda) R_k(\lambda)\, \mathrm{d}\lambda \end{bmatrix} \tag{10.38}$$

$$= \left(\sum_{k=1}^{n} \beta_k M_k \right) \boldsymbol{\alpha}, \tag{10.39}$$

where the jth column of M_k, denoted as $(M_k)_j$, equals

$$(M_k)_j = \begin{bmatrix} \int_{400}^{700} r(\lambda) I_j(\lambda) R_k(\lambda)\, \mathrm{d}\lambda \\ \int_{400}^{700} g(\lambda) I_j(\lambda) R_k(\lambda)\, \mathrm{d}\lambda \\ \int_{400}^{700} b(\lambda) I_j(\lambda) R_k(\lambda)\, \mathrm{d}\lambda \end{bmatrix} \tag{10.40}$$

and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_m]^T$. Thus, it is clear that the equation above is linear in $\boldsymbol{\alpha}$. In fact, it is bilinear in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, where $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_n]^T$, because the above equation can also be written by interchanging illumination and spectral reflectance [27].

Given this bilinearity, we can use the observed pixel data to vote for the set of illuminant and reflectance parameters in a way similar to the Hough transform. The procedure consists of the following steps:

1. *Selection of the reflectance parameters* — We pick a set of β_j, which determine the object reflectance given the basis functions $R_j(\lambda)$ in Equation 10.2. Since there could be many possible object spectral reflectances in the scene, we have to go through the three-step procedure many times, each with a different set of β_j.

2. *Determination of illumination parameters* — We solve for α_j using Equation 10.39 above. Note that this is an inverse problem. Provided that $(\sum_{k=1}^{n} \beta_k M_k)$ is not singular, a solution can be found. However, it should also be noted that if the matrix is ill-conditioned [28], [29], the solution can be very sensitive to noise. To deal with this problem, typically we retain only the cases where the system matrix is well-conditioned.

3. *Casting of vote* — A vote is cast for the α obtained. As with most implementations of Hough transform, this is quantized so that similar values are grouped together. Otherwise, there will be too many singleton votes.

After repeating the procedure for different object reflectances, the one with the most votes is deemed the illuminant.

10.4.5 Color by Correlation

The fundamental premise of the color by correlation method is that although there are numerous possible spectral power distributions such as those in Figure 10.1, for instance, different hours of the day and different days would present different spectral power distributions of sunlight, there are only a small selection of substantially different illuminants (e.g., sunlight, fluorescent light, tungsten light, etc.). Some of these are modes or illumination conditions used in semi-automatic white balancing, where the user selects the particular mode and the camera performs white balancing accordingly. Similar to the previous method, the goal of AWB is achieved through illuminant identification, but the difference between the two is that the current method seeks not just a simple answer of the illumination function $I(\lambda)$, but a set of possible illuminants together with their likelihoods. Thus, not only does it determine the most likely illuminant, but it also computes the likelihood of all other illuminants so that the *error margin* of the subsequent choices is also known.

A prerequisite for this method is that we need to know the range and distribution of image colors that can be recorded by the camera under a set of possible lights [12]. We can then correlate the observed image with these distributions and identify the closest one as the most likely illuminant. More precisely, assume that there are k possible illuminants altogether. Instead of working with three sensory responses, we deal with only the chromaticity, where we can compute the chromaticities (c_1, c_2) as

$$c_1 = \frac{R_{\text{sensor}}}{G_{\text{sensor}}} \qquad \text{and} \qquad c_2 = \frac{B_{\text{sensor}}}{G_{\text{sensor}}}. \tag{10.41}$$

In practice, Reference [12] advocates the equation

$$c_1 = \left(\frac{R_{\text{sensor}}}{G_{\text{sensor}}}\right)^{\frac{1}{3}} \qquad \text{and} \qquad c_2 = \left(\frac{B_{\text{sensor}}}{G_{\text{sensor}}}\right)^{\frac{1}{3}} \tag{10.42}$$

which leads to chromaticities that are more uniformly distributed.

We partition the space of all chromaticities into $N \times N$ bins. The task is now to determine the possible (c_1, c_2) under each illuminant. There are a few possibilities:

1. The empirical way is to take the camera and capture a wide range of objects with various surface reflectances under each illuminant. We can then obtain the gamut of colors which the camera records under each lighting condition. This approach however can be rather cumbersome when there are many possible illuminants, and some may not be easily obtained at will (e.g., a bright sunlight illumination when it happens that the experimenters are experiencing rainy days!).

2. We can generate the chromaticities using Equations 10.12 and 10.42. This requires us to know the spectral response characteristics of the camera (such as in Figure 10.5b), the spectral power distribution of each illuminant, and the surface reflectance of a range of objects. In addition, we can take the convex hull of these chromaticities to form the gamut, setting all entries inside the gamut to be 1 and those outside to be 0.

3. We can further refine the scheme above by assigning the probability of each chromaticity value as the entries. This is computed empirically from the relative frequencies of occurrence estimated from the number of surfaces falling in each bin of the discretized chromaticity space.

We record the information above in an $N^2 \times k$ correlation matrix C, where each column (denoted as $(C)_j$ for the jth column) corresponds to a possible illuminant. Its row entry is the likelihood of observing that chromaticity under the particular illumination.

Now for a given image, we correlate the above information with that present in the image. We transform the pixel intensities to chromaticity values using the same formula as above, such as Equation 10.42. We then form a vector $\boldsymbol{\omega}$ of length N^2, where the jth element ω_j is one if the corresponding chromaticity value is present in the image, and zero otherwise. We can then compute the most appropriate illuminant $\hat{j}$ by the formula

$$\hat{j} = \arg\max_j \langle \boldsymbol{\omega}, (C)_j \rangle \tag{10.43}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. Another way to view this is that if we compute

$$\boldsymbol{\chi} = \boldsymbol{\omega}^T C \tag{10.44}$$

then the vector $\boldsymbol{\chi}$ is a row vector of length k, where each value suggests the likelihood of the illuminant. Thus, in a single operation we can find not only the most likely illuminant but also the error margin of the others.

In summary, the color by correlation method entails the following three-step process:

1. *Preprocessing step* — Information about the interaction between image colors and illuminants is coded. This is considered the prior information about the illuminants.

2. *Correlation step* — This prior information is correlated with the information that is present in a particular image. In other words, the colors in an image determine the likelihood of each possible illuminant.

3. *Recovery step* — These likelihoods are used to recover an estimate of the scene illuminant.

10.4.6 Other Methods

The above discussion of AWB techniques is by no means exhaustive. Other promising techniques include the gamut mapping algorithm using coefficient rule (CRULE) [30], color in perspective [31], Bayesian formulation [10], neural networks [32], adaptive gains [33], [34], and combined strategies [14]. We refer readers to these original papers for further descriptions of their methods.

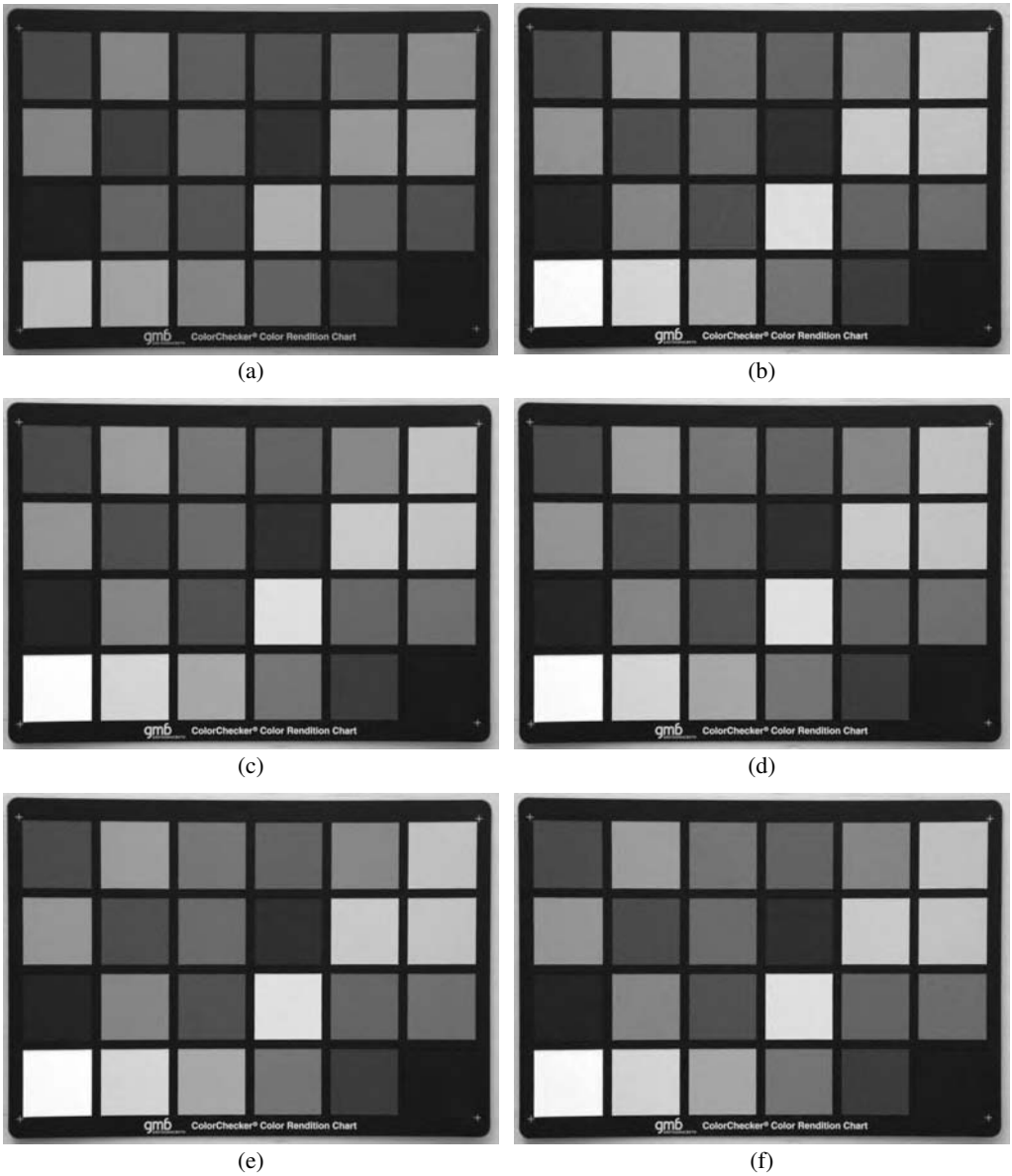


FIGURE 10.9 (See color insert.)

AWB methods for the Macbeth color chart: (a) original image, (b) gray world, (c) white patch, (d) iterative white balancing, (d) illuminant voting, and (f) color by correlation.

10.5 Implementations and Quality Evaluations

In this section, we consider the performance of the above methods using a few test images. Our aim is not to extensively compare the various methods, which can often be found in the respective original papers and others written specifically for such a purpose [35], [36], but to give readers some general ideas of the performance of these techniques. It is also

known to be difficult to objectively evaluate image quality. With synthetic data, we can generate an *ideal* image with the desirable illumination, and a *test* image captured with another illumination but corrected with one of the AWB algorithms. If the former has intensities $\{R_{\text{ideal}}, G_{\text{ideal}}, B_{\text{ideal}}\}$ in the three channels and the latter has intensities $\{R_{\text{test}}, G_{\text{test}}, B_{\text{test}}\}$, we can compute the mean square error (MSE) between these images using the formula

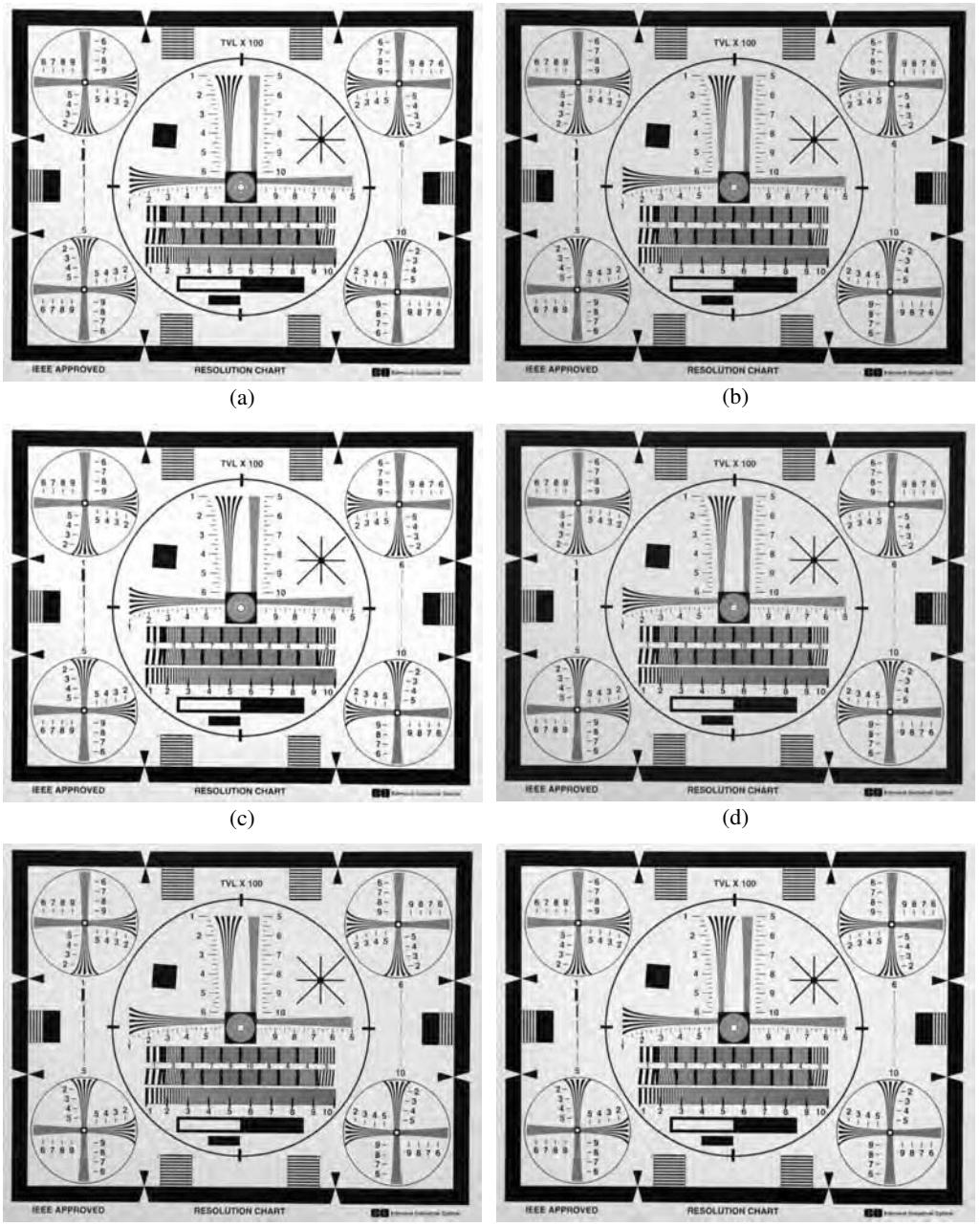
$$\text{MSE} = \sum_{x=1}^{n} \sum_{y=1}^{n} \left[(R_{\text{ideal}} - R_{\text{test}})^2 + (G_{\text{ideal}} - G_{\text{test}})^2 + (B_{\text{ideal}} - B_{\text{test}})^2 \right], \tag{10.45}$$

where each of the quantities above has the argument (x, y). A large MSE means that the ideal and test images are dissimilar, and suggests that the AWB algorithm may not be working well.

Unfortunately, MSE is commonly not a good metric for two reasons [37]. First, with real data we may not have the ideal image that we can compare with the test image. Even if we do, a second problem is that MSE does not correspond to the human perception of images. For instance, if we scale the intensities of the test image or shift it spatially by a small amount, the effects may be rather negligible perceptually, but mathematically the MSE can be significantly increased. There are other possibilities to compare the color differences objectively, such as using S-CIELAB [38] and CIEDE2000 [39]. Below, however, we mainly evaluate three sets of images subjectively to give readers a flavor of the AWB algorithms.

This first one is shown in Figure 10.9. In Figure 10.9a, the original image is seen to have a reddish-orange hue. The performance of the AWB algorithms is shown in Figure 10.9b to Figure 10.9f. In this particular case, the results of these algorithms are all quite satisfactory, and resemble each other. This can be attributed to the nature of this color chart object. For example, because many colors are present and there is no bias towards, for example, red, green, or blue, the assumption that the average intensity should be gray is quite agreeable. In a similar way, the object has a white patch that should correspond to the maximum intensity of the red, green, and blue channels. After the correction, the white patch is clearly evident.

Next, we consider the performance on another set of images shown in Figure 10.10. As in the previous example, Figure 10.10a is the original image. In this case, a bluish-green cast is visible in the image. Note that the actual image should be binary with black and white only. After the correction on the original image, the gray world technique shown in Figure 10.10b and the iterative white balancing scheme in Figure 10.10d both result in images that contain shades of gray only. However, the white background is rendered somewhat grayish in both cases. This can be attributed to the fact that both algorithms aim at reducing the chromatic components of the image, but do not have any mechanism that favors white to gray. On the other hand, the white patch technique in Figure 10.10c and the color by correlation method in Figure 10.10f are better in forcing the background to appear white. Note that in the white patch algorithm, we have filtered out the isolated bright pixels as mentioned in Section 10.4.2. Finally, the illuminant voting algorithm in Figure 10.10e seems to have over-compensated for the bluish cast and the resulting corrected image now contains mild shades of yellow.



(a) (b) (c) (d) (e) (f)

FIGURE 10.10 (See color insert.)

AWB methods for the resolution chart: (a) original image, (b) gray world, (c) white patch, (d) iterative white balancing, (d) illuminant voting, and (f) color by correlation.

The third set of experimental results is given in Figure 10.11. This is an image of a bookshelf taken under fluorescent light but with incorrect white balance setting in the camera. We can observe that the white patch method in Figure 10.11c seems to perform the best in this case, whereas the gray world method in Figure 10.11b and the iterative white

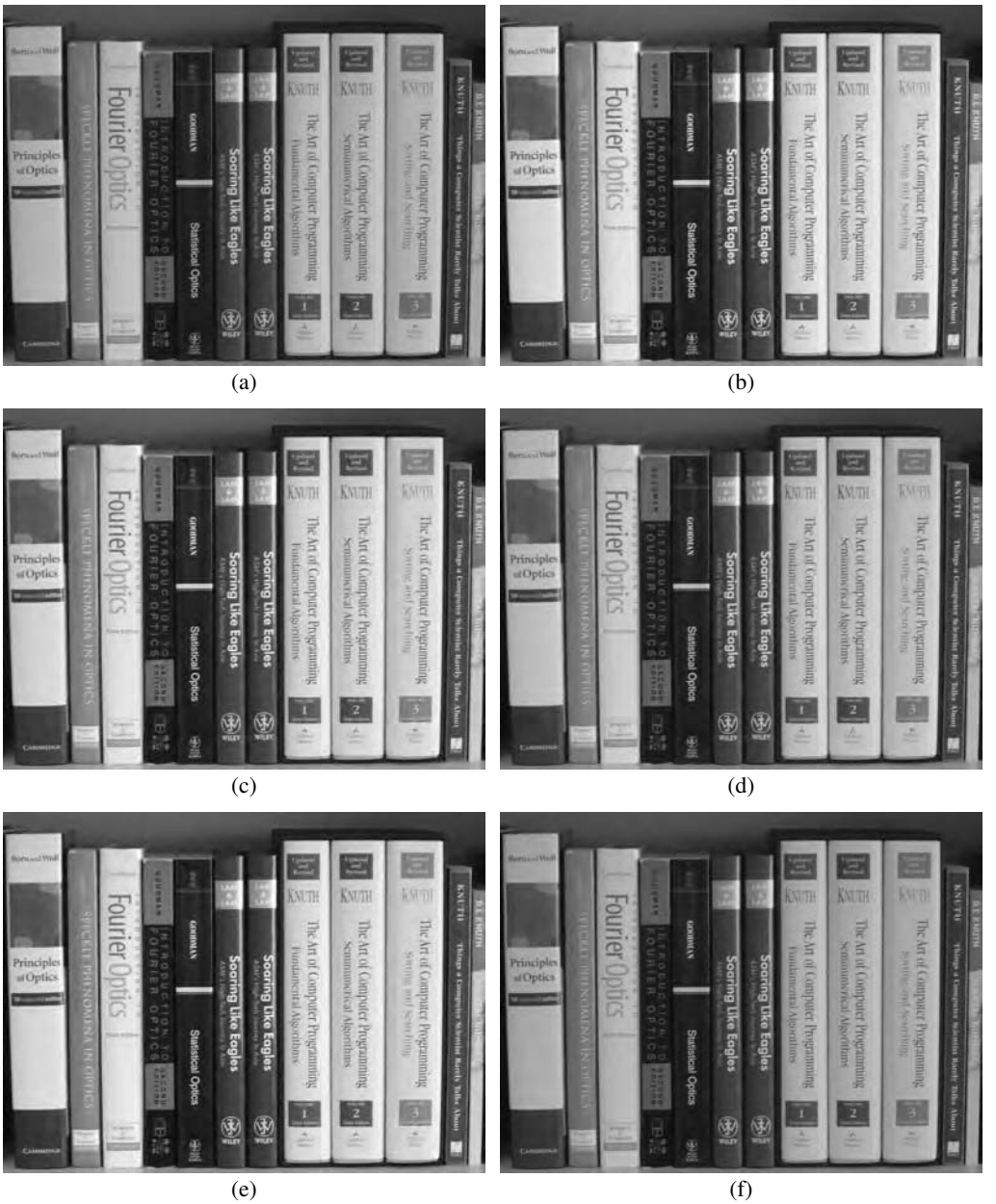


FIGURE 10.11 (See color insert.)

AWB methods for the *bookshelf* image: (a) original image, (b) gray world, (c) white patch, (d) iterative white balancing, (d) illuminant voting, and (f) color by correlation.

balancing scheme in Figure 10.11d again appear to produce an image that is slightly grayish. Both the illuminant voting and color by correlation methods, in Figure 10.11e and Figure 10.11f respectively, perform white balancing insufficiently. This result seems to agree with the comment made in Reference [36] that algorithms taking advantage of the chromaticity statistics seem to perform worse than expected.

10.6 Conclusion

In this chapter we considered the issue of automatic white balancing (AWB) in digital photography. We discussed the nature of the problem, and various algorithms that could be implemented to achieve AWB, including gray world, white patch, iterative white balancing, illuminant voting, and color by correlation. Nevertheless, we should note that color constancy — the root problem of AWB — is still recognized as a difficult problem that has not been solved satisfactorily, or even understood well enough how humans and some other animals possess this fascinating quality. Even the state-of-the-art algorithms are not nearly as good as our own color constancy [10] or in some cases, sufficient for machine vision tasks such as object recognition [40]. Evidently, there is much room for research both in understanding color constancy in human visual systems and the possibility of adapting it or using some other methods to achieve AWB in digital camera systems design.

Acknowledgments

This work is supported in part by Grant 10204548 from the University Research Committee of the University of Hong Kong and by the Research Grants Council of the Hong Kong Special Administrative Region, China under Project HKU 7143/05E. Dong Liang of the University of Hong Kong put much effort in the implementation of the various AWB algorithms. Experience gained when the principal author was involved with the Stanford programmable digital camera project also helped shape the content of this chapter.

References

[1] E. Land, "Recent advances in retinex theory and some implications for cortical computations: Color vision and the natural images," *Proceedings of the National Academy of Science*, vol. 80, no. 16, pp. 5163–5169, August 1983.

[2] E.Y. Lam, "Image restoration in digital photography," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 2, pp. 269–274, May 2003.

[3] R. Lukac, "Single-sensor imaging in consumer digital cameras: A survey of recent advances and future directions," *Journal of Real-Time Image Processing*, vol. 1, no. 1, pp. 45–52, October 2003.

[4] R. Hunt, *The Reproduction of Colour*. Chichester, West Sussex, UK: John Wiley & Sons, 2004.

[5] D.B. Judd, D.L. MacAdam, and G. Wyszecki, "Spectral distribution of typical daylight as a function of correlated color temperature," *Journal of the Optical Society of America*, vol. 54, no. 8, pp. 1031–1040, 1964.

[6] J. Cohen, "Dependency of the spectral reflectance curves of the Munsell color chips," *Psychoneurological Science*, vol. 1, no. 12, pp. 369–370, 1964.

[7] M.J. Vrhel, R. Gershon, and L.S. Iwan, "Measurement and analysis of object reflectance spectra," *Color Research and Application*, vol. 19, no. 1, pp. 4–9, February 1994.

[8] E. Giorgianni and T. Madden, *Digital Color Management*. Reading, MA: Addison Wesley, 1998.

[9] L. Arend and A. Reeves, "Simultaneous color constancy," *Journal of the Optical Society of America A*, vol. 3, no. 10, pp. 1743–1751, October 1986.

[10] D.H. Brainard, W.A. Brunt, and J.M. Speigle, "Color constancy in the nearly natural image. I. Asymmetric matches," *Journal of the Optical Society of America A*, vol. 14, no. 9, pp. 2091–2110, September 1997.

[11] D. Qian, J. Toker, and S. Bencuya, "An automatic light spectrum compensation method for CCD white balance measurement," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 2, pp. 216–220, May 1997.

[12] G.D. Finlayson, S.D. Hordley, and P.M. Hubel, "Color by correlation: A simple, unifying framework for color constancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1209–1221, November 2001.

[13] L.G. Shapiro and G.C. Stockman, *Computer Vision*. Upper Saddle River, NJ: Prentice Hall, 2001.

[14] S. Bianco, F. Gasparini, and R. Schettini, "Combining strategies for white balance," in *Proceedings of the SPIE, Digital Photography III*, San Jose, CA, USA, January 2007, vol. 6502, id. 65020D.

[15] M. Fairchild, *Color Appearance Models*. Chichester, West Sussex, UK: John Wiley & Sons, 2005.

[16] Y. Kim, H.S. Lee, and A.W. Morales, "A video camera system with enhanced zoom tracking and auto white balance," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 3, pp. 428–434, August 2002.

[17] E. Land, "The Retinex," *American Scientist*, vol. 52, no. 2, pp. 247–264, 1964.

[18] E. Land and J. McCann, "Lightness and Retinex theory," *Journal of the Optical Society of America*, vol. 61, no. 1, pp. 1–11, 1971.

[19] N. Kehtarnavaz, H. Oh, and Y. Yoo, "Development and real-time implementation of auto white balancing scoring algorithm," *Real-Time Imaging*, vol. 8, no. 5, pp. 379–386, October 2002.

[20] C. Weng, H. Chen, and C. Fuh, "A novel automatic white balance method for digital still cameras," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, vol. 4, pp. 3801–3804.

[21] E.Y. Lam, "Combining gray world and Retinex theory for automatic white balance in digital photography," in *Proceedings of the International Symposium on Consumer Electronics*, Macau, China, June 2005, pp. 134–139.

[22] N. Nakano, R. Nishimura, H. Sai, A. Nishizawa, and H. Komstsu, "Digital still camera system for megapixel CCD," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 460–466, August 1998.

[23] R.Z. Zhou, J. He, and Z.L. Hong, "Adaptive algorithm of auto white balance for digital camera," *Journal of Computer-Aided Design and Computer Graphics*, vol. 17, no. 3, pp. 529–533, March 2005.

[24] J. Huo, Y. Chang, J. Wang, and X. Wei, "Robust automatic white balance algorithm using gray color points in images," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 541–546, May 2006.

[25] G. Sapiro, "Color and illuminant voting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 11, pp. 1210–1215, November 1999.

[26] R. Gonzalez and R. Woods, *Digital Image Processing*. Upper Saddle River, New Jersey: Prentice Hall, 2002.

[27] G. Sapiro, "Bilinear voting," in *Proceedings of the Sixth International Conference on Computer Vision*, Bombay, India, November 1998, pp. 178–183.

[28] E.Y. Lam and J.W. Goodman, "Iterative statistical approach to blind image deconvolution," *Journal of the Optical Society of America A*, vol. 17, no. 7, pp. 1177–1184, July 2000.

[29] G. Golub and C. Van Loan, *Matrix Computations*. Baltimore, Maryland: Johns Hopkins University Press, 1996.

[30] D.A. Forsyth, "A novel algorithm for color constancy," *International Journal of Computer Vision*, vol. 5, no. 1, pp. 5–36, August 1990.

[31] G.D. Finlayson, "Color in perspective," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 1034–1038, October 1996.

[32] B.V. Funt, V. Cardei, and K. Barnard, "Learning color constancy," in *Proceedings of the Fourth Color Imaging Conference*, Scottsdale, AZ, USA, November 1996, pp. 58–60.

[33] R. Lukac, "Refined automatic white balancing," *Electronic Letters*, vol. 43, no. 8, pp. 445–446, April 2007.

[34] R. Lukac, "New framework for automatic white balancing of digital camera images," *Signal Processing*, vol. 88, no. 3, pp. 582–593, March 2008.

[35] K. Barnard, V. Cardei, and B. Funt, "A comparison of color constancy algorithms - Part I: Methodology and experiments with synthesized data," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 972–983, September 2002.

[36] K. Barnard, B. Funt, L. Martin, and A. Coath, "A comparison of color constancy algorithms - Part II: Experiments with image data," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 985–996, September 2002.

[37] E.Y. Lam, "Robust minimization of lighting variation for real-time defect detection," *Real-Time Imaging*, vol. 10, no. 6, pp. 365–370, December 2004.

[38] X. Zhang and B. Wandell, "Color image fidelity metrics evaluated using image distortion maps," *Signal Processing*, vol. 70, no. 3, pp. 201–214, November 1998.

[39] G. Johnson and M. Fairchild, "A top down description of S-CIELAB and CIEDE2000," *Color Research and Application*, vol. 28, no. 6, pp. 425–435, December 2003.

[40] B. Funt, K. Barnard, and L. Martin, "Is machine colour constancy good enough?," in *Proceedings of the Fifth European Conference on Computer Vision*, Freiburg, Germany, June 1998, pp. 455–459.

11

Enhancement of Digital Photographs Using Color Transfer Techniques

François Pitié, Anil Kokaram, and Rozenn Dahyot

11.1 Introduction

Color is an essential aspect of a picture which conveys to the viewer many emotions and symbolic meanings. Adjusting the color grade of pictures is therefore an important step in professional photography. This process is part of the larger activity of grading in which the color and grain aspects of the photographic material are digitally manipulated. The term color grading will be used specifically to refer to the matching of color. Color grading is a delicate task since the slightest of color variations can alter the mood of a picture. For instance, changing the white point of a picture can make a picture look warmer or more

(a) (b) (c)

FIGURE 11.1 (See color insert.)

Color transfer example: (a) original image, (b) image with the target palette, and (c) output image. A color mapping is applied on the original picture to match the palette of an example provided by the user.

metallic. Refer to Chapter 10 for details on white balancing. Increasing the contrast can give a sharper aspect to the picture. Color grading is an even more critical step in movie postproduction. Like with photographic stills, it gives the movie a unique artistic signature. For instance the movie *Amélie* (2002) has a look of its own. Bright colors, deep black levels and saturated colors all contribute to create an artificial expressionistic world. The main problem of grading a movie is to adjust the color consistently across all the shots, even though the movie has been edited with heterogeneous video material. Shots taken at different times under natural light can have a substantially different *feel* due to even slight changes in lighting. Currently in the industry, color balancing (as it is called) is achieved by experienced artists who use very expensive edit hardware and software to manually match the color between frames by tuning parameters. Typical tuning operations comprise adjusting the exposure, brightness and contrast, calibrating the white point or finding a color mapping curve for the luminance levels and the three color channels. For instance, in an effort to balance the contrast of the red color, the digital samples in the red channel in one frame may be multiplied by some gain factor and the output image viewed and compared to the color of some other (a target) frame. The gain is then adjusted if the match in color is not quite right. The amount of adjustment and whether it is an increase or decrease depends crucially on the experience of the artist. This is a delicate task since the change in lighting conditions induces a very complex change of illumination. It would be therefore beneficial to automate this task in some way.

The scope of color grading goes beyond the sole photographic and postproduction activities. In digital restoration [1], the goal is to recover the original colors of paintings that have been faded by smoke or dust. The process can also be used for color image equalization for scientific data visualization [2], [3]. Also, the specialized activities of high dynamic range tone mapping [4], [5], as well as the grayscale to color [6] and color to grayscale processes [7], [8], could be considered as special instances of color grading.

11.1.1 How Does Color Grading Work?

The color grading workflow typically begins with grading the tone of the entire picture and then proceeds to local color correction where specific areas of the image are isolated for dedicated color grading. When dealing with image sequences, a tracking operation is

then necessary to keep local color corrections accurate across the whole shot. This chapter presents techniques which aim at facilitating the choice of the color mappings. These techniques belong to the class of example-based *color transfer* methods. The idea, first formulated by Reinhard et al. [9], has raised a lot of interest recently [10], [11], [12], [13]. Figure 11.1 illustrates this with an example. The original picture (Figure 11.1a) is transformed so that its colors match the palette of the target image (Figure 11.1b), regardless of the content of the pictures. Consider the two pictures as two sets of three-dimensional (3D) color pixels. A way of treating the recoloring problem would be to find a one-to-one color mapping that is applied to every pixel in the original image. For example in Figure 11.1, every blue pixel is recolored in green. The new picture is identical in every aspect to the original picture, except that the picture now exhibits the same color statistics, or palette, as the target picture.

Other color grading workflows could also be considered. Another way of treating the problem would be to recolor the picture in compliance with the actual 3D illumination structure of the scene. This kind of idea has been explored by Shen and Xin [14] with the aim of producing realistic recoloring under different illuminant. The method, derived from a dichromatic reflection model for the 3D objects, is however limited in practice by the difficulty of retrieving the necessary underlying 3D information from real photographic material. Another approach to color grading is to combine the color transfer mapping with the texture information. In particular it is worth mentioning the colorization of grayscale images by Welsh et al. [6] which draws from the recent success of non-parametric texture synthesis and transfer by Efros [15], [16] and Hertzmann [17]. The proposal is to replace the color of each pixel in the original image with the color of the best matching pixel in the example image. The best matching pixel is found by looking at the closest luminance image patch in the example image. The idea is that the color mapping will be more accurate if it is guided by the content information given by the neighboring texture. Bae et al. [18] have proposed a decomposition of the picture into a textured and a textureless layer followed by an application of the color transfer separately for the textured layer of the image and the textureless layer. This chapter will however not consider the texture information directly. Instead it will focus on the sole problem of finding a color mapping that transfers the color statistics of a picture example back to the original picture. Note that Welsh's and Bae's techniques still manipulate this idea of transfer of statistics, except that in their case the transfer is conditioned to the texture information.

The notion of transfer of statistics encompasses an entire range of possibilities from the simple match of the mean, variances [9], covariance [10], [19] to the exact transfer of the exact distribution of the color samples [11], [13], [20]. Thus, depending on how close the graded picture should match the color distribution of the example image, multiple techniques could be used. As will be explained, finding a color mapping is actually closely related to the *mass transportation* problem, which has a well established mathematical background [21], [22], [23]. This chapter aims then at conducting a comprehensive review of existing color statistic transfer techniques under this new perspective. The review, presented in Section 11.2, references existing work but also discloses new techniques that could be advantageously used.

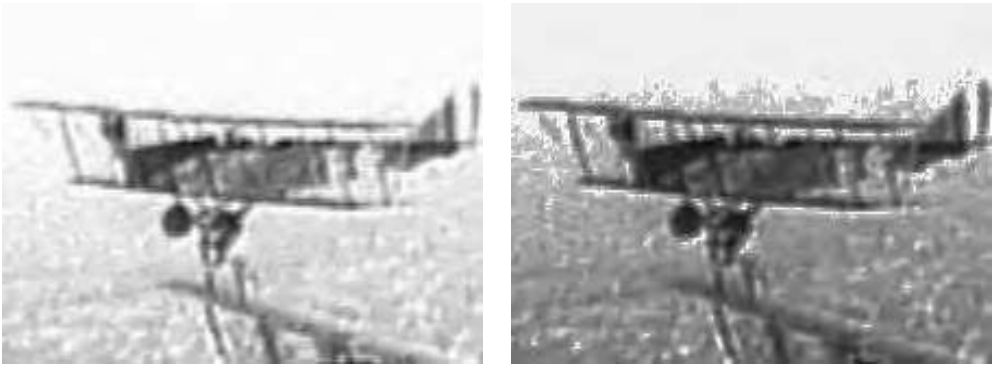


FIGURE 11.2

Example of grain induced by color transfer: (left) original image, and (right) image after mapping.

11.1.2 How to Deal with Content Variations

One important aspect of the color transfer problem is the change of content between the two pictures. Consider a pair of landscape images where the sky in one picture covers a larger area than in the other. When transferring the color from one picture to the other, the excess of sky color may be used in parts of the scenery of the ground in the other. This is because all color transfer algorithms are sensitive to variations in the areas of the image occupied by the same color. They risk overstretching the color mappings and thus producing unbelievable renderings as visible on Figure 11.2 which are very grainy. To deal with this issue a simple solution [9], [10], [14] is to manually select swatches in both pictures and thus associate color clusters corresponding to the same content. This is tantamount to performing manual image segmentation, and is simply impractical for a large variety of images, and certainly for sequences of images. The solution commonly adopted [11], [13] for color transfer techniques is to simply reduce the accuracy of both distributions by smoothing the color histograms. This simple technique avoids artifacts at the expense of an accurate transfer.

Methods that make use of the spatial information to constrain the color mapping [6], [7], [24] can be successful but are usually computationally demanding. Another solution is to restrict the variability on the color mapping. For example, Chang [25] proposed classification of pixels of both images in a restricted set of basic color categories, derived from psycho-physiological studies (red, blue, pink, etc.). The color transfer ensures, for instance, that blueish pixels remain blueish pixels. This gives a more natural transformation. The disadvantage is that it limits the range of possible color transfers.

The solution adopted here is to treat the grainy artifacts after the color grading. The noise could be attenuated by employing various color filtering techniques [26], [27] but these may not be the best solution for the the intended application due to the application specifics. Instead, a dedicated postprocessing is proposed which aims at protecting the original picture by preserving its original gradient field while preserving the color transfer. This balance is done using a variational approach inspired by Poisson editing techniques [28]. Protecting the gradient of the original picture particularly protects the flat areas and more generally results in the exact reproduction of film grain/noise as in the original image.

This chapter is organized as follows. A review of techniques for transferring color statistics is presented in Section 11.2. The review is accompanied with a table of comparative results. The techniques suitable for finding linear mappings are presented in Section 11.3 and nonlinear mappings in Section 11.4. Section 11.5 deals with the impact of the color space. The problem of dealing with content variations and the steps of the regraining stage are then explained in Section 11.6. The chapter is concluded in Section 11.7 with some examples coming from applications where color transfer is applied.

11.2 Color Distribution Transfer

The notion of color statistics can be understood by considering that an image can be represented as a set of color samples. When working in RGB color space, the image is represented by the set of the RGB color samples $(R^{(i)}, G^{(i)}, B^{(i)})_{1 \leq i \leq M}$. In a probabilistic sense, these color samples are realizations of a 3D color random variable which will be denoted as u for the original image and v for the target palette image. The color palettes of the original and target pictures correspond then to the distributions of u and v. To simplify the presentation of the problem, it supposed here that both distributions have absolutely continuous probability density function (pdf) f and g. In practice, pdfs can only be numerically approximated. The simplest form of approximation would be color histograms. Since only a finite number of color samples are available, color histograms are usually very sparse and rough. The pdfs estimation can be improved by smoothing the histograms or even better by using standard kernel density approaches. The reader is invited to read Silverman [29] for more details on density estimation. One general idea of density estimation is that the amount of smoothing controls the degree of accuracy of the pdfs. At the extreme, the pdfs can be approximated up to simple multivariate Gaussian (MVG) distributions by estimating only the mean and the correlation matrices of u and v. Now that the notion of a color statistic has been properly introduced, the color transfer problem can then be defined as follows.

The problem of color transfer is to find a C^1 continuous mapping $u \rightarrow t(u)$, such that the new color distribution of $t(u)$ matches the target distribution g. This latter problem, illustrated in Figure 11.3, is also known as the *mass preserving transport problem* in the mathematic literature [21], [22], [23].

To characterize the mapping, it is worth noticing that the mapping is, in essence, a change of variables. Thus the transfer equation can be written as:

$$f(u)du = g(v)dv \quad \Rightarrow \quad f(u) = g(t(u))|\det J_t(u)| \tag{11.1}$$

where $J_t(u)$ is the Jacobian of t taken at u. Using the cumulative distribution functions F and G for f and g, the condition for the mapping to realize the transfer is derived as follows:

$$\forall u \in \mathbb{R}^N, F(u) = G(t(u)) \tag{11.2}$$

It is essential to realize here that the color mapping corresponds to a warping of the cumulative distribution function and not of the pdf. In grayscale images [30] where $N = 1$,

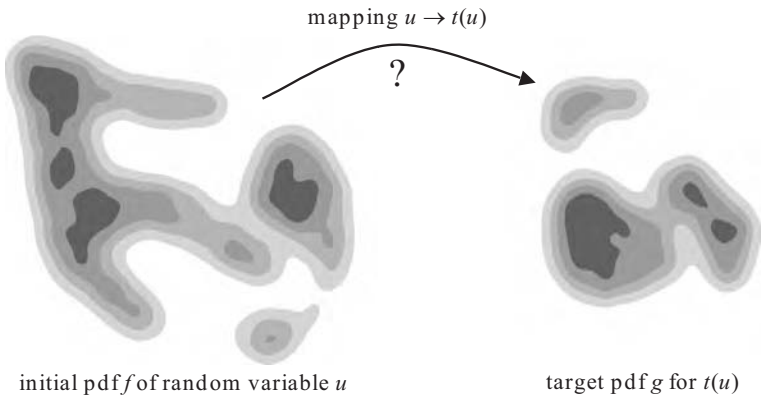


FIGURE 11.3

Distribution transfer concept. How to find a mapping that transforms the distribution on the left to the distribution on the right?

it is possible to invert the cumulative distribution function G, and the mapping has this simple form:

$$\forall u \in \mathbb{R}\ , t(u) = G^{-1}(F(u)) \tag{11.3}$$

where $G^{-1}(\alpha) = \inf\{u | G(u) \geq \alpha\}$. The mapping can then easily be solved by using discrete look-up tables. For higher dimensions, that is, with color images, the cumulative distribution function cannot be inverted since multiple solutions are possible. Thus multiple ways of finding a valid mapping can be achieved. Actually numerous different methods have been proposed and successfully applied to color transfer. These will be presented hereafter in Sections 11.3 and 11.4. The problem encountered with most of them is that the geometry of the resulting mapping might not be as it was intended and it is possible that an exact transfer maps black pixels to white and white pixels to black. The resulting picture will have the color proportions as expected, but locally the colors will have been swapped. To avoid this effect, a good solution is to further constrain the transfer problem and to ask the mapping to also minimize its displacement cost:

$$I[t] = \int_u \|t(u) - u\|^2 f(u) du \tag{11.4}$$

Finding this minimal mapping is known as the *Monge's optimal transportation problem*. Monge's problem has raised a major interest in mathematics in recent years [21], [22], [23] as it has been found to be relevant for many scientific fields like fluid mechanics. Another formulation of this problem is the *Kantorovitch*'s optimal transportation problem which offers a relaxation of the one-to-one mapping constraint by allowing one color to be mapped into multiple colors:

$$I[t] = \int_{u,v} \pi(u,v) \|v - u\|^2 du dv \tag{11.5}$$

The associations are described in $\pi(\cdot,\cdot)$ which is the joint pdf between u and v. The Kantorovitch solution is related to the Monge problem since for continuous distributions, it

turns out that the Kantorovitch solution coincides with the Monge solution, i.e., that for continuous pdfs, the best association is a one-to-one mapping: $\pi(u,v) = \delta(t(u) - v)$. In the rest of the chapter, the problem will be thus referred to as the Monge-Kantorovitch (MK) problem.

This chapter is not intended to be a course on the Monge-Kantorovitch problem and the reader eager to develop a better mathematical insight is invited to read a more specialized mathematical bibliography [21], [22], [23]. To understand the interest of the MK solution in color grading, three aspects of the MK solution will however be reported here. The first result of importance is that the MK solution always exists for continuous pdfs and is *unique*. This means that there will no room left for ambiguity. The second property is that the MK solution is consistent with orthogonal basis changes (the MK does not depend more on one component than the other). The last result, which is of interest here, is that the MK solution is the gradient of a convex function[1]:

$$t = \nabla\phi \quad \text{where} \quad \phi : \mathbb{R}^N \to \mathbb{R} \quad \text{is convex} \tag{11.6}$$

This might seem quite obscure at first sight, but this property is the equivalent of monotonicity for one-dimensional (1D) functions in $\mathbb{R}$. This property is thus quite intuitive for the color transfer problem. For instance, the brightest areas of a picture will still remain the brightest areas after mapping.

The following presents techniques that have been explored in the color grading literature, starting with the methods that consider only a linear transformation of the color samples and then presenting the solutions that can match any color distribution. Most of these techniques do not solve for the Monge-Kantorovitch problem. In particular, existing literature in the domain does not propose the MK solution for the linear case but it has been introduced here and turns out to outperform other linear methods. The different techniques are compared to each other so the reader will be able to make an independent judgment.

11.3 Linear Color Distribution Transfer Techniques

The linear case considers the problem of finding linear mappings of the form $t(u) = Tu + t_0$ where T is a $N \times N$ matrix. It is not necessarily possible to find a linear mapping in the general case, but this can always be achieved when both the original distributions f and the target distributions g are multivariate Gaussian distributions (MVG) $\mathcal{N}(\mu_u, \Sigma_u)$ and $\mathcal{N}(\mu_v, \Sigma_v)$

$$\begin{aligned} f(u) &= \frac{1}{(2\pi)^{N/2}|\Sigma_u|^{1/2}} \exp\left(-\frac{1}{2}(u-\mu_u)^T\Sigma_u^{-1}(u-\mu_u)\right) \\ g(v) &= \frac{1}{(2\pi)^{N/2}|\Sigma_v|^{1/2}} \exp\left(-\frac{1}{2}(v-\mu_v)^T\Sigma_v^{-1}(v-\mu_v)\right) \end{aligned} \tag{11.7}$$

[1] A convex function [22] $\phi : \mathbb{R}^N \to \mathbb{R}$ is such that $\forall u_1, u_2 \in \mathbb{R}^N, \alpha \in [0;1],\ \phi(\alpha u_1 + (1-\alpha)u_2) \leq \alpha\phi(u_1) + (1-\alpha)\phi(u_2)$.

with Σ_u and Σ_v the covariance matrices of u and v. Note that when the distributions are not MVG, a MVG approximation can always be obtained by estimating the mean and the covariance matrices of the distributions. To have the pdf transfer condition $g(t(u)) \propto f(u)$, it must hold that $(t(u)-\mu_v)^T\Sigma_v^{-1}(t(u)-\mu_v) = (u-\mu_u)^T\Sigma_f^{-1}(u-\mu_u)$. Thus t must satisfy the following condition:

$$t(u) = T(u-\mu_u)+\mu_v \quad \text{with} \quad T^T\Sigma_v^{-1}T = \Sigma_u^{-1} \tag{11.8}$$

It turns out that there are numerous solutions for the matrix T and thus multiple ways of transferring the color statistics.

11.3.1 Independent Transfer

The first method, used by Reinhard et al. [31] in their original paper on color transfer, is to simply match the means and the variances of each component independently. This means that both distributions are separable, thus the covariance matrices are diagonal and $\Sigma_u = \text{diag}(\text{var}(u_1),\ldots,\text{var}(u_N))$ and $\Sigma_v = \text{diag}(\text{var}(v_1),\ldots,\text{var}(v_N))$. It yields for the mapping that

$$T = \begin{bmatrix} \sqrt{\frac{\text{var}(v_1)}{\text{var}(u_1)}} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\frac{\text{var}(v_N)}{\text{var}(u_N)}} \end{bmatrix} \tag{11.9}$$

The independence assumption is simplistic since it is rarely true for real images. The poor quality of the transfer in the results in Figure 11.4c shows that this is indeed not always the case. The solution proposed by Reinhard is to work in the decorrelated color space $l\alpha\beta$ of Ruderman [32]. This helps to some extent but cannot guarantee a full decorrelation between components.

11.3.2 Cholesky Decomposition

Since both matrices Σ_u and Σ_v are symmetric semi definite positive, a solution is to use the Cholesky decomposition of $\Sigma_u = L_u L_u^T$ and $\Sigma_v = L_v L_v^T$ where L_u and L_v are lower triangular matrices with strictly positive diagonal elements. This decomposition yields the following solution:

$$T = L_v L_u^{-1} \tag{11.10}$$

Note that this solution is dependent on the ordering of the color components. Figure 11.4d shows however some improvements on the previous method. This method is quite successful in practice but is not as reliable as the MK solution.

11.3.3 Principal Axes Transfer

Another popular solution [10], [19], [33], [34], [35] is to find the mapping that realigns the principal axes of Σ_v to that of Σ_u. This can be done by using the square root operator for symmetric positive semidefinite matrices. The square root matrices $\Sigma_u^{1/2}$ and $\Sigma_v^{1/2}$ are

FIGURE 11.4 (See color insert.)

Results for linear techniques: (a) original image, (b) target palette, (c) separable linear transfer, (d) Cholesky based transfer, (e) principal axes transfer, and (f) linear Monge-Kantorovitch transfer. All transfers are done in the RGB color space.

obtained through the spectral decomposition of Σ_u and Σ_v:

$$\Sigma_u = P_u D_u P_u^T \text{ and } \Sigma_u^{1/2} = P_u D_u^{1/2} P_u^T \tag{11.11}$$

$$\Sigma_v = P_v D_v P_v^T \text{ and } \Sigma_v^{1/2} = P_v D_v^{1/2} P_v^T \tag{11.12}$$

where P_u and P_v are orthogonal matrices and D_u and D_v the diagonal matrix containing the (positive) eigenvalues of Σ_u and Σ_v. Note that the square roots $\Sigma_u^{1/2}$ and $\Sigma_v^{1/2}$ are uniquely defined. These decompositions lead to the following mapping:

$$T = \Sigma_v^{1/2} \Sigma_u^{-1/2} \tag{11.13}$$

Results displayed in Figure 11.4e show an improvement over the mapping based on the Cholesky decomposition. Note in particular that the violet color of the grass in Figure 11.4e is not present here. This might be due to fact that the mapping does not depend on the ordering of the color components.

11.3.4 Linear Monge-Kantorovitch Solution

A better approach would be to use the Monge-Kantorovitch solution for MVG distributions. The MK solution is geometrically more intuitive than the Cholesky or the Principal Axes solution. One could be concerned that the MK solution might not be linear, but fortunately it is actually linear and it admits a simple closed form. The detailed proof of how to find the MK mapping can be found in Reference [36]. The mainstay of the reasoning is that since the MK solution is the gradient of a convex function, the matrix T has to be

symmetric definite positive, which leads to this unique solution for T:

$$T = \Sigma_u^{-1/2} \left(\Sigma_u^{1/2} \Sigma_v \Sigma_u^{1/2} \right)^{-1} \Sigma_u^{-1/2} \tag{11.14}$$

The corresponding results in Figure 11.4f are convincing. The results are slightly better than the ones of the Principal Axes method. For instance a pink trace on the grass in front of the house is visible in Figure 11.4e but not in Figure 11.4f. The MK solution is thus interesting as it provides an intuitive and probably better mapping for a similar computational complexity to the popular methods.

11.4 Nonlinear Color Distribution Transfer Techniques

11.4.1 Independent Transfer

Extending the solutions for MVG to any distribution is sadly more complicated. The approach used at the moment in professional color grading tools is to extend the pdf transfer to higher dimensions by performing the 1D nonlinear pdf matching separately for each channel [37]. Like in the linear case, this is only exact if both distributions are separable, i.e., if the joint distribution is the product of its marginals:

$$f(u_1, u_2, \cdots, u_N) = f(u_1) f(u_2) \cdots f(u_N) \tag{11.15}$$

This is however not realistic in most situations. The result displayed in Figure 11.5c reflects this.

11.4.2 Composition Transfer

This method, that has been applied to the color transfer problem by Neumann [11], builds on the fact that the correlated variables $u_1, \cdots, u_N$ can be recombined into the following independent variables $u'_1, u'_2, \cdots, u'_N$:

$$u'_1 \sim f(u_1) \quad u'_2 \sim f(u_2|u_1) \quad \cdots \quad u'_N \sim f(u_N|u_1, \cdots, u_{N-1}) \tag{11.16}$$

The independence becomes apparent from the following conditional decomposition:

$$f(u_1, u_2, \cdots, u_N) = f(u_1) f(u_2|u_1) \cdots f(u_n|u_1, \cdots, u_{n-1}) \tag{11.17}$$

Since these conditional variables are independent, it is possible to use 1D pdf transfers separately for each of the conditional variables. The final mapping is then composed as follows:

$$t(u_1, \cdots, u_N) = (t_1(u_1), t_2(u_2|u_1), \cdots, t_N(u_N|u_1 \cdots, u_{N-1})) \tag{11.18}$$

and each 1D mapping $t_1, \cdots, t_N$ is found by using the corresponding pdf transfer:

$$\begin{aligned} t_1(u_1) &: f(u_1) \Rightarrow g(v_1) \\ t_2(u_2|u_1) &: f(u_2|u_1) \Rightarrow g(v_2|v_1) \\ &\vdots \\ t_N(u_N|u_1, \cdots, u_{N-1}) &: f(u_N|u_1, \cdots, u_{N-1}) \Rightarrow g(v_N|v_1, \cdots, v_{N-1}) \end{aligned} \tag{11.19}$$

FIGURE 11.5 (See color insert.)

Results for different scenarios: (a) original image, (b) target palette, (c) separable transfer, (d) composition transfer, (e) discrete Kantorovitch, and (f) IDT.

This technique suffers from two main problems. Firstly the mapping itself depends heavily on the order in which the variables are conditioned to each other. For instance matching $f(u_1)$ and then $f(u_2|u_1)$ results in a different mapping than mapping $f(u_2)$ and then $f(u_1|u_2)$. The second issue is that even for large pictures, the estimation of the conditional marginals like $f(u_N|u_1,\cdots,u_{N-1})$ is based on only a very small number of color samples since only a few pixels will have exactly the same color. This means in practice for a RGB color grading, that if the transfer of the first red component is accurate, the transfer of last blue component is however poor. This is reflected in Figure 11.5d where the blue gain has been overestimated. The situation can be improved by using some proper density estimation scheme [29] which mainly implies smoothing the original 3D histogram. Note that smoothing the pdfs is a delicate operation that requires some computational time for large color histograms.

11.4.3 The Discrete Kantorovitch Solution

As with the linear case, the Monge-Kantorovitch solution seems then to be more appropriate here. Numerical solutions exist for N-dimensional variables, but they involve heavy computational loads as they require the use of an iterative partial differential equation solver in the N-dimensional distribution [21]. However, to reduce the computational complexity, it is possible to segment the actual pdf to a smaller number of colors. Since the pdfs are now discretized, it might not be possible to find a one-to-one mapping that transfers one pdf exactly to another. Instead, the one-to-one mapping assumption is relaxed and it is allowed for one histogram bin to be mapped onto multiple bins. The problem is then to find the flow $\pi()$ (joint distribution) that minimizes the overall transportation cost:

$$\hat{\pi} = \inf_{\pi} \sum_{i,j} \pi(u_i, v_j)|u_i - v_j|^2 \tag{11.20}$$

with $\forall j,\ \sum_i \pi(u_i, v_j) = g(u_j) \quad \forall i,\ \sum_j \pi(u_i, v_j) = f(u_i)$. Note that this formulation of the transportation problem has been introduced in the computer vision community by Rubner under the name of Earth Mover Distance [38]. The discretized problem can be numerically solved by linear programming using the Simplex algorithm [39]. Several specialized algorithms for solving the transport problem also exist, notably the *northwest corner* method and the *Vogels approximation* [40]. Since a color can be mapped into multiple colors, the issue is now of deciding which color to assign to a particular pixel. Morovic proposes to decide for each pixel on a random basis. This process is, in essence, similar to a randomized dithering.

One remarkable result of the MK theory is that, in the continuous case, the Kantorovich solution is actually Monge's one-to-one mapping. This simply means that increasing the number of bins will reduce the dithering and gives a better approximation of Monge's color mapping. The problem is that the Simplex algorithm is quite slow and becomes intractable when dealing with large histograms. The result in Figure 11.5e has been obtained for a color histogram of 300 color bins. Despite these limitations, the method still produces visibly better results than the previously presented methods.

11.4.4 Transfer via the Radon Transform

Recently, Pitié et al. [41] have proposed another solution to the distribution transfer problem which is based on the iterative use of 1D transfers for various directions in the N-dimensional space. The idea is to break down the problem into a succession of 1D distribution transfer problems. Consider the use of the N-dimensional Radon Transform. It is widely acknowledged that via the Radon Transform, any N-dimensional function can be uniquely described as a series of projections onto 1D axes [42] (see Figure 11.6). In this case, the function considered is a N-dimensional pdf, hence the Radon Transform projections result in a series of 1D marginal pdfs from which can be derived the corresponding 1D pdf transfers along these axes. Intuitively then, operations on the N-dimensional pdf should be possible through applying the 1D pdf transfer along these axes. Consider that after some sequence of such manipulations, all 1D marginals match the corresponding marginals of the target distribution. It then follows that, by nature of the Radon Transform, the transformed f, corresponding to the transformed 1D marginals, now matches g.

The operation applied to the projected marginal distributions is thus similar to that used in 1-dimension. Denote a particular axis by its vector direction $e \in \mathbb{R}^N$. The projection of both pdfs f and g onto the axis e results in two 1D marginal pdfs f_e and g_e. Using the 1-dimensional pdf transfer mapping of Equation 11.3 yields a 1D mapping t_e along this axis:

$$\forall u \in \mathbb{R},\ t_e(u) = G_e^{-1}(F_e(u)) \tag{11.21}$$

For a N-dimensional sample $u = [u_1, \cdots, u_N]^{\mathrm{T}}$, the projection of the sample on the axis is given by the scalar product $e^{\mathrm{T}}u = \sum_i e_i u_i$, and the corresponding displacement along the axis is

$$u \rightarrow u + \delta \quad \text{with} \quad \delta = (t_e(e^{\mathrm{T}}u) - e^{\mathrm{T}}u)\, e \tag{11.22}$$

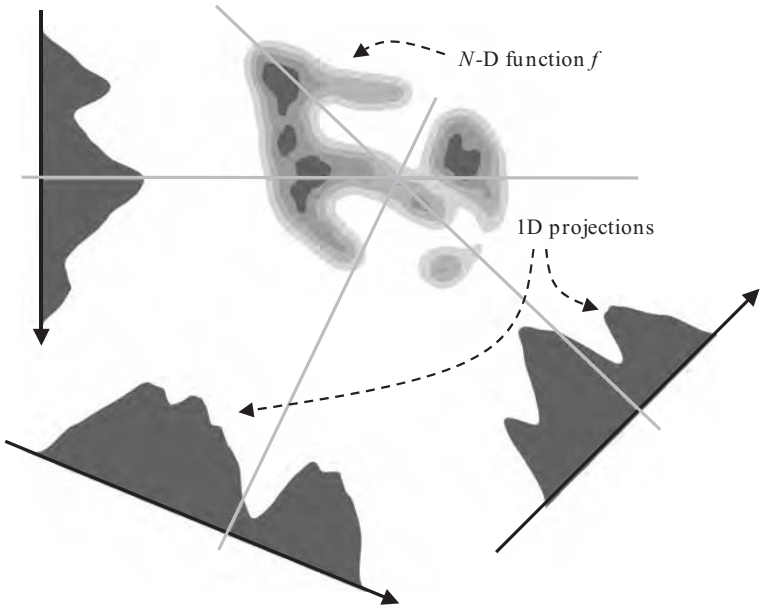


FIGURE 11.6

N-dimensional Radon transform of a distribution's pdf. The pdf is projected onto every possible axis. Each projection corresponds to a 1D marginal of the distribution. The ensemble of all projections uniquely describes the distribution.

ALGORITHM 11.1 Iterative distribution transfer using the Radon transform.

1. Initialization of the data set source u, set the displacement δ to zero: $k \leftarrow 0$, $\delta^{(0)} \leftarrow 0$
2. Repeat the following:
   - Take a rotation matrix $R = [e_1, \cdots, e_N]$.
   - For every rotated axis i of the rotation, get the projections $f_i^{(k)}$ of $\{u + u^{(k)}\}$ and g_i of $\{v\}$.
   - For every rotation axis i, find the 1D transformation t_i that matches the marginals f_i into g_i.
   - Update the displacement

$$\delta^{(k+1)} = \delta^{(k)} + R \begin{bmatrix} t_1(e_1^{\mathrm{T}}(u+\delta^{(k)}) - e_1^{\mathrm{T}}(u+\delta^{(k)})) \\ \vdots \\ t_N(e_1^{\mathrm{T}}(u+\delta^{(k)}) - e_N^{\mathrm{T}}(u+\delta^{(k)})) \end{bmatrix}.$$

   - Update $k \leftarrow k+1$.

   until convergence on all marginals for every possible rotation.
3. The final one-to-one mapping T is given by $\forall\ j\ , u_j \mapsto t(u_j) = u_j + \delta_j^{(\infty)}$.

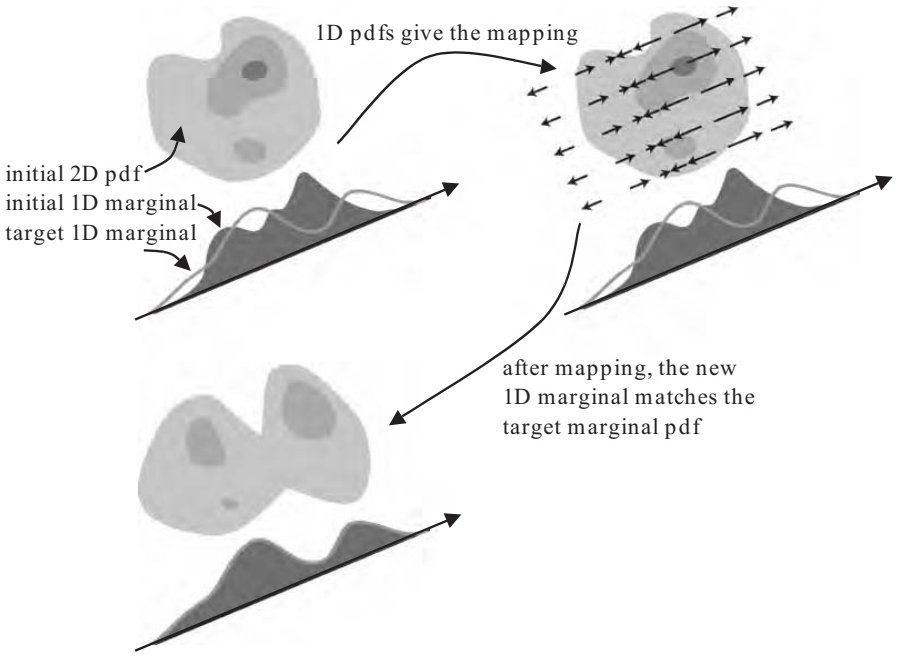


FIGURE 11.7

Illustration of the data manipulation, based on the 1D pdf transfer on one axis.

After transformation, the projection f'_e of the new distribution f' is now identical to g_e. The manipulation is explained in Figure 11.7. Considering that the operation can be done independently on orthogonal axes, the proposed manipulation consists in choosing an orthogonal basis $R = (e_1, \cdots, e_N)$ and then applying the following mapping:

$$u \rightarrow u + R \begin{bmatrix} t_1(e_1^{\mathrm{T}} u) - e_1^{\mathrm{T}} u \\ \vdots \\ t_N(e_N^{\mathrm{T}} u) - e_N^{\mathrm{T}} u \end{bmatrix} \tag{11.23}$$

where t_i is the 1D pdf transfer mapping for the axis e_i.

The idea is that iterating this manipulation over different axes will result in a sequence of distributions $f^{(k)}$ that hopefully converges to the target distribution g. The overall algorithm (Algorithm 11.1) will be referred to as the *iterative distribution transfer* (IDT).

A theoretical and numerical study of the method is developed in more depth in Reference [41]. The experimental study strongly suggests that convergence occurs for any distribution. The study also considers the problem of finding a sequence of axes that maximizes the convergence speed of the algorithm. The sequence of axes is designed to minimize the correlation between the directions. These directions for $N = 3$ are listed in Table 11.1.

As in the composition transfer case, the manipulations are based on the use of 1D marginals. The difference is that the marginals are not based on conditional probabilities. This means that the operation is independent of the channel ordering. Also, the estimation of the marginals does not suffer from the data sparseness and the N-dimensional pdf does not need to be smoothed. In contrast with the discrete Kantorovitch method, the method

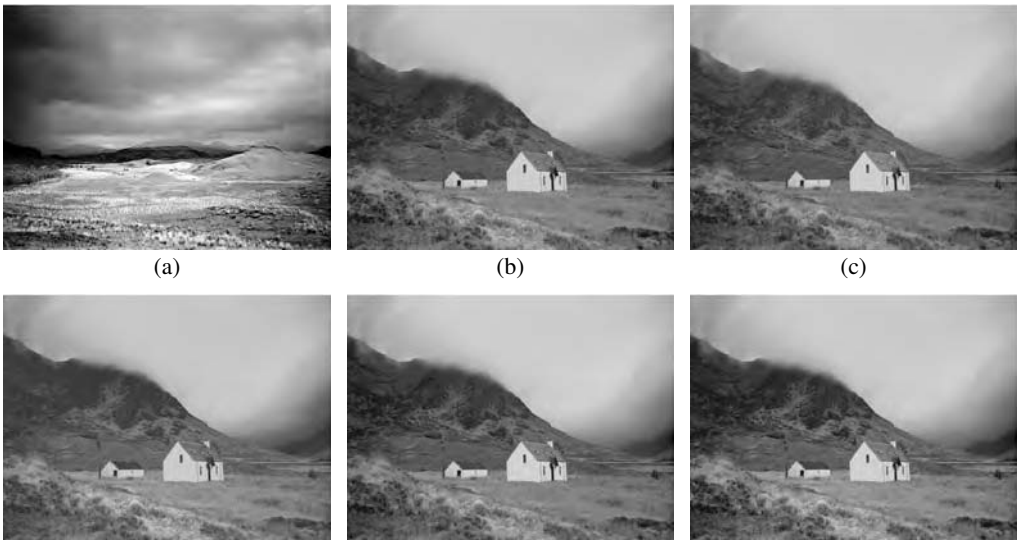
TABLE 11.1
Optimized rotations for $N = 3$.

No.	1			2		
x	1	0	0	0.333333	0.666667	0.666667
y	0	1	0	0.666667	0.333333	–0.666667
z	0	0	1	–0.666667	0.666667	–0.333333
No.	3			4		
x	0.577350	0.211297	0.788682	0.577350	0.408273	0.707092
y	–0.577350	0.788668	0.211352	–0.577350	–0.408224	0.707121
z	0.577350	0.577370	–0.577330	0.577350	–0.816497	0
No.	5			6		
x	0.332572	0.910758	0.244778	0.243799	0.910726	0.333376
y	–0.910887	0.242977	0.333536	0.910699	–0.333174	0.244177
z	–0.244295	0.333890	–0.910405	–0.333450	–0.244075	0.910625
No.	7			8		
x	–0.109199	0.810241	0.575834	0.759262	0.649435	–0.041906
y	0.645399	0.498377	–0.578862	0.143443	–0.104197	0.984158
z	0.756000	–0.308432	0.577351	0.634780	–0.753245	–0.172269
No.	9			10		
x	0.862298	0.503331	–0.055679	0.982488	0.149181	0.111631
y	–0.490221	0.802113	–0.341026	0.186103	–0.756525	–0.626926
z	–0.126988	0.321361	0.938404	–0.009074	0.636722	–0.771040
No.	11			12		
x	0.687077	–0.577557	–0.440855	0.463791	0.822404	0.329470
y	0.592440	0.796586	–0.120272	0.030607	–0.386537	0.921766
z	–0.420643	0.178544	–0.889484	–0.885416	0.417422	0.204444

is treated with one-to-one mapping all along, thus no dithering postprocess is necessary. Figure 11.5f shows that the results are quite similar to the discrete Kantorovitch solution.

11.5 What Color Space to Choose?

If a method can exactly transfer the complete color statistics, then this method will work regardless of the chosen color space. Thus, in that respect, the color space is not important to transfer the color *feel* of an image. The color space has however an influence on the geometrical form of the color mapping. In the MK formulation for instance, the cost of transportation is related to the color difference, which depends on the chosen color space. Thus uniform color spaces like YUV, or better CIELAB and CIELUV, are preferable to the basic RGB to obtain coherent color mappings. Comparative results for RGB, YUV, CIE XYZ, CIELAB and CIELUV are displayed in Figure 11.8 for the linear MK solution and in Figure 11.9 for the proposed IDT method. Differences might be difficult to see on printed



(a) (b) (c)

(d) (e) (f)

FIGURE 11.8 (See color insert.)

Results of the linear Monge-Kantorovitch transfer for different color spaces: (a) target color palette, (b) RGB, (c) YUV, (d) XYZ, (e) CIELAB, and (f) CIELUV.

(a) (b) (c)

(d) (e) (f)

FIGURE 11.9 (See color insert.)

Results of the IDT transfer for different color spaces: (a) target color palette, (b) RGB, (c) YUV, (d) XYZ, (e) CIELAB, and (f) CIELUV.

images, but these are very significant when displayed on a big screen, especially if they are images in a sequence which is played back. The CIELAB color space overall offers better renderings, for the linear and the nonlinear case. This is because it is designed to measure the difference between colors under different illuminants.

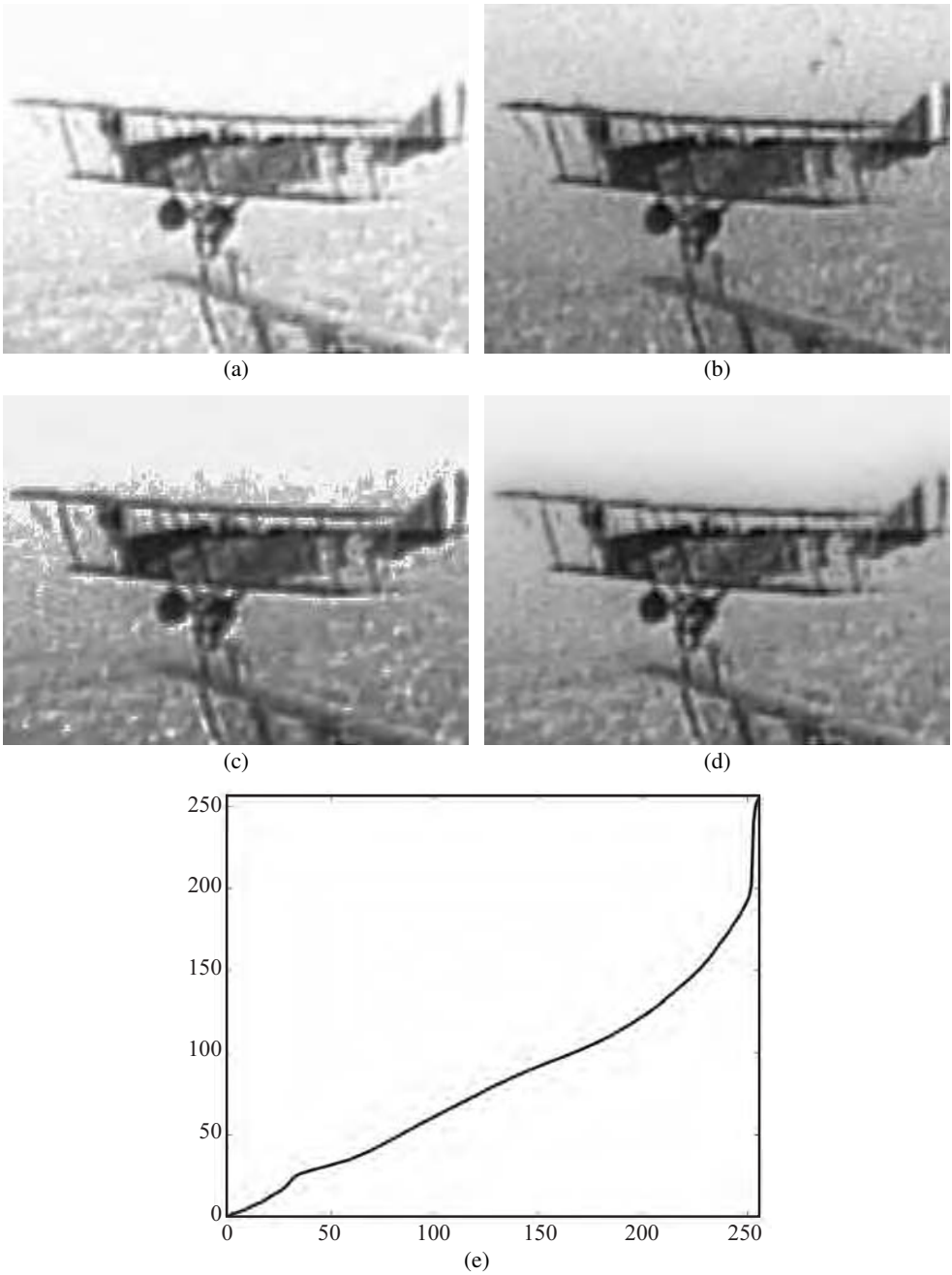


FIGURE 11.10

Result of grain reduction: (a, b) two consecutive archive frames suffering from extreme brightness variation known as flicker [45], (c) mapped original frame, (d) mapped original frame after grain artifact reduction, and (e) employed mapping transformation. Since the corresponding mapping transformation is overstretched, the mapped original frame has an increased level of noise. The proposed grain artifact reducer is able to reproduce the noise level of the original picture. The top of the original picture is saturated and cannot be retrieved but the algorithm succeeds in preserving the soft gradient.

(a) (b) (c)

FIGURE 11.11

Artifact grain reduction in color picture: (a) original image, (b) after IDT color transfer, and (c) after regraining.

11.6 Reducing Grain Noise Artifacts

Figure 11.10 and Figure 11.11 show that mapping the colors of the picture might produce some grain artifacts. When the content differs, or when the dynamic ranges of both pictures are too different, the resulting mapping function can be stretched on some parts (see Figure 11.10e), and thus enhances the noise level (see Figure 11.10c). This can be understood by taking the simple example of a linear transformation t of the original picture: $t(u) = a\ u + b$. The overall variance of the resulting picture is changed to $\text{var}\{t(u)\} = a^2\ \text{var}\{u\}$. This means that a greater stretching ($a > 1$) produces more noise.

The problem is that it is impossible to say a priori if the large difference in the distributions is due to a drastic color mapping, which is always possible, or due to a content variation. The difficulty of differentiating these cases is the Achilles' heel of example based color transfer techniques. Since any distribution can map any other distribution, it is impossible to impose any prior on the distributions. This issue remains an open problem but a few ad hoc solutions can help here. Solutions in the literature [11], [13] have been proposed to use a preprocessing smoothing operation on the pdfs. The motivation for these solutions, exposed hereafter, is to reduce the over-stretching of the mapping. This chapter also proposes a postprocessing method to protect the original picture gradient and thus reduce the amount of artifacts.

11.6.1 Reducing the Stretching by Adjusting the Distributions

One origin of the artifacts is that the target pdf is usually only a rough estimate of what the true palette should be. This means that using very accurate pdf approximations might lead to erroneous transfers. In that sense, using simple MVG approximations is always a safer solution. Also an advantage of linear mappings is that they have a constant stretching $|t'(u)| = |\Sigma_v|^{1/2}/|\Sigma_u|^{1/2}$ over the color distribution. The color deformation is thus also constant over the whole image and the artifacts are uniformly distributed over the picture.

Nonlinear mappings have however to be used when dealing with complex illuminations. In order to avoid unnecessary finesse in the approximations, what is desired is a mechanism that controls the level of approximation. This control can be achieved by smoothing the color histograms by a variable amount. The smoothing can be done by employing a kernel

filter K with bandwidth h. For instance, one can use the Epanechnikov kernel, which is the function $K_h(u) = (3/4)(1 - \|u/h\|^2)$ for $\|u/h\| < 1$ and zero for $\|u/h\|$ outside that range. The smoothness of the pdf is then controlled by the bandwidth parameter h. Decreasing h increases the detail level of the pdf.

Smoothing the pdf reduces distortions that are due to fine disparities, but does not specifically address the problem of change of color proportions. Consider the example, previously discussed, where the sky in one picture covers a larger area than in the other. The peak in the pdf corresponding to the blue color is at the same location in both pdfs, but the mass of this peak is then different. Smoothing the pdf solves this problem only partially. A solution used by Neumann [11] and Pitié [13] is to reduce the relative size of each peak in the pdfs. In this way, the variation in color proportions is limited. Combining both smoothing and this dominant color correction idea results in the following ad hoc smoothing operation on the pdf:

$$\tilde{f}(u) = (K_h * f(u) + \varepsilon)^{(1/p)} \tag{11.24}$$

where the exponent $p > 1$ controls the relative size of the pdf peaks and ε avoids problems when $f(u) = 0$. When working with 1D marginals, like the IDT algorithm does, the smoothing can be applied to the 1D marginals and not the original ND pdf.

Bear in mind however that this smoothing operation should not be used if the target distribution is actually the one desired, since in that case the resulting mapping would then be erroneous. The best solution is to leave most of the correction process to the regrain postprocessing which is explained in the following section.

11.6.2 Reducing the Artifacts by Adjusting the Gradient Fields

A solution to reduce the grain is to run a postprocessing algorithm that forces the noise level to remain the same. The idea presented by Pitié et al. [41] is to adjust the gradient field of the resulting picture so that it matches the gradient field of the original picture. If the gradient fields of both pictures are similar, the noise level will be the same. Matching the gradient of a picture has been addressed in different computer vision applications like high dynamic range compression [4]; the value of this idea has been thoroughly demonstrated by Pérez et al. in Reference [28]. Manipulating the image gradient can be efficiently done by using a variational approach. The problem here is slightly different, since recoloring also implies changing the contrast levels. Thus, the new gradient field should only loosely match the original gradient field.

Denote $I(x,y)$ the 3D original color picture. To simplify the discussion, coordinates are omitted in the expressions and I, J, ψ,ϕ, etc. actually refer to $I(x,y)$, $J(x,y)$, $\psi(x,y)$ and $\phi(x,y)$. Let $t : I \rightarrow t(I)$ be the color transformation. The problem is to find a modified image J of the mapped picture $t(I)$ that minimizes on the whole picture range Ω:

$$\min_J \iint_\Omega \phi \cdot ||\nabla J - \nabla I||^2 + \psi \cdot ||J - t(I)||^2 \mathrm{d}x\mathrm{d}y \tag{11.25}$$

with Neumann boundaries condition $\nabla J|_{\partial\Omega} = \nabla I|_{\partial\Omega}$ so that the gradient of J matches with the gradient of I at the picture border $\partial\Omega$. The term $||\nabla J - \nabla I||^2$ forces the image gradient to be preserved. The term $||J - t(I)||^2$ ensures that the colors remain close to the target picture

and thus protects the contrast changes. Without $||J-t(I)||^2$, a solution of Equation 11.25 will be actually the original picture I.

The weight fields $\phi(x,y)$ and $\psi(x,y)$ affect the importance of both terms. Many choices are possible for ϕ and ψ, and the following study could easily be changed, depending on the specifications of the problem.

The weight field $\phi(x,y)$ has been here chosen to emphasize that only flat areas have to remain flat but that gradient can change at object borders:

$$\phi = \frac{30}{1+10\,||\nabla I||} \tag{11.26}$$

The weight field $\psi(x,y)$ accounts for the possible stretching of the transformation t. Where ∇t is big, the grain becomes more visible:

$$\psi = \begin{cases} 2/\left(1+||(\nabla t)(I)||\right) & \text{if } ||\nabla I|| > 5 \\ ||\nabla I||/5 & \text{if } ||\nabla I|| \leq 5 \end{cases} \tag{11.27}$$

where $(\nabla t)(I)$ is the gradient of t for the color I and thus refers to the color stretching. The case $||\nabla I|| \leq 5$ is necessary to re-enforce that flat areas remain flat. While the gradient of t is easy to estimate for grayscale pictures, it might be more difficult to obtain for color mappings. The field can then be changed into:

$$\psi(x,y) = \begin{cases} 1 & \text{if } ||\nabla I|| > 5 \\ ||\nabla I||/5 & \text{if } ||\nabla I|| \leq 5 \end{cases} \tag{11.28}$$

The minimization problem in Equation 11.25 can be solved using the variational principle which states that the integral must satisfy the Euler-Lagrange equation:

$$\frac{\partial F}{\partial J} - \frac{d}{dx}\frac{\partial F}{\partial J_x} - \frac{d}{dy}\frac{\partial F}{\partial J_y} = 0 \tag{11.29}$$

where

$$F(J,\nabla J) = \phi \cdot ||\nabla J - \nabla I||^2 + \psi \cdot ||J - t(I)||^2 \tag{11.30}$$

from which the following can be derived:

$$\phi \cdot J - \mathrm{div}\left(\psi \cdot \nabla J\right) = \phi \cdot t(I) - \mathrm{div}\left(\psi \cdot \nabla I\right) \tag{11.31}$$

The above is an elliptic partial differential equation. The expression $\mathrm{div}\left(\psi \cdot \nabla I\right)$ at pixel $\mathbf{x} = (x,y)$ can be approximated using standard finite differences [43] by:

$$\mathrm{div}\left(\psi \cdot \nabla I\right)(\mathbf{x}) \approx \sum_{\mathbf{x}_n \in \mathcal{N}_\mathbf{x}} \frac{\psi_{\mathbf{x}_n} + \psi_\mathbf{x}}{2}\left(I_{\mathbf{x}_n} - I_\mathbf{x}\right) \tag{11.32}$$

where $\mathcal{N}_\mathbf{x}$ corresponds to the four neighboring pixels of $\mathbf{x}$. Using this in Equation 11.31 yields a linear system as follows:

$$\begin{aligned} a_6(x,y) = {} & a_1(x,y)J(x,y-1) + a_2(x,y)J(x,y+1) \\ & + a_3(x,y)J(x-1,y) + a_4(x,y)J(x+1,y) + a_5(x,y)J(x,y) \end{aligned} \tag{11.33}$$

with

$$a_1(x,y) = -\frac{\psi(x,y-1)+\psi(x,y)}{2} \tag{11.34}$$

$$a_2(x,y) = -\frac{\psi(x,y+1)+\psi(x,y)}{2} \tag{11.35}$$

$$a_3(x,y) = -\frac{\psi(x-1,y)+\psi(x,y)}{2} \tag{11.36}$$

$$a_4(x,y) = -\frac{\psi(x+1,y)+\psi(x,y)}{2} \tag{11.37}$$

$$\begin{aligned} a_5(x,y) = \frac{1}{2}\Big(&4\psi(x,y)+\psi(x,y-1)+\psi(x,y+1)\\ &+\psi(x-1,y)+\psi(x+1,y)\Big)+\phi(x,y) \end{aligned} \tag{11.38}$$

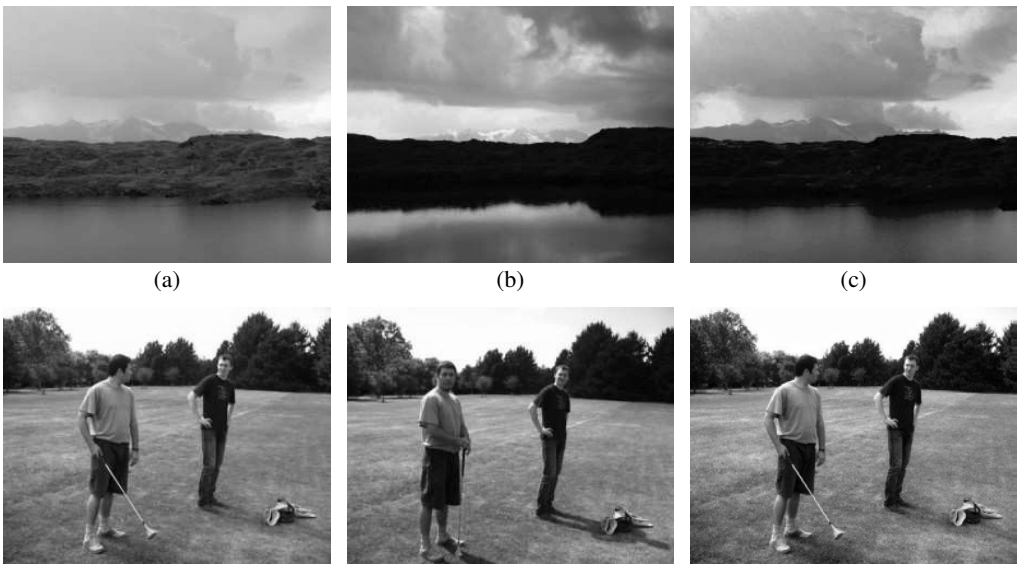
$$\begin{aligned} a_6(x,y) = \frac{1}{2}\Big(&(\psi(x,y)+\psi(x,y-1))(I(x,y-1)-I(x,y))\\ &+(\psi(x,y)+\psi(x,y+1))(I(x,y+1)-I(x,y))\\ &+(\psi(x,y)+\psi(x-1,y))(I(x-1,y)-I(x,y))\\ &+(\psi(x,y)+\psi(x+1,y))(I(x+1,y)-I(x,y))\Big)\phi(x,y)I(x,y) \end{aligned} \tag{11.39}$$

The system can be solved by standard iterative methods like SOR or Gauss-Seidel with multigrid approach. Implementations of these numerical solvers are widely available and one can refer for instance to Reference [44]. The main step of these methods is to solve iteratively for $J(x,y)$. Note that $J(x,y)$ and $a_i(x,y)$ are of dimension 3, but that each color component can be treated independently. For instance, the iteration for the red component field is of the form

$$\begin{aligned} J_R^{(k+1)}(x,y) = \frac{1}{a_5^R(x,y)}\Big(&a_5^R(x,y)-a_1^R(x,y)J_R^{(k)}(x,y-1)-a_2^R(x,y)J_R^{(k)}(x,y+1)\\ &-a_3^R(x,y)J_R^{(k)}(x-1,y)-a_4^R(x,y)J_R^{(k)}(x+1,y)\Big) \end{aligned} \tag{11.40}$$

where $J_R^{(k)}(x,y)$ is the result in the red component at the k^{th} iteration.

The overall method takes less than a second per image at PAL resolution (720×576) on a 2GHz machine using a commercial implementation of the method. Figure 11.10d and Figure 11.11c show the efficiency of the method. In Figure 11.10d, the top of the original frame is clamped, thus resulting in the loss of grain texture. Note that the regraining method is well designed for this kind of situation where the mapping is actually correct and artifacts only come from the discrete nature of the image. The case of Figure 11.11 is more difficult since the target color pdf can only be a crude approximation of what is desired. However, the regraining tool is quite successful at recovering the original gradient information in the resulting image (Figure 11.11c).



(a) (b) (c)

(d) (e) (f)

FIGURE 11.12 (See color insert.)

Examples of color grading for matching lighting conditions: (a-c) the color properties of the sunset are used to synthesize the evening scene depicted at sunset, (d-f) the color grading corrects the change of lighting conditions induced by clouds, and (c, f) the color transfer achieved by employing IDT followed by the regraining process.

(a) (b) (c)

(d) (e)

FIGURE 11.13 (See color insert.)

Example of color grading for image and video restoration used to recreate different atmospheres: (a) original frame, (b) 1970s atmosphere, (c) pub atmosphere, (d) linear MK result using 1970s atmosphere, and (e) IDT followed by regraining using pub atmosphere.

11.7 Application Results

The color transfer techniques are tested here for some color grading applications. Considering the advantages of the MK solution for the linear case and of the IDT method for the nonlinear case, only these methods will be used in the following examples. Also, the IDT will be systematically used in conjunction with the regraining process. The algorithms work in the CIELAB color space.

Matching lighting conditions is illustrated in Figure 11.12 in two typical situations. In the first case, the color properties of the sunset (Figure 11.12b) are used to synthesize the *evening* scene (Figure 11.12a) depicted at sunset (Figure 11.12c). This kind of grading is frequent when shooting a movie at sunset since the light is changing quickly. To be able to cope with the nonlinearity of the contrast change, the color transfer is performed by using IDT. In the second case (Figure 11.12d to Figure 11.12f), the color grading corrects another classical change of lighting conditions due to passing clouds. Even when using the grain artifact reducer, an unavoidable limitation of color grading is the clipping of the color data: saturated areas cannot be retrieved (for instance, the sky on the golf image cannot be recovered). A general rule is to match pictures from higher to lower range dynamics.

Figure 11.13 displays examples of *movie restoration* using color grading. The idea is similar to color grading, i.e., to enhance the color and match a desired atmosphere. Experimentation showed that the linear MK method can be enough to enhance properly the faded color palette. The IDT can however recreate a wider variety of grades than the linear MK method. For instance, the IDT can produce both images shown in Figure 11.13d and Figure 11.13e whereas the linear MK solution will fail at reproducing the strong contrasts of Figure 11.13e. To obtain Figure 11.13e, the smoothing process (see Section 11.6.1) has been used before starting the IDT.

Figure 11.14 and Figure 11.15a to Figure 11.15d present two examples of *color grading for photography*. Because images are not played back in photography, the consistency is not as critical as it is in movie postproduction. If the IDT method provides the closest color feel to the target picture, the linear MK transformation can sometimes satisfy aesthetically the artist and serves as a starting point for further editing.

It is possible with the IDT algorithm to perform true *color equalization* by choosing the uniform distribution as a target. As illustrated in Figure 11.15e, the equalization process puts the whole color spectrum in the image. Note that even though the mapping is extremely stretched, the smoothness of the picture is still preserved by the regraining process.

11.8 Parting Remarks

This chapter has presented a review of color transfer techniques that are based on one-to-one color mappings. It has also established that these methods have a common mathematical background which is the mass transportation theory and that a good way of dealing with the problem is to use the Monge-Kantorovitch mappings. Monge-Kantorovitch map-

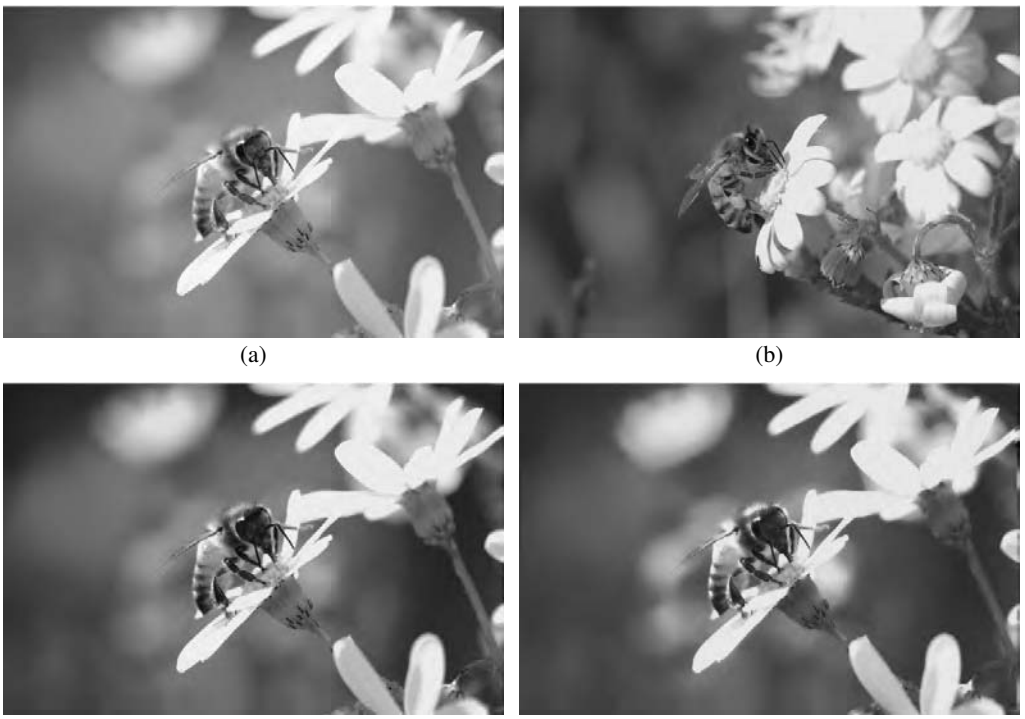


FIGURE 11.14 (See color insert.)

Color grading results: (a) original image, (b) target palette, (c) linear MK result, and (d) IDT result.

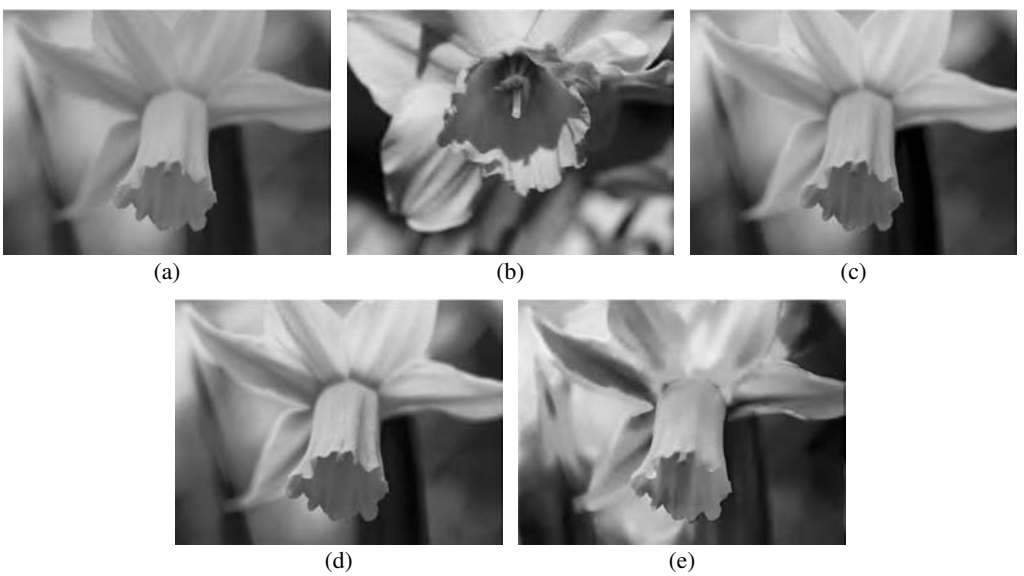


FIGURE 11.15 (See color insert.)

Color grading results: (a) original image, (b) target palette, (c) linear MK result, (d) IDT followed by the regraining process resulting in better color contrast, (e) original image recolored with the whole color spectrum using the IDT algorithm followed by the regraining process.

pings have intuitive geometrical properties and turn out to be robust when used in real applications. Thus finding mappings that transfer color statistics is now a well understood problem. The algorithms that have been proposed here, that is, the linear MK solution and the nonlinear IDT method, have already been implemented in industrial applications and are currently used by artists. Grain artifacts resulting from the mapping correction have also been addressed and a practical solution has been found and can be used in tandem with the IDT color transfer.

It is important to realize that one-to-one color transfer techniques are not the universal answer to color grading. The methods are indeed limited by the ability of one-to-one mapping to model change of color grade. Also, these methods assume that the target image contains the exact color distribution. In practice however, target images are only an approximation of the desired color palette. Dealing with content variations is still an open problem, even though some pre and post processing can improve the robustness.

The problems and techniques discussed in this chapter can be viewed as a set of tools that one can use confidently, provided that their conditions of use are well understood.

Acknowledgments

The authors would like to acknowledge the helpful discussions with the people at The Foundry and GreenParrotPictures as well as Dr. Naomi Harte for her precious help.

References

[1] M. Pappas and I. Pitas, "Digital color restoration of old paintings," *Transactions on Image Processing*, vol. 9, no. 2, pp. 291–294, February 2000.

[2] L. Lucchese and S.K. Mitra, "A new method for color image equalization," in *Proceedings of the IEEE International Conference on Image Processing*, Thessaloniki, Greece, October 2001, vol. 1, pp. 133–136.

[3] E. Pichon, M. Niethammer, and G. Sapiro, "Color histogram equalization through mesh deformation," in *Proceedings of the IEEE International Conference on Image Processing*, Barcelona, Spain, September 2003, vol. 3, pp. 117–120.

[4] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, San Antonio, TX, USA, July 2002, pp. 249–256.

[5] P. Debevec, E. Reinhard, G. Ward, and S. Pattanaik, "High dynamic range imaging," in *ACM SIGGRAPH 2004 Course Notes*, 2004, p. 14.

[6] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 227–280, July 2002.

[7] Y. Ji, H.B. Liu, X.K. Wang, and Y.Y. Tang, "Color transfer to greyscale images using texture spectrum," in *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, Shanghai, China, August 2004, vol. 7, pp. 4057–4061.

[8] A.A. Gooch, S.C. Olsen, J. Tumblin, and B. Gooch, "Color2Gray: Salience-preserving color removal," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 634–639, July 2005.

[9] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics Applications*, vol. 21, no. 5, pp. 34–41, September / October 2001.

[10] A. Abadpour and S. Kasaei, "A fast and efficient fuzzy color transfer method," in *Proceedings of the IEEE Symposium on Signal Processing and Information Technology*, Athens, Greece, December 2005, pp. 491–494.

[11] L. Neumann and A. Neumann, "Color style transfer techniques using hue, lightness and saturation histogram matching," in *Proceedings of the Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*, Girona, Spain, May 2005, pp. 111–122.

[12] C.M. Wang and Y.H. Huang, "A novel color transfer algorithm for image sequences," *Journal of Information Science and Engineering*, vol. 20, no. 6, pp. 1039–1056, November 2004.

[13] F. Pitié, A. Kokaram, and R. Dahyot, "N-dimensional probability density function transfer and its application to colour transfer," in *Proceedings of the International Conference on Computer Vision*, Beijing, China, October 2005, vol. 2, pp. 1434–1439.

[14] H.L. Shen and J.H. Xin, "Transferring color between three-dimensional objects," *Applied Optics*, vol. 44, no. 10, pp. 1969–1976, April 2005.

[15] A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the IEEE International Conference on Computer Vision*, Corfu, Greece, September 1999, pp. 1033–1038.

[16] A. Efros and W. Freeman, "Image quilting for texture synthesis and transfer," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA, August 2001, pp. 341–346.

[17] A. Hertzmann, C. Jacobsk, N. Oliver, B. Curless, and D. Salesin, "Image analogies," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, CA, USA, August 2001, pp. 327–340.

[18] S. Bae, S. Paris, and F. Durand, "Two-scale tone management for photographic look," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 637–645, July 2006.

[19] H. Kotera, "A scene-referred color transfer for pleasant imaging on display," in *Proceedings of the IEEE International Conference on Image Processing*, Genoa, Italy, September 2005, vol. 2, pp. 5–8.

[20] J. Morovic and P.L. Sun, "Accurate 3D image colour histogram transformation," *Pattern Recognition Letters*, vol. 24, no. 11, pp. 1725–1735, July 2003.

[21] L.C. Evans, "Partial differential equations and Monge-Kantorovich mass transfer," *Current Developments in Mathematics*, pp. 65–126, 1998.

[22] W. Gangbo and R. McCann, "The geometry of optimal transport," *Acta Mathematica*, vol. 177, pp. 113–161, 1996.

[23] C. Villani, *Topics in Optimal Transportation*. Providence, RI: American Mathematical Society, 2003.

[24] J. Jia, J. Sun, C.K. Tang, and H.Y. Shum, "Bayesian correction of image intensity with spatial consideration," in *Proceedings of the 8th European Conference on Computer Vision*, Prague, Czech Republic, May 2004, pp. 342–354.

[25] Y. Chang, S. Saito, K. Uchikawa, and M. Nakajima, "Example-based color stylization of images," *ACM Transactions on Applied Perception*, vol. 2, no. 3, pp. 322–345, July 2005.

[26] R. Lukac, B. Smolka, K. Martin, K.N. Plataniotis, and A.N. Venetsanopoulos, "Vector filtering for color imaging," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 74–86, January 2005.

[27] R. Lukac and K.N. Plataniotis, "A taxonomy of color image filtering and enhancement solutions," *Advances in Imaging and Electron Physics*, P.W. Hawkes (ed.), San Diego, CA: Elsevier / Academic Press, vol. 140, pp. 187–264, June 2006.

[28] P. Pérez, A. Blake, and M. Gangnet, "Poisson image editing," *ACM Transactions on Graphics*, vol. 232, no. 3, pp. 313–318, July 2003.

[29] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*. London, UK: Chapman & Hall, 1986.

[30] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Boston, MA, USA: Addison Wesley, 1992.

[31] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 267–276, July 2002.

[32] D.L. Ruderman, T.W. Cronin, and C.C. Chiao, "Statistics of cone responses to natural images: Implications for visual coding," *Journal of the Optical Society of America*, vol. 15, no. 8, pp. 2036–2045, August 1998.

[33] F. Pitié, "Statistical Signal Processing Techniques for Visual Post-Production," Ph.D. thesis, University of Dublin, Trinity College, April 2006.

[34] A. Abadpour and S. Kasaei, "An efficient PCA-based color transfer method," *Journal of Visual Communication and Image Representation*, vol. 18, no. 1, pp. 15–34, February 2007.

[35] H.J. Trussell and M.J. Vrhel, "A fast and efficient fuzzy color transfer method," in *Proceedings of the SPIE*, vol. 1452, pp. 2–9, June 1991.

[36] I. Olkin and F. Pukelsheim, "The distance between two random vectors with given dispersion matrices," *Linear Algebra and its Applications*, vol. 48, pp. 257–263, 1982.

[37] M. Grundland and N.A. Dodgson, "Color histogram specification by histogram warping," in *Proceedings of the SPIE Conference on Color Imaging X: Processing, Hardcopy, and Applications*, San Jose, CA, USA, January 2005, vol. 5667, pp. 610–624.

[38] Y. Rubner, C. Tomasi, and L.J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal Computer Vision*, vol. 40, no. 2, pp. 99–121, October 2000.

[39] F.L. Hitchcock, "The distribution of a product from several sources to numerous localities," *Journal of Mathematics and Physics*, vol. 20, pp. 224–230, 1941.

[40] N. Wu and R. Coppins, *Linear Programming and Extensions*. New York: McGraw-Hill, 1981.

[41] F. Pitié, A. Kokaram, and R. Dahyot, "Automated colour grading using colour distribution transfer," *Journal of Computer Vision and Image Understanding*, vol. 107, no. 1–2, pp. 123–137, July / August 2007.

[42] S. Helgason, *The Radon Transform, 2nd edition*. Basel, Switzerland: Birkhäuser, 1999.

[43] J. Weickert, B. ter Haar Romeny, and M. Viergever, "Efficient and reliable schemes for nonlinear diffusion filtering," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 398–410, March 1998.

[44] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. New York: Cambridge University Press, 1992.

[45] F. Pitié, R. Dahyot, F. Kelly, and A.C. Kokaram, "A new robust technique for stabilizing brightness fluctuations," in *Proceedings of the 2nd Workshop on Statistical Methods in Video Processing*, Prague, Czech Republic, May 2004, pp. 153–164.

12

Exposure Correction for Imaging Devices: An Overview

Sebastiano Battiato, Giuseppe Messina, and Alfio Castorina

12.1 Introduction

One of the main problems affecting image quality, leading to unpleasant pictures, comes from improper exposure to light. Beside the sophisticated features incorporated in today's cameras (i.e., automatic gain control algorithms), failures are not unlikely to occur. Digital

consumer devices make use of ad-hoc strategies and heuristics to derive exposure setting parameters. Typically such techniques are completely blind with respect to the specific content of the scene. Some techniques are completely automatic, such as those based on *average / automatic exposure metering* or more complex *matrix / intelligent exposure metering*. Others provide the photographer with a certain control over the selection of the exposure, thus allowing space for personal taste or enabling the user to satisfy particular needs. In spite of the great variety of methods for regulating the exposure and the complexity of some of them, it is not rare for images to be acquired with a nonoptimal or incorrect exposure. This is particularly true for handset devices (e.g., mobile phones) where several factors contribute to badly exposed pictures: poor optics, absence of flash, complex scene lighting conditions, and so forth.

There is no exact definition of what a correct exposure should be. It is possible to abstract a generalization and to define the best exposure that enables one to reproduce the most important regions (according to contextual or perceptive criteria) with a level of gray or brightness, more or less in the middle of the possible range. In any case, if the dynamic range of the scene is sensibly high, there is no way to acquire the overall details. One of the main issues of this chapter is devoted to giving an effective overview of the technical details involved in:

- exposure settings of imaging devices before acquisition phase (i.e., pre-capture phase) [1];
- content-dependent enhancement strategies applied as postprocessing [2];
- advanced solutions where multi-picture acquisition of the same scene with different exposure time allows to reproduce the radiance map of the real world [3].

.

Namely, Section 12.2 discusses in detail traditional and advanced approaches related to the pre-capture phase (i.e., the sensor is read continuously and the output is analyzed in order to determine a set of parameters strictly related with the quality of the final picture [1]). The role of exposure setting will be analyzed by considering some case studies where, by making use of some assumptions about the dynamic range of the real scene, it is possible to derive effective strategies. Section 12.3 describes the work presented in Reference [2] where by using postprocessing techniques an effective enhancement has been obtained through analysis of some content dependent features of the picture. Section 12.4 surveys advanced approaches devoted to improving acquiring capabilities by using multi-picture acquisition (i.e., bracketing). In particular, this section focuses on popular techniques able to reproduce effectively the salient part of a real scene after having computed a reliable high dynamic range (HDR) [3]. Finally, conclusions are offered in Section 12.5.

12.2 Exposure Metering Techniques

Metering techniques built into the camera are improving with the introduction of computers; however, limitations still remain. For example, taking a picture of a snow scene or trying to photograph a black locomotive without overriding the camera calculated metering is very difficult. The most important aspect of the exposure duration is to guarantee that the

acquired image falls in a good region of the sensor's sensitivity range. In many devices, the selected exposure value is the main processing step for adjusting the overall image intensity that the consumer will see. Many of older digital cameras used a separate metering system to set exposure duration, rather than using data acquired from the sensor chip. Integrating exposure-metering function into the main sensor (through-the-lens, or TTL, metering) may reduce system cost. The imaging community uses a measure called *exposure value* (EV) to specify the relationship between the f-number[1], F, and exposure duration, T:

$$EV = \log_2\left(\frac{F^2}{T}\right) = 2\log_2(F) - \log_2(T) \tag{12.1}$$

The exposure value (Equation 12.1) becomes smaller as the exposure duration increases, and it becomes larger as the f-number grows. Most auto-exposure algorithms work in the following manner:

1. Take a picture with a pre-determined exposure value, EV_{pre};
2. Convert the RGB values to brightness, B;
3. Derive a single value B_{pre} (like center-weighted mean, median, or more complicated weighted method as in matrix-metering) from the brightness picture;
4. Based on linearity assumption and Equation 12.1, the optimum exposure value EV_{opt} should be the one that permits a correct exposure. The picture taken at this EV_{opt} should give a number close to a pre-defined ideal value B_{opt}, thus:

$$EV_{opt} = EV_{pre} + \log_2(B_{pre}) - \log_2(B_{opt}) \tag{12.2}$$

The ideal value B_{opt} for each algorithm is typically selected empirically. Different algorithms mainly differ in how the single number B_{pre} is derived from the picture.

12.2.1 Classical Approaches

The metering system in a digital camera measures the amount of light in the scene and calculates the best-fit exposure value based on the metering mode explained below. Automatic exposure is a standard feature in all digital cameras. After having selected the metering mode, the picture is captured by pointing the camera and pressing the shutter release. The metering method defines which information of the scene is used to calculate the exposure value and how it is determined. Cameras generally allow the user to select between spot, center-weighted average, or multi-zone metering modes.

12.2.1.1 Spot Metering

Spot metering allows the user to meter the subject in the center of the frame (or on some cameras at the selected *AutoFocus* (AF) point). Only a small area of the whole frame

[1] Aperture values or f-numbers, are measurement of the size of the hole that the light passes through the rear of the lens, relative to the focal length. The smaller the f-number, the more light gets through the lens.

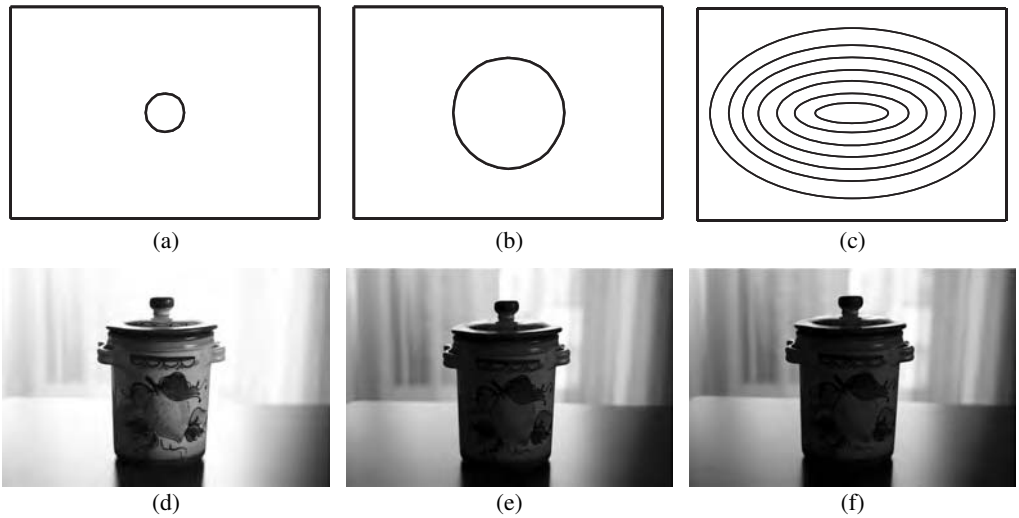


FIGURE 12.1

Metering examples: (a) spot area, (b) partial area, (c) center-weighted area, (d) spot example, (e) partial area example, and (f) center-weighted area example.

(between 1-5% of the viewfinder area) is metered while the rest of the frame is ignored. In this case, B_{pre} in Equation 12.2 is the mean of the center area (Figure 12.1a). This will typically be the effective center of the scene, but some cameras allow the user to select a different off-center spot, or to recompose by moving the camera after metering. A few models support a multi-spot mode which allows multiple spot meter readings to be taken of a scene that are averaged. Some cameras also support metering of highlight and shadow areas. Spot metering is very accurate and is not influenced by other areas in the frame. It is commonly used to shoot very high contrast scenes. For example (see Figure 12.1d), if the subject's back is being hit by the rising sun and the face is a lot darker than the bright halo around the subject's back and hairline (the subject is "backlit"), spot metering allows the photographer to measure the light bouncing off the subject's face and expose properly for that, instead of the much brighter light around the hairline. The area around the back and hairline will then become over-exposed. Spot metering is a method upon which the zone system depends.[2]

12.2.1.2 Partial Area Metering

This mode meters a larger area than spot metering (around 10–15% of the entire frame), and is generally used when very bright or very dark areas on the edges of the frame would otherwise influence the metering unduly. Like spot metering, some cameras can use variable points to take readings from (in general autofocus points), or have a fixed point in the

[2]The Zone System is a photographic technique for determining optimal film exposure and development, formulated by Ansel Adams and Fred Archer in 1941. The Zone System provides photographers with a systematic method of precisely defining the relationship between the way they visualize the photographic subject and the final results. Although it originated with black and white sheet film, the Zone System is also applicable to roll film (black and white, color, negative, and reversal) and to digital photography.

center of the viewfinder. Figure 12.1e shows an example of partial metering on a backlight scene; this method permits to equalize much more the global exposure.

12.2.1.3 Center-Weighted Average Metering

This method is probably the most common metering method implemented in nearly every digital camera: it is also the default for those digital cameras which do not offer metering mode selection. In this system, as described in Figure 12.1c, the meter concentrates between 60 to 80 percent of the sensitivity towards the central part of the viewfinder. The balance is then "feathered" out towards the edges. Some cameras allow the user to adjust the weight/balance of the central portion to the peripheral one. One advantage of this method is that it is less influenced by small areas that vary greatly in brightness at the edges of the viewfinder; as many subjects are in the central part of the frame, consistent results can be obtained. Unfortunately, if a backlight is present in the scene, the central part appears darker than the rest of the scene (Figure 12.1f), and an unpleasant underexposed foreground is produced.

12.2.1.4 Average Metering

In this mode, the camera will use the light information coming from the entire scene and averages for the final exposure setting, giving no weighting to any particular portion of the metered area. This metering technique has been replaced by center-weighted metering, thus is obsolete and only present in older cameras.

12.2.2 Advanced Approaches

12.2.2.1 Matrix or Multi-Zone Metering

This mode is also called matrix, evaluative, honeycomb, segment metering, or esp (electro selective pattern) metering on some cameras. It was first introduced by the Nikon FA, where it was called *automatic multi-pattern metering*. On a number of cameras, this is the default/standard metering setting. The camera measures the light intensity in several points of the scene, and then combines the results to find the settings for the best exposure. How they are combined/calculated deviates from camera to camera. The actual number of zones used varies wildly, from several to over a thousand. However performance should not be concluded on the number of zones alone, or the layout. As shown in Figure 12.2, the layout can change drastically from a manufacturer to another; also within the same company the use of different multi-zone metering can change due to several reasons (e.g., the dimension of the final pixel matrix).

Many manufacturers are not open about the exact calculations used to determine the exposure. A number of factors are taken into consideration; these include: AF point, distance to subject, areas in-focus or out-of-focus, colors / hues of the scene and backlighting. Multi-zone tends to bias its exposure towards the autofocus point being used (while taking into account other areas of the frame too), thus ensuring that the point of interest has been properly exposed. It is also designed to avoid the need to use exposure compensation in most situations. A database of many thousands of exposures is pre-stored in the camera, and the processor can use a selective pattern to determine what is being photographed.

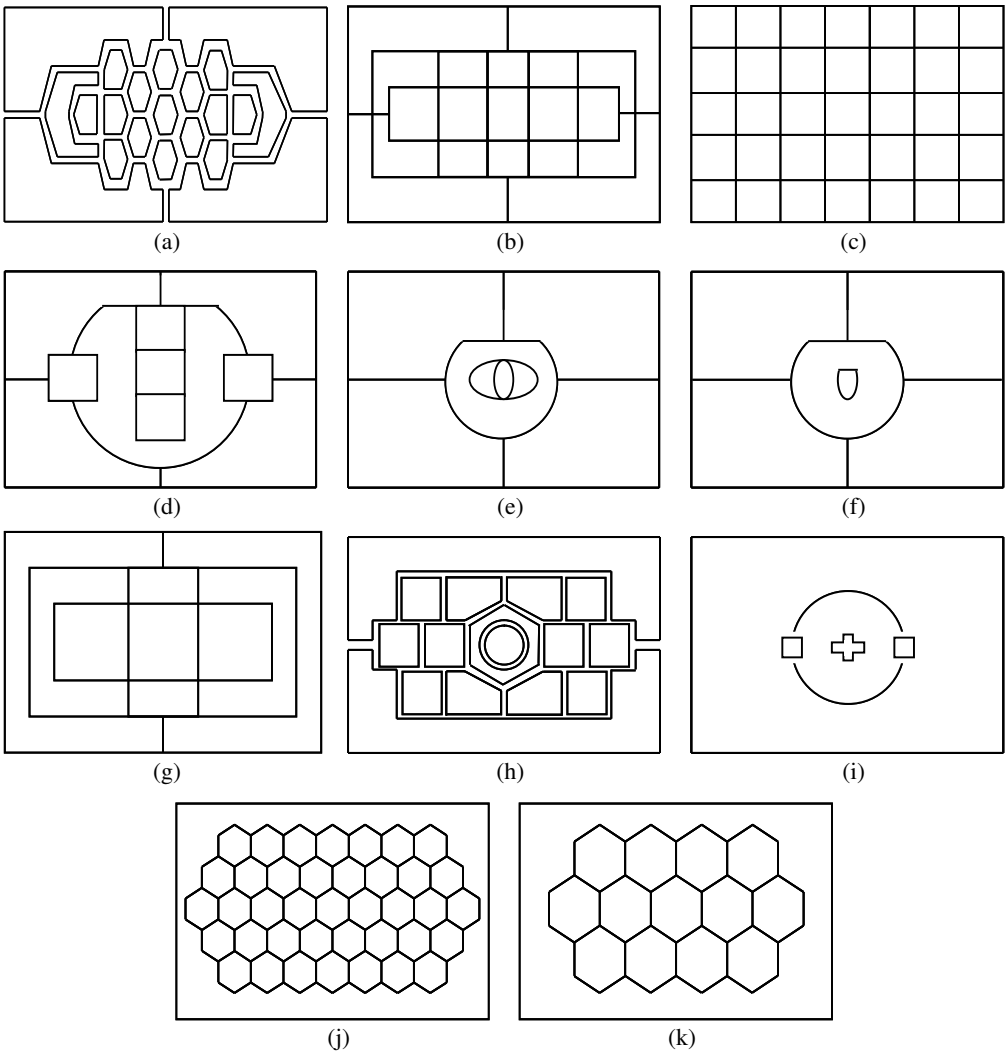


FIGURE 12.2

Examples of different kinds of multi-zone metering modes used by several camera manufacturers: (a) Canon 21 zones, (b) Canon 16 zones, (c) Canon 35 zones, (d) Nikon 10 segments, (e) Nikon 7 segments, (f) Nikon 6 segments, (g) Sigma 9 zones, (h) Samsung 16 zones, (i) Olympus ESP, (j) Konica Minolta 40 zone honeycombs, (k) Konica Minolta 14 zone honeycombs.

Some cameras allow the user to link (or unlink) the autofocus and metering, giving the possibility to lock exposure once AF confirmation is achieved. This is called *auto exposure lock* (AEL). Using manual focus, on many compact cameras, the AF point is not used as part of the exposure calculation. In such instances it is common for the metering to default to a central point in the viewfinder, using a pattern based off of that area. Some users have problems with wide angle shots in high contrast, due to the large area which can vary greatly in brightness; it is important to understand that even in this situation, the focus point can be critical to the overall exposure.

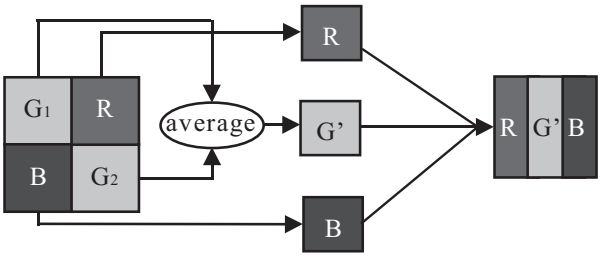


FIGURE 12.3

Bayer data subsampling generation.

12.3 Exposure Correction Content Dependent

As explained in Section 12.2, it is possible to define the best exposure able to reproduce the most important regions (according to contextual or perceptive criteria) with a level of gray or brightness, more or less in the middle of the possible range. After acquisition phase, typical postprocessing techniques try to realize an effective enhancement via global approaches, such as histogram specification, histogram equalization and gamma correction to improve global contrast appearance [4] by stretching the global distribution of the intensity. More adaptive criterions are needed to overcome such drawbacks. The exposure correction technique [2] described in this section is designed essentially for mobile sensor applications. This new element, present in newest mobile devices, is particularly harmed by "backlight" when the user utilizes a mobile device for video phoning. The detection of skin characteristics in captured images allows selection and proper enhancement and/or tracking of regions of interest (e.g., faces). If no skin is present in the scene the algorithm switches automatically to other features (such as contrast and focus) tracking for visually relevant regions. This implementation differs from the algorithm described in Reference [5] because the whole processing can also be performed directly on Bayer pattern images [6] (for detailed information on Bayer pattern and single-sensor imaging fundamentals refer to Chapter 1), and simpler statistical measures were used to identify *information carrying* regions. The algorithm is defined as follows:

1. Luminance extraction. If the algorithm is applied on Bayer data, in place of the three full color planes, a subsampled (quarter size) approximated input data (Figure 12.3) is used.

2. Using a suitable feature extraction technique the algorithm fixes a value to each region. This operation permits to seek visually relevant regions (for contrast and focus, the regions are block based; whereas for skin detection, the regions are associated to each pixel).

3. Once the visually important pixels are identified (e.g., the pixels belonging to skin features) a global tone correction technique is applied using as main parameter the mean gray level of the relevant regions.

12.3.1 Feature Extraction: Contrast and Focus

To identify regions of the image that contain more information, luminance plane is subdivided in N blocks of equal dimensions (in our experiments we employed $N = 64$ for VGA images). For each block, statistical measures of *contrast* and *focus* are computed. Therefore, it is assumed that well focused or high-contrast blocks are more relevant compared to the others. Contrast refers to the range of tones present in the image. A high contrast leads to a higher number of perceptually significant regions inside a block. Focus characterizes the sharpness or edgeness of the block and is useful in identifying regions where high frequency components (i.e., details) are present. If the aforementioned measures were simply computed on highly underexposed images, then the regions having better exposure would always have higher contrast and edgeness compared to those that are obscured. In order to perform a visual analysis revealing the most important features regardless of lighting conditions, a new *visibility image* is constructed by pushing the mean gray level of the input green Bayer pattern plane (or the Y channel for color images) to 128. The push operation is performed using the same function that is used to adjust the exposure level and it will be described later. The contrast measure is computed by simply building a histogram for each block and then calculating its deviation from the mean value. A high deviation value denotes good contrast and vice versa. In order to remove irrelevant peaks, the histogram is slightly smoothed by replacing each entry with its mean in a ray 2 neighborhood. Thus, the original histogram entry is replaced with the gray-level $\tilde{I}[i]$:

$$\tilde{I}[i] = \frac{(I[i-2] + I[i-1] + I[i] + I[i+1] + I[i+2])}{5} \tag{12.3}$$

Histogram deviation D is computed as:

$$D = \frac{\sum_{i=0}^{255} |i - M| \cdot \tilde{I}[i]}{\sum_{i=0}^{255} \tilde{I}[i]} \tag{12.4}$$

where M is the mean value:

$$M = \frac{\sum_{i=0}^{255} i \cdot \tilde{I}[i]}{\sum_{i=0}^{255} \tilde{I}[i]} \tag{12.5}$$

The focus measure is computed by convolving each block with a simple 3×3 Laplacian filter.

In order to discard irrelevant high frequency pixels (mostly noise), the output of the convolution at each pixel location is thresholded. The mean focus value of each block is computed as:

$$F = \frac{\sum_{i=1}^{N} thresh_{\tau}\big(lapl(i)\big)}{N} \tag{12.6}$$

where N is the number of pixels, $lapl(i)$ is the output of the convolution of each block with a 3×3 Laplacian kernel, and the $thresh_{\tau}(\cdot)$ operator discards values lower than a fixed threshold τ. Once the values F and D are computed for all blocks, relevant regions will be classified using a linear combination of both values. Feature extraction pipeline is illustrated in Figure 12.4.

(a)

(b)

m_1	m_2	m_3	m_4	m_5
m_6	m_7	m_8	m_9	m_{10}
m_{11}	m_{12}	m_{13}	m_{14}	m_{15}
m_{16}	m_{17}	m_{18}	m_{19}	m_{20}
m_{21}	m_{22}	m_{23}	m_{24}	m_{25}

(c)

(d)

FIGURE 12.4

Feature extraction pipeline for focus and contrast with $N = 25$: (a) input image, (b) luminance blocks, (c) relevance measures, and (d) relevant blocks. Visual relevance of each luminance block of the input image is based on relevance measures able to obtain a list of relevant blocks.

(a) (b) (c)

(d) (e) (f)

FIGURE 12.5 (See color insert.)

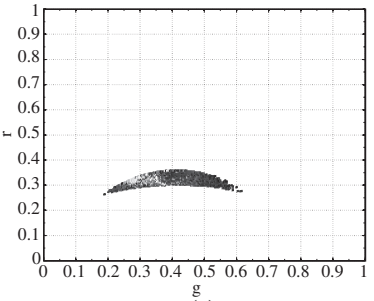
Skin detection examples on RGB images: (a,d) original images compressed in JPEG format, (b,e) simplest threshold method output, and (c,f) probabilistic threshold output.

12.3.2 Feature Extraction: Skin Detection

As explained before, a *visibility image* obtained by forcing the mean gray level of the luminance channel to be about 128 is built. Most existing methods for skin color detection usually threshold some sort of measure of the likelihood of skin colors for each pixel and treat them independently. Human skin colors form a special category of colors, distinctive from the colors of most other natural objects. It has been found that human skin colors are clustered in various color spaces [7], [8]. The skin color variations between people are mostly due to intensity differences. These variations can therefore be reduced by using

(a)

(b)



(c)

FIGURE 12.6

Skin detection examples on Bayer pattern image: (a) original image in Bayer data; (b) recognized skin with probabilistic approach; and (c) threshold skin values on $r-g$ bidirectional histogram (skin locus).

chrominance components only. Yang et al. [9] have demonstrated that the distribution of human skin colors can be represented by a two-dimensional (2D) Gaussian function on the chrominance plane. The center of this distribution is determined by the mean vector $\vec{\mu}$ and its shape is determined by the covariance matrix Σ; both values can be estimated from an appropriate training data set. The conditional probability $p(\vec{x}|s)$ of a block belonging to the skin color class, given its chrominance vector $\vec{x}$, is then represented by:

$$p(\vec{x}|s) = \frac{1}{2\pi}|\Sigma|^{-\frac{1}{2}} \exp\left(\frac{-[d(\vec{x})]^2}{2}\right) \tag{12.7}$$

where $d(\vec{x})$ is the Mahalanobis distance between $\vec{x}$ and $\vec{\mu}$:

$$[d(\vec{x})]^2 = (\vec{x}-\vec{\mu})'\Sigma^{-1}(\vec{x}-\vec{\mu}) \tag{12.8}$$

The value $d(\vec{x})$ determines the probability that a given block belongs to the skin color class. This probability thus reduces with the increase of the distance $d(\vec{x})$. Such class has been experimentally derived using a large dataset of images acquired at different conditions and resolution using CMOS-VGA sensor on *STV6500-E01* Evaluation Kit equipped with *502 VGA sensor* [10]. Due to the large quantity of color spaces, distance measures and 2D distributions, many skin detection algorithms can be used. The skin color algorithm is independent from exposure correction; thus we introduce two alternative techniques aimed to recognize skin regions (as shown in Figure 12.5):

1. By using the input YCbCr image and $p(\vec{x}|s)$ from Equation 12.7, each pixel is classified as a skin or nonskin pixel. Then a new image with normalized grayscale values is derived, where skin areas are properly highlighted (Figure 12.5c). The higher the gray value the bigger the probability to compute a reliable identification.

2. By processing an input RGB image, a 2D chrominance distribution histogram is computed via $r = R/(R+G+B)$ and $g = G/(R+G+B)$. Chrominance values representing skin are clustered in a specific area of the normalized r and g planes, called *skin locus* (Figure 12.6c), as defined in Reference [11]. Pixels having a chrominance value belonging to the skin locus will be selected to correct exposure.

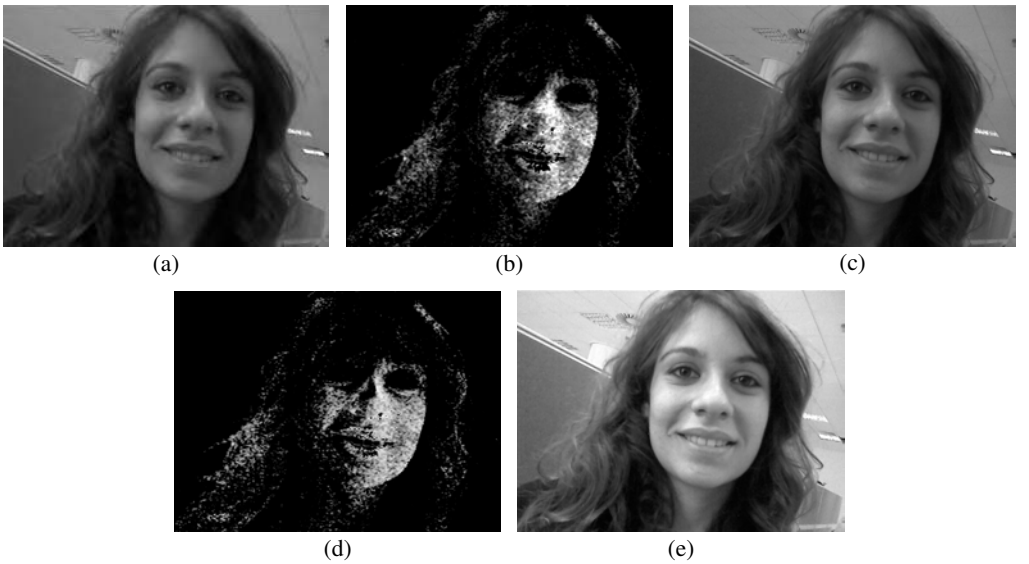


FIGURE 12.7 (See color insert.)

Exposure correction results by real-time and post processing: (a) Bayer image, (b) skin detected in the Bayer image in real-time, (c) RGB color-interpolated image from Bayer data, (d) skin detected in RGB data using postprocessing, and (e) exposure-corrected image obtained from RGB image.

For Bayer data, the skin detection algorithm works on the RGB image created by subsampling the original picture, as described in Figure 12.3.

12.3.3 Exposure Correction

Once the visually relevant regions are identified, the exposure correction (Figure 12.7) is carried out using the mean gray value of those regions as reference point. A simulated camera response curve is used for this purpose. This function can be expressed by using a simple parametric closed form representation:

$$f(q) = \frac{255}{(1 + e^{-Aq})^C} \tag{12.9}$$

where q represents the light quantity and the final pixel value is obtained by means of the parameters A and C used to control the shape of the curve. The q term is supposed to be expressed in 2-based logarithmic unit (usually referred as *stops*). These parameters could be estimated, depending on the specific image acquisition device or chosen experimentally, as better specified below (see Section 12.4). The offset from the ideal exposure is computed using the f curve and the average gray level of visually relevant regions avg, as:

$$\Delta = f^{-1}(Trg) - f^{-1}(avg) \tag{12.10}$$

where Trg is the desired target gray level. The value of Trg should be around 128 but it could be slightly changed especially when dealing with Bayer pattern data where some postprocessing is often applied.

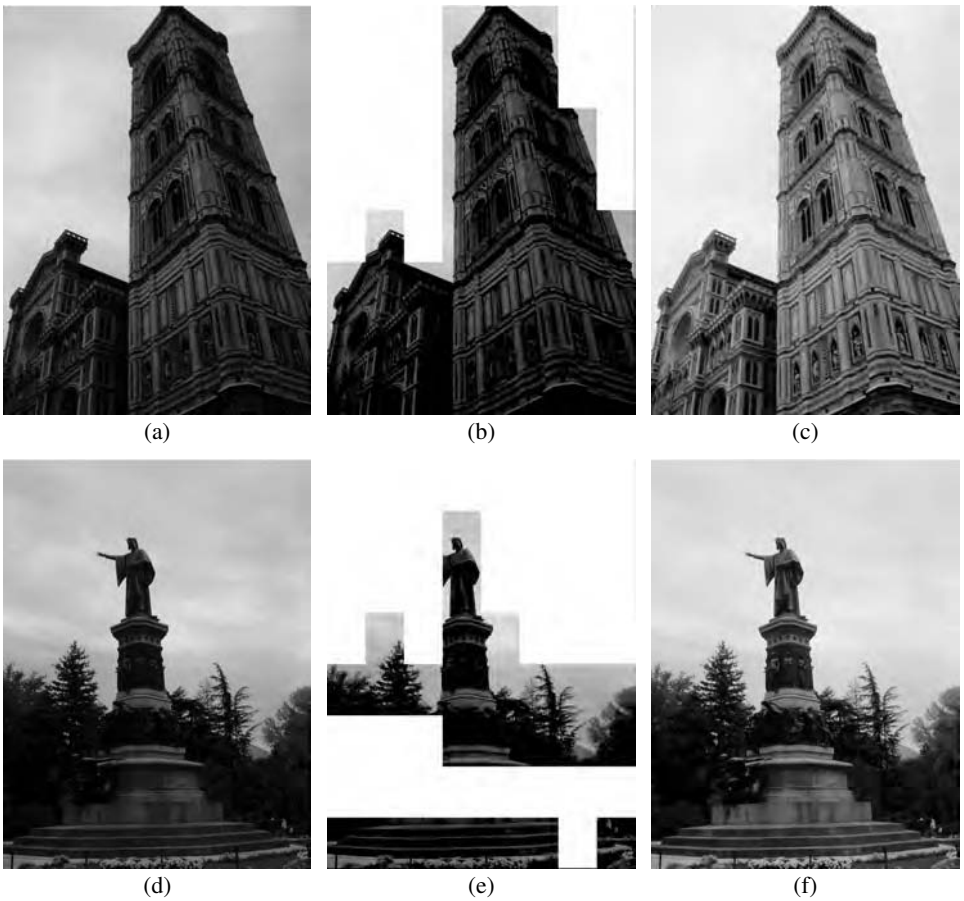


FIGURE 12.8 (See color insert.)

Exposure correction results by postprocessing: (a,d) original color input images, (b,e) contrast and focus visually significant blocks detected, and (c,f) exposure-corrected images obtained from RGB images.

The luminance value $Y(x,y)$ of a pixel (x,y) is modified as follows:

$$Y'(x,y) = f\left(f^{-1}(Y(x,y)) + \Delta\right) \tag{12.11}$$

Basically, this step is often implemented as a lookup table (LUT) transform. This concludes the correction process; all pixels are now corrected.

12.3.4 Exposure Correction Results

The described technique has been tested using a large database of images acquired at different resolutions, with different acquisition devices, both in Bayer and RGB format. In the Bayer case, the algorithm was inserted in a real-time framework, using a CMOS-VGA sensor on *STV6500-E01* Evaluation Kit equipped with *502 VGA sensor* [10]. Examples of skin detection by using real time processing are reported in Figure 12.7. In the RGB case the algorithm could be implemented as postprocessing step. Examples of skin and contrast/focus exposure correction are respectively shown in Figure 12.8 and Figure 12.9.

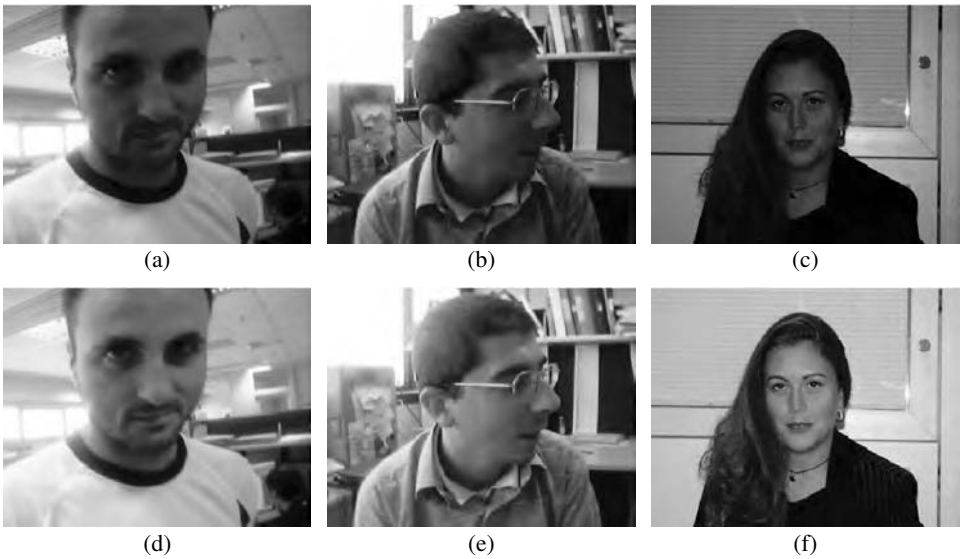


FIGURE 12.9 (See color insert.)

Exposure correction results: (a-c) original images, (d-f) corrected images. Images in (a) and (b) were acquired by Nokia 7650 VGA sensor and compressed in JPEG format whereas the image in (c) was acquired with Olympus E-10 camera equipped with a 4.1 Mega-pixel CCD sensor.

Results show how the features analysis capability of the proposed algorithm permits contrast enhancement taking into account some strong peculiarity of the input image. Major details and experiments can be found in Reference [2].

12.4 Bracketing and Advanced Applications

In order to attempt to recover or enhance a badly exposed image, even if some kind of postprocessing is available, there are situations where this strategy is not possible or leads to poor results. The problem comes from the fact that badly captured data can be enhanced, but if no data exists at all there is nothing to enhance. Today, almost all digital photo-cameras still deal with limited dynamic range and inadequate data representation, which make critical lighting situations, and the real world has tons of them, difficult to handle. This occurs despite the great advancements realized by digital photography, which has made available large resolution even for mass market oriented products. Therefore, multiple exposure capture stands as a useful alternative to overpass actual technology limits. Even if the idea of combining multiple exposed data is just recently receiving great attention, the methodology itself is very old. In the early 1960s, well before the advent of digital image processing Charles Wyckoff [12] was able to capture high dynamic range images by using photographic emulsion layers of different sensitivity to light. The information coming from each layer was printed on paper using different colors, thus obtaining a pseudo-color image depiction.

TABLE 12.1
Typical world luminance levels.

Scene	*Illumination*
Starlight	$10^{-3} cd/m^2$
Moonlight	$10^{-1} cd/m^2$
Indoor light	$10^{2} cd/m^2$
Sunlight	$10^{5} cd/m^2$

12.4.1 The Sensor Versus the World

Dynamic range refers to the ratio of the highest and lowest sensed level of light. For example, a scene where the quantity of light ranges from 1000 cd/m^2 to 0.01 cd/m^2 has a dynamic range of 1000/0.01=100,000. The simultaneous presence in real world scenes poses great challenges on image capturing devices, where usually the available dynamic range is not capable of coping with that coming from the outside world. High dynamic range scenes are not uncommon; imagine a room with a sunlit window, environments presenting opaque and specular objects and so on.

Table 12.1 shows typical luminance values for different scenes, spanning a very wide range from starlight to sunlight. On the other side dynamic range (*DR*) of a digital still camera (DSC) is defined as the ratio between the maximum charge that the sensor can collect (*full well capacity, FWC*), and the minimum charge that is just above sensor noise (*noise floor, NF*). This quantity is usually expressed in logarithmic units:

$$DR = log_{10}\left(\frac{FWC}{NF}\right) \tag{12.12}$$

The dynamic range, which is seldom in the same order of magnitude of those coming from real world scenes, is further affected by errors coming from analog to digital conversion (*ADC*) of sensed light values. Once the light values are captured, they are properly quantized to produce digital codes, that for common eight-bit data fall in the $[0:255]$ range. This means that a sampled, coarse representation of the continuously varying light values is produced.

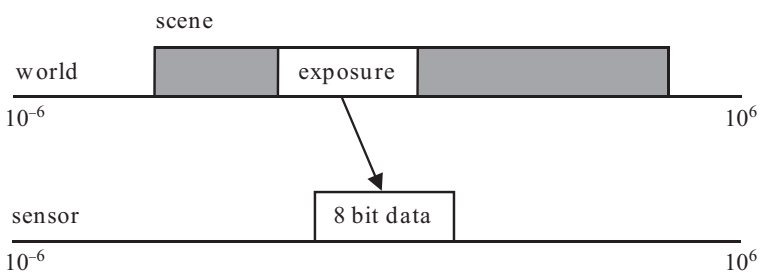


FIGURE 12.10
Due to limited camera dynamic range, only a portion, depending on exposure settings, of the scene can be captured and digitized.

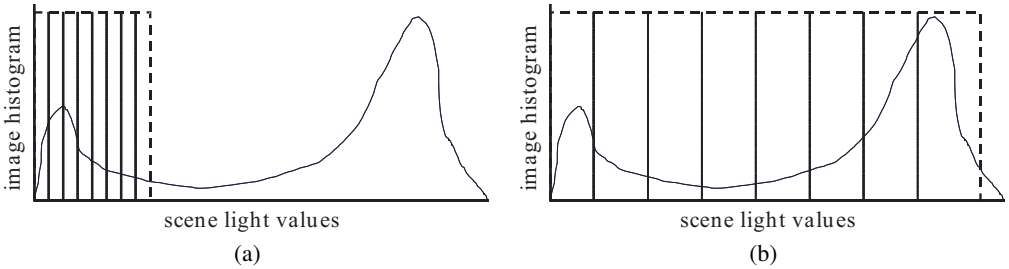


FIGURE 12.11

Information loss for: (a) high exposure, and (b) low exposure. For simplicity, only eight quantization levels, shown with dotted lines, are considered.

Limited dynamic range and quantization thus leads to loss of information and to inadequate data representation. This process is synthetically shown in Figure 12.10, where the dynamic range of a scene is converted to the digital data of a *DSC*. Note that only part of the original range is captured; the remaining part is lost. The portion of the dynamic range where the loss occurs depends on employed exposure settings. Low exposure settings, by preventing information loss due to saturation of highlights, allow to capture highlight values, but lower values will be easily overridden by sensor noise. On the other side, high exposure settings allow a good representation of low light values, but the higher portion of the scene will be saturated. Once again a graphical representation gives a good explanation of the different scenarios.

Figure 12.11a shows a high exposure capture. Only the portion of the scene under the green area is sensed with a very fine quantization; the other portion of the scene is lost due to saturation which happens at the luminance level corresponding to the end of the green area. Figure 12.11b shows a low exposure capture. This time saturation, which happens at the light level corresponding to the end of the red area, is less severe due to low exposure settings and the complete scene is captured (the red area). Unfortunately, due to very widely spanned sampling intervals, quality of captured data is damaged by quantization noise and errors.

In summary, data captured by different exposure settings allows to cover a wider range, and reveals more detail than would have been possible by a single shot. The process is usually conveyed by different steps: i) camera response function estimation, ii) high dynamic range construction, and iii) tone mapping to display or print medium.

12.4.2 Camera Response Function

In order to properly compose a high dynamic range image, using information coming from multiple low dynamic range (*LDR*) images, the camera response function must be known. This function describes the way a camera reacts to changes in exposures, thus providing digital measurements.

Camera exposure X, which is the quantity of light accumulated by the sensor in a given time, can be defined as:

$$X = It \tag{12.13}$$

where I is the irradiance and t is the integration time.

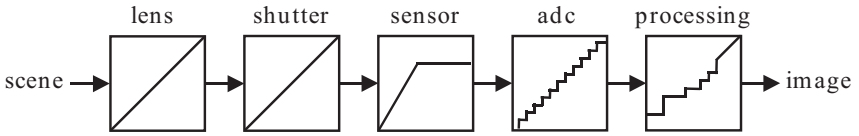


FIGURE 12.12

The full pipeline from scene to final digital image. The main problem behind assembling the high dynamic range from multiple exposures lies in recovering the function synthesizing the full process.

When a pixel value Z is produced, it is known that it comes from some scene radiance I sensed for a given time t, mapped into the digital domain through some function f. Even if most CCD and CMOS sensors are designed to produce electric charges that are strictly proportional to the incoming amount of light (up to the near saturation point, where values are likely to fluctuate), the final mapping is seldom linear. Nonlinearities can come from the *ADC* stage, sensor noise, gamma mapping and specific processing introduced by the manufacturer. In fact often DSC cameras have a built-in nonlinear mapping to mimic a film-like response, which usually produces more appealing images when inspected on low-dynamic displays.

The full pipeline, from the scene to the final pixel values, is shown in Figure 12.12 where prominent nonlinearities can be introduced in the final, generally unknown, processing.

The most obvious solution to estimate the camera response function is to use a picture of uniformly lit different patches, such as the Macbeth Chart [13], and establish the relationship between known light values and recorded digital pixel codes. However this process requires expensive and controlled environment and equipment. This is why several chartless techniques have been investigated. One of the most flexible algorithms has been described in Reference [14], which only requires an estimation of exposure ratios between the input images. Of course, exposure ratios are at hand given the exposure times, as produced by almost all photo-cameras. Given N digitized *LDR* pictures representing the same scene and acquired with timings t_j, for $j = 1,..,N$, exposure ratios $R_{j,j+1}$ can be easily described as

$$R_{j,j+1} = \frac{t_j}{t_{j+1}} \tag{12.14}$$

Thus, the following equation relates the ith pixel of the jth image, $Z_{i,j}$, to the underlying unknown radiance value I_i:

$$Z_{i,j} = f(I_i t_j) \tag{12.15}$$

which is the aforementioned camera response function. The principle of high dynamic range composing is the estimation for each pixel, of the radiance values behind it, in order to obtain a better and more faithful description of the scene that has originated the images. This means that we are interested in finding the inverse of Equation 12.14; a mapping from pixel value to radiance value is needed:

$$g(Z_{i,j}) = f^{-1}(Z_{i,j}) = I_i t_j \tag{12.16}$$

The nature of the function $g(\cdot)$ in the above equation is unknown; the only assumption is that it must be monotonically increasing. That is why a polynomial function of order K is

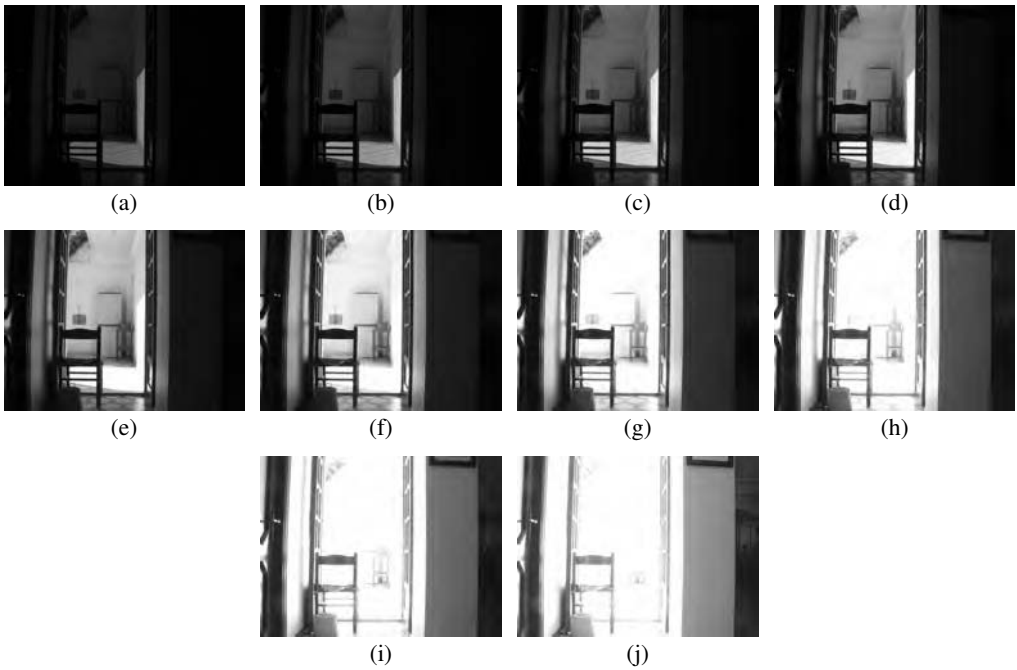


FIGURE 12.13

A sequence of ten images, captured at ISO 50, f-6.3, and exposures ranging from $1/1600$ to $1/4$ seconds: (a) $1/1600$s, (b) $1/800$s, (c) $1/400$s, (d) $1/200$s, (e) $1/100$s, (f) $1/50$s, (g) $1/25$s, (h) $1/13$s, (i) $1/8$s, and (j) $1/4$s.

used:

$$Ie = g(Z) = \sum_{k=0}^{K} c_k Z^k \tag{12.17}$$

The problem thus becomes the estimation of the order K and the coefficients c_k appearing in Equation 12.17. If the ratios between successive image pairs $(j, j+1)$ are known, the following relation holds:

$$\frac{I_i t_j}{I_i t_{j+1}} = \frac{g(Z_{i,j})}{g(Z_{i,j+1})} = R_{j,j+1} \tag{12.18}$$

Using Equation 12.18, parameters are estimated by minimizing the following objective function:

$$O = \sum_{j=1}^{N} \sum_{i=1}^{P} \left(\sum_{k=0}^{K} c_k Z_{i,j}^k - R_{j,j+1} \sum_{k=0}^{K} c_k Z_{i,j+1}^k \right)^2 \tag{12.19}$$

where N is the number of images and P the number of pixels. The system can be easily solved by using the least squares method. The condition $g(1) = 1$ is enforced to fix the scale of the solution, and different K orders are tested. The K value that better minimizes the system is retained.

To limit the number of equations to be considered, not all pixels of the images should be used and some kind of selection is advised by respecting the following rules: i) pixels should be well spatially distributed, ii) pixels should sample the input range, and iii) pixels should be picked from low variance (homogenous) areas.

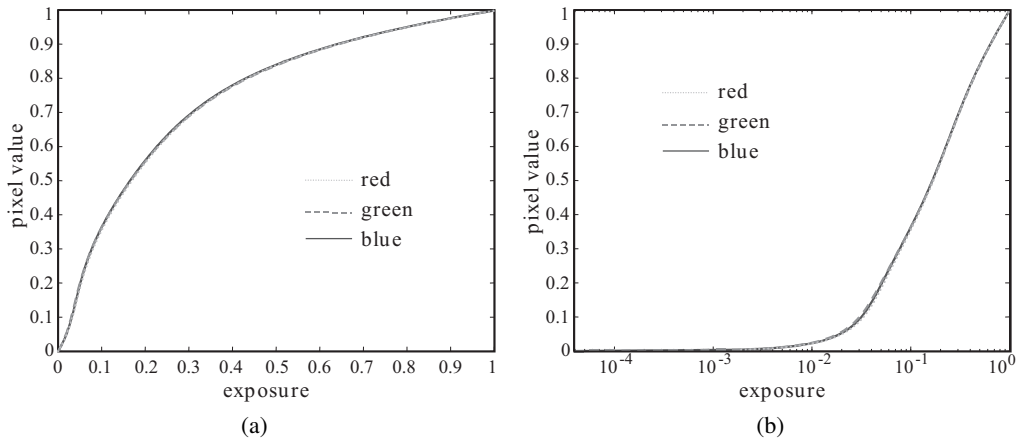


FIGURE 12.14

Camera response functions derived from images depicted in Figure 12.13: (a) response curve on linear scale, and (b) response curve on logarithmic scale.

A different approach for feeding the linear system in Equation 12.19 could be done by replacing pixel value correspondences by *comparagram* pairs. *Comparagrams* have been well described in Reference [15] and provide an easy way to represent how pixels of one image are mapped to the same image with different exposure. This mapping is usually called brightness transfer function (*BTF*).

It is worth noting that if direct access to raw data is available, and known to be linear, the response curve estimation step could be avoided in this case since the function equals a simple straight line normalized in the range $[0,..,1]$. Figure 12.13 shows 10 images captured at different exposure settings, from $\frac{1}{1600}$ *sec* to $\frac{1}{4}$ *sec*, while Figure 12.14 shows the recovered response curve on both linear (left) and logarithmic units.

12.4.3 High Dynamic Range Image Construction

Once the response function (estimated or a priori known) is at hand, the high dynamic range image, usually referred to as *radiance map* and composed of floating point values having greater range and tonal resolution than usual *low dynamic range* (*LDR*) data, can be assembled. The principle is that each pixel in each image provides a more or less accurate estimation of the radiance value of the scene in the specific position. For example, very low pixel values coming from low exposure images are usually noisy, and thus not reliable, but the same pixels are likely to be well exposed in images acquired with higher exposure settings.

Given N images, with exposure ratios $e_i : i = 1 : N$ and considering Equation 12.16 the sequence $\{g(Z_{i,1})/t_1, g(Z_{i,2})/t_2, ..., g(Z_{i,N})/t_N\}$ of estimates for a pixel in position i is obtained. Different estimates should be assembled by means of a weighted average taking into account reliability of the pixel itself. Of course, the weight should completely discard pixels that appear as saturated and assign very low weight to pixels whose value is below some noise floor, since they are unable to provide decent estimation.

One possible weighting function could be a hat or Gaussian shaped function centered around mid-gray pixel values, which are far from noise and saturation. As a rule of thumb, for each pixel there should be at least one image providing a useful pixel (e.g., that is not saturated, nor excessively noisy). Given the weighting function $w(Z)$ the radiance estimate for a given position i is calculated as follows:

$$I_i = \frac{\sum_{j=1}^{N} w(Z_{i,j}) \frac{g(Z_{i,j})}{t_j}}{\sum_{j=1}^{N} w(Z_{i,j})} \tag{12.20}$$

12.4.4 The Scene Versus the Display Medium

Once the high dynamic range image has been assembled, what is usually required is a final rendering on the display medium, such as a CRT display or a printer. The human eye is capable of seeing a huge range of luminance intensities, thanks to its capability to adapt to different values. Unfortunately, this is not the way most image rendering systems work. Hence they are usually not able to deal with the full dynamic range contained in images that provide an approximation of real world scenes. Most CRT displays have a useful dynamic range in the order of nearly 1:100. It is certain that in the near future, high dynamic reproduction devices will be available, but for the moment they are far from mass market consumers. Simply stated, *tone mapping* is the problem of converting an image containing a large range of numbers, usually expressed in floating point precision, into a meaningful number of discrete gray levels (usually in the range $0, ..., 255$), that can be used by any imaging device. So, we can formulate the topic as that of the following quantization problem:

$$Q(val) = |(N-1)F(val) + 0.5|; \quad \text{with } F : [L_{w_{min}} : L_{w_{max}}] \rightarrow [0:1] \tag{12.21}$$

where $[L_{w_{min}} : L_{w_{max}}]$ is the input range, N the number of allowed quantization levels, and F the tone mapping function. A simple linear scaling usually leads to the loss of a high amount of information on the reproduced image. Figure 12.15a shows the result obtained by linearly scaling a high dynamic range image, constructed from the sequence in Figure 12.13 using the techniques described above. As can be seen, only a portion of the scene is visible, so better alternatives for F are needed.

Two different categories of tone mapping exist:

- *tone reproduction curve (TRC)* where the same function is applied for all pixels; and
- *tone reproduction operator (TRO)* where the function acts differently depending on the value of a specific pixel and its neighbors.

In what follows, several techniques — such as histogram adjustment from the TRC category and Chiu's local operator, bilateral filtering, photographic tone reproduction, and gradient compression, all from the TRO category — will be briefly described and applied on the input HDR image, assembled from the sequence in Figure 12.13. The recorded input was in the range of $0.00011 : 32$.

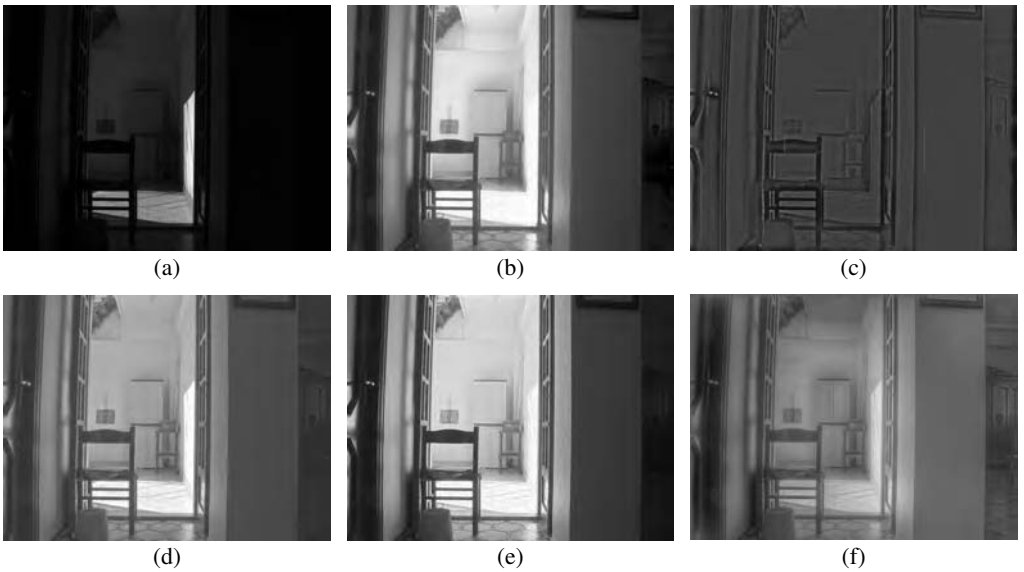


FIGURE 12.15 (See color insert.)

(a) HDR image built from the sequences of images shown in Figure 12.13 using linear scaling in the $[0,...,1]$ range and quantization to 8 bits, (b) image obtained using histogram adjustment mapping, (c) image mapped using Chiu's algorithm with some halo artifacts highlighted, (d) image mapped using bilateral filtering, (e) photographic tone reproduction mapping, and (f) gradient compression mapping.

12.4.4.1 Histogram Adjustment

The algorithm described in Reference [16] is based on ideas coming from image enhancement techniques, specifically histogram equalization. While histogram equalization is usually employed to expand contrast images, in this case it is adapted to map the high dynamic range of the input image within that of the display medium, while preserving the sensation of contrast. The process starts by computing a downsampled version of the image, with a resolution that equals to 1 degree of visual angle. Luminance values of this so-called *fovea* image are then converted in the *brightness* domain, which can be approximated by computing logarithmic values. For the logarithm-valued image, a histogram is built where values between minimum and maximum bounds $L_{w_{min}}$ and $L_{w_{max}}$ (of the input radiance map) are equally distributed on the logarithmic scale. Usually employing around 100 histogram bins each having a size of $\Delta b = \frac{\log(L_{wmax}) - \log()L_{w_{min}})}{100}$ provides sufficient resolution. The cumulative distribution function, normalized by the total number of pixels T, is defined as:

$$P(b) = \sum_{b_i<b} f(b_i)/T; \quad \text{with } T = \sum_{b_i} f(b_i) \tag{12.22}$$

where $f(b_i)$ is the frequency count for bin i. The derivative of this function can be expressed as follows:

$$\frac{\partial P(b)}{\partial b} = \frac{f(b)}{T\Delta b} \tag{12.23}$$

Applying a histogram equalization on the input, the result is an image where all brightness values have equal probability. The equalization formula, which provides a way to map

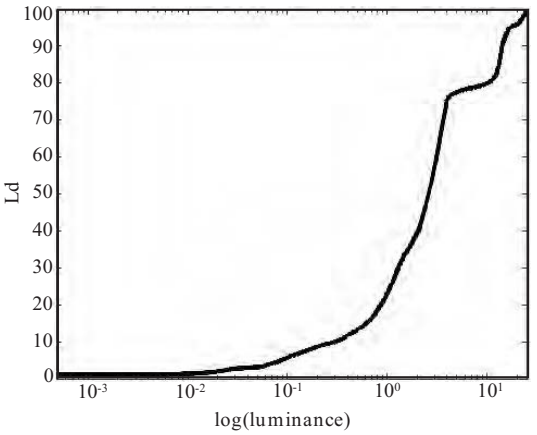


FIGURE 12.16

Histogram adjustment mapping function.

luminance values to display values, can be expressed as:

$$\log\left(L_d(x,y)\right) = \log\left(L_{d_{min}}\right) + \left(\log\left(L_{d_{max}}\right) - \log\left(L_{d_{min}}\right)\right) \cdot P\left(\log(L_w(x,y))\right) \tag{12.24}$$

where $L_{d_{min}}$ and $L_{d_{max}}$ stay for minimum and maximum display values. This means that the equalized brightness is fit into the available display's dynamic range. Unfortunately naive equalizations tend to over-exaggerate contrast in correspondence of highly populated bins (histogram peaks), leading to undesirable effects. To prevent this, a ceiling procedure is applied on the histogram, imposing that contrast should never exceed those obtained by a linear mapping. The ceiling can be written in terms of the derivative of the mapping (which is indicative of contrast):

$$\frac{\partial L_d}{\partial L_w} \leq \frac{L_d}{L_w} \tag{12.25}$$

By combining Equations 12.23 and 12.24 the final histogram constraint is obtained:

$$f\left(b\right) \leq \frac{T\Delta b}{\log\left(L_{d_{max}}\right) - \log\left(L_{d_{in}}\right)} \tag{12.26}$$

Thus, in order to prevent excessive contrast, histogram values are repetitively cut to satisfy Equation 12.26. Using some features of the human visual system (HVS), the operator has been further refined by the authors to include more sophisticated ceiling procedure, simulation of color and contrast sensivity. Figure 12.15b shows a sample radiance map, tonemapped to display using Ward's operator in its basic implementation. Figure 12.16 plots the resultant mapping.

12.4.4.2 Chiu's Local Operator

One of the first works performing spatially varying local processing, for visualization of high dynamic range image data, was described in Reference [17] whose importance is mainly due to historical reasons. The idea is to apply at each pixel position a specific

scaling, $s(x,y)$. The output image, to be rendered on the medium, is then obtained by multiplying the input by the scaling function:

$$L_d(x,y) = s(x,y) L_w(x,y) \tag{12.27}$$

The scaling function $s(x,y)$ is defined as

$$s(x,y) = \frac{1}{L_{blur}(x,y)k} L_w(x,y) \tag{12.28}$$

where $L_{blur}(x,y)$ is a Gaussian filtered version of $L_w(x,y)$ and k a user parameter. A value of $k = 2$ brings values near the local average (given by low-pass filtered data) to a value of 0.5. This does not leave too much room for the mapping of brighter pixels, hence a value of $k = 8$ is suggested. The main problem with this technique, and which is the main concern of all local operators, is the appearance of so-called *halo artifacts* which easily manifest themselves around areas of relevant brightness transition. This is due to the fact that the local average for pixels around transition zones carries unwanted information which is not related to the luminance value of the specific pixel. For example, for a bright pixel near a very dark region, the local average $L_{blur}(x,y)$ will be very low due to the influence of dark pixels, leading to a poor scaling of the bright pixel and thus to the appearance of a *bright halo*. On the other side, a *dark halo* is likely to appear for dark pixels near bright regions, where the scaling will be excessive. Figure 12.15c shows the result of Chiu's algorithm with $k = 8$ and Gaussian filtering with a pixel width of 15 where some of the halo artifacts are highlighted by red squares.

12.4.4.3 Bilateral Filtering

Durand et al. [18] consider the input image as separable into two different layers: a *base layer* and a *detail layer*. The first is related to the low frequency content of the image and the second to the high frequency content. Thus an input image $L_w(x,y)$ can be expressed as the multiplication of its two layers: $L_w(x,y) = Base(x,y) \cdot Detail(x,y)$. In order to properly scale the high dynamic range data, the base layer is fed to a compressive function, while the detail layer is leaved unchanged. This helps the preservation of subtle local contrast, and is also related to the concept that the base layer represents the influence of lighting conditions (and thus the scene dynamic range). All the processing is done in the logarithmic domain, where the two layers are separated, processed and recombined. The basic steps of the algorithm are:

1. Express the data in the logarithmic domain $l(x,y) = \log(L_w(x,y))$.
2. Compute the base layer $Base(x,y)$.
3. Compute the detail layer $Detail(x,y) = Base(x,y) - l(x,y)$.
4. Compress the base layer obtaining $comp(Base(x,y))$.
5. Recombine the layers and exponentiate the result, to produce the final image, $L_d(x,y) = exp(comp(Base(x,y)) + Detail(x,y))$.

The compression of the base layer is simply done by scaling it by a multiplicative factor m, such that its range equals a desired contrast c:

$$\begin{gathered} c = m\,(max(Base(x,y)) - min(Base(x,y))) \\ \text{with } comp(Base(x,y)) = mBase(x,y) \end{gathered} \tag{12.29}$$

The most relevant feature of the algorithm is the way in which the base layer is computed, which should be a low-pass filtered version of the image, but without the unwanted issues of the Gaussian filtering employed by the aforementioned Chiu's algorithm. In other words, the low-pass filtering process should not take into consideration each pixel, in particular those corresponding to luminance values that are very different from the luminance of the actual pixel. To achieve this, a *bilateral filter* is considered. Bilateral filtering

$$\begin{aligned} bil(x,y) &= \frac{1}{k(x,y)} \sum_{(u,v)\in\Omega} g(x-u,y-v)l(u,v)w(d(l(u,v),l(x,y))),d(u,v) \\ &= |l(u,v) - l(x,y)| \end{aligned} \tag{12.30}$$

is obtained by adding the usual Gaussian filter with a spatial kernel g, and a further Gaussian weighting function w whose weights decrease as the difference in luminance value between the central pixel and its neighbors in a surround Ω increase. In the above equation, $k(x,y)$ is the normalization term. Since bilateral filtering in the spatial domain can be computationally very slow, the authors have developed a very fast approximation in the frequency domain. Figure 12.15d shows the result of the algorithm for $c = \log(50)$.

12.4.4.4 Photographic Tone Reproduction

Reinhard et al. [19] have developed an operator based on some simple photographic principles such as *automatic exposure* and *dodge and burning*, where the first provides a global mapping of the image and the latter exploits some local features. The global part of the operator analyzes the concept of *scene's key*, which is a measure of how overall dark or bright the images is. This quantity is approximated with the log average value $\overline{L}_w$ of the image luminance values. According to photographic principles, where the key is usually printed (or displayed) to have the 18% reflectance of the medium, an initial global mapping is performed using the following equation:

$$L_m\,(x,y) = \frac{0.18}{\overline{L}_w} \cdot L_w\,(x,y) \tag{12.31}$$

In this way, a kind of automatic exposure setting is provided for the scene; it is even done *ex post facto*, since the scene has already been captured by the camera (as radiance maps provide a floating point description of the initial scene, this allow us to do such virtualizations of the photographic process). Even if in Equation 12.31, the scene's key value is linearly mapped to the value of 0.18, different values could be used depending on the specific image content (e.g., a nightlife picture should be scaled to a very low value). No matter what the dynamic range of the initial scene is, the luminance values exposed by means of Equation 12.31 are forced to fit inside the medium dynamic range (which is supposed to vary within $[0,..,1]$) using a compressive function, which is particularly

effective on very high luminance values:

$$L_d(x,y) = \frac{L_m(x,y)}{1+L_m(x,y)} \tag{12.32}$$

This function scales input values differently, according to their magnitude. Namely, small values, usually $\ll 1$ are almost leaved unchanged, while very high values, usually $\gg 1$, are scaled by a very large amount (the quantity $\frac{1}{L_m(x,y)}$).

The scaling function is further refined to include some local processing, similar to *dodge and burning* procedures, where a dark value in a bright surround is heavily compressed (burned), and a bright pixel on a dark surround is only mildly compressed (dodged). To exploit these local properties, filtered versions at different scales $s = 1,..,S$ of L_m are produced as

$$L_{blur_s} = L_m * R_s \tag{12.33}$$

and in Equation 12.32, the quantity $L_m(x,y)$ on the denominator is replaced by

$$L_d(x,y) = \frac{L_m(x,y)}{1+L_{blur_s}(x,y)} \tag{12.34}$$

where $*$ equals to the convolution operator, and R_s are different Gaussian kernels, having different pixel widths $w(s)$, for $s = 1,2,...,S$:

$$w(s) = e\left(\{\log(w_{min}) + \frac{s}{S}(\log(w_{max}) - \log(w_{min}))\right) \tag{12.35}$$

The term w_{max} and w_{min} are the maximum and minimum allowed pixel widths, and are fixed respectively to 1 and 43. Thus the smallest scale for a pixel in position (x,y) equals to the pixel itself. To avoid halo artifacts, for each pixel the largest scale $s_{max} : |V_{smax}(x,y)| < \varepsilon$, where V_s denotes the difference between two successive scales, is computed as follows:

$$V_s(x,y) = \frac{L_{blur_s} - L_{blur_{s+1}}}{2^{\Phi}0.18/s^2 + L_{blur_s}}. \tag{12.36}$$

Note that according to the authors, a value of $\Phi = 8$ should be used. Practically, Equation 12.36 is the search of the largest surround across a pixel in position (x,y) whose value is reasonably similar to that of the pixel. This avoids the appearance of severe halo artifacts, similar to those seen by the application of Chiu's algorithm. Figure 12.15e shows the result of the algorithm of Reinhard et al. on our example radiance map, where the parameters $S = 8, \Phi = 8$ have been used.

12.4.4.5 Gradient Compression

The last technique, belonging to the family of TRO, that we are going to describe was developed by Fattal et al. [20], and it is far more sophisticated than those that have been described before. Even if sometimes output images can have an unnatural appearance, in most cases results can look very appealing. This algorithm does not operate directly on the spatial domain, but instead computes the *gradient field* of the input image and after manipulating it, reconstructs by means of Poisson integration the image having the new

gradients. This derives from the observation that an image exhibiting a high dynamic range will be characterized by gradients of large magnitude around zones of brightness transition. Hence attenuating those gradients seems like a viable way for building a LDR depiction of the scene, suitable for inspection on common displays. Similarly to the pipeline of the algorithm based on bilateral filtering, gradient compression works on logarithmic data, and so just before producing the output image, the result undergoes exponentiation. Indicating with $l(x,y)$ the data in the logarithmic domain, the gradient field $\nabla l(x,y)$ is computed as follows:

$$\nabla l(x,y) = \big(l(x+1,y) - l(x,y), l(x,y+1) - l(x,y)\big) \tag{12.37}$$

Attenuation of the gradient field is obtained by multiplication with a proper scaling function $\Phi(x,y)$:

$$G(x,y) = \nabla l(x,y)\Phi(x,y) \tag{12.38}$$

The attenuated gradient field $G(x,y)$ is then inverted by solving the Poisson equation

$$\nabla^2 \tilde{l}(x,y) = div\ G(x,y) \tag{12.39}$$

Since edges (and thus gradients) exist at multiple resolution levels, a Gaussian pyramid representation $< l_0, l_1, ..., l_S >$ is constructed, and for each level the gradient field is computed. The attenuation function is then computed on each level and reported to the upper level in *bottom to top* fashion. The attenuation function at the top level is the one that can be effectively used in Equation 12.38. Attenuation function at each level s is computed as follows:

$$\Psi_s(x,y) = \frac{\alpha}{||\nabla l_s(x,y)||} \left(\frac{||\nabla l_s(x,y)||}{\alpha} \right)^{\beta} \tag{12.40}$$

The α parameter in Equation 12.40 determines which gradient magnitudes are left untouched, while the β exponent amplifies magnitudes greater than α. Suggested values are $\beta = 0.9$ and α equal to average gradient magnitude multiplied by 0.1. Since the attenuation function is computed for each resolution level s, the propagation to full resolution is done by scaling the attenuation function from level $s-1$ to s, and accumulating the values to obtain the full resolution attenuation function $\Phi(x,y)$ that will be effectively used (authors claim that by using the attenuation function just at full resolution halo artifacts are mostly invisible). This can be expressed by the following equations:

$$\Phi_d(x,y) = \Psi_d(x,y) \tag{12.41}$$

$$\Phi_k(x,y) = L(\Phi_{k+1})(x,y) \cdot \Psi_d(x,y) \tag{12.42}$$

$$\Phi(x,y) = \Phi_0(x,y) \tag{12.43}$$

where d is the smallest resolution level and L is the bilinear up-sampling operator. Figure 12.15f shows the result of applying the gradient compression operator on our sample HDR image. The operator looks computationally more complicated than others that have been described but as can be seen, the mapped image looks far more impressive in terms of high-light and low-light visibility than the previous renderings.

12.5 Conclusion

The problem of the proper exposure settings for image acquisition is of course strictly related with the dynamic range of the real scene. In many cases, some useful insights can be achieved by implementing ad-hoc metering strategies. Alternatively, it is possible to apply some tone correction methods that enhance the overall contrast of the most salient regions of the picture. The limited dynamics range of the imaging sensors does not allow to recover the dynamic of the real world; in that case only by using "bracketing" and acquiring several pictures of the same scene with different exposure timing a final good rendering can be realized.

In this chapter, we have presented a review of automatic digital exposure correction methods and reported the specific peculiarities of each solution. Just for completeness, we report that recently, Raskar et al. [21] have proposed a novel strategy devoted to "flutter" the camera's shutter open and closed during the chosen exposure time with a binary pseudo-random sequence. In this way high-frequency spatial details can be recovered especially when movements with constant speed are present. In particular a robust deconvolution process is achieved just considering the so-called coded-exposure that makes the problem well-posed. We think that Raskar's technique could also be used in multi-picture acquisition just to limit the overall number of images needed to reconstruct a reliable HDR map.

References

[1] M. Mancuso and S. Battiato, "An introduction to the digital still camera technology," *ST Journal of System Research*, vol. 2, no. 2, pp. 1–9, January 2001.

[2] S. Battiato, A. Bosco, A. Castorina, and G. Messina, "Automatic image enhancement by content dependent exposure correction," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 12, pp. 1849–1860, September 2004.

[3] S. Battiato, A. Castorina, and M. Mancuso, "High dynamic range imaging for digital still camera: An overview," *Journal of Electronic Imaging*, vol. 12, no. 3, pp. 459–469, July 2003.

[4] R. Gonzalez and R. Woods, *Digital Image Processing*. Addison-Wesley Longman Publishing, 2nd edition, Boston, MA, USA, 1992.

[5] S. Bhukhanwala and T. Ramabadran, "Automated global enhancement of digitized photographs," *IEEE Transactions on Consumer Electronics*, vol. 40, no. 1, pp. 1–10, February 1994.

[6] B.E. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 1976.

[7] S. Phung, A. Bouzerdoum, and D. Chai, "A novel skin color model in YCBCR color space and its application to human face detection," in *Proceedings of the IEEE International Conference on Image Processing*, Rochester, NY, USA, September 2002, vol. 1, pp. 289–292.

[8] B. Zarit, B. Super, and F. Quek, "Comparison of five colour models in skin pixel classification," in *Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, Corfu, Greece, September 1999, pp. 58–63.

[9] J. Yang, W. Lu, and A. Waibel, "Skin-colour modeling and adaptation," Technical Report CMU-CS-97-146 School of Computer Science, Carnegie Mellon University, 1997.

[10] STMicroelectronics, "Colour sensor evaluation kit vv6501." Edinburgh, available online: www.edb.st.com/ products/image/sensors/501/6501evk.htm.

[11] M. Soriano, B. Martinkauppi, and M.L.S. Huovinen, "Skin color modeling under varying illumination conditions using the skin locus for selecting training pixels," in *Proceedings of the International Workshop on Real-time Image Sequence Analysis*, Oulu, Finland, August 2000, pp. 43–49.

[12] C. Wyckoff, "An experimental extended response film," *SPIE Newsletter*, pp. 16–20, June / July 1962.

[13] Y.C. Chang and J.F. Reid, "RGB calibration for analysis in machine vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 10, pp. 1414–1422, October 1996.

[14] T. Mitsunaga and S. Nayar, "Radiometric self calibration," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997, vol. 2, pp. 374–380.

[15] S. Mann, "Comparametric equations with practical applications in quantigraphic image processing," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1389–1406, August 2000.

[16] G.W. Larso, H. Rushmeier, and C. Piatko, "A visibility matching tone reproduction operator for high dynamic range scenes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 4, pp. 291–306, October - December 1997.

[17] B. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang, and K. Zimmerman, "Spatially nonuniform scaling functions for high contrast images," in *Proceedings of the Graphics Interface Conference*, Toronto, ON, Canada, May 1993, pp. 245–253.

[18] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high dynamic range images," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, San Antonio, TX, USA, July 2002, pp. 257–266.

[19] E. Reinhar, M. Stark, P. Shirley, and J. Ferweda, "Photographic tone reproduction for digital images," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, San Antonio, TX, USA, July 2002, pp. 267–276.

[20] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, San Antonio, TX, USA, July 2002, pp. 249–256.

[21] R. Raskar, A. Agrawal, and J. Tumblin, "Coded exposure photography: Motion deblurring using fluttered shutter," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 795–804, July 2006.

13

Digital Camera Image Storage Formats

Kenneth A. Parulski and Robert Reisch

13.1 Introduction

This chapter describes the image formats used by most current digital cameras. These include the Exif (Exchangeable image file) format [1], [2] used to store processed, JPEG (Joint Photographic Experts Group) compressed images in most consumer cameras, and the TIFF (Tagged Image File Format) compatible raw formats [3] used by many DSLR (Digital Single Lens Reflex) cameras. These current formats have evolved over time, adopting some features first introduced in other, less successful, image formats. Thus, a historical perspective can be helpful in understanding the image formats used in today's single sensor color cameras.

The chapter begins with Section 13.2 which provides a high level description of image formats, memory card formats, and image metadata. Following this introduction, a timeline showing the evolution of digital camera image formats is presented in Section 13.3. Section 13.4 describes the structure of the Exif-JPEG image format and Section 13.5 focuses on the metadata included in Exif-JPEG image files. This is followed by Section 13.6 which provides a description of raw image file formats, using the TIFF/EP (Tagged Image File Format for Electronic Photography) format as an example. Next, the DCF (Design rule for Camera File system) directory structure, and the DPOF (Digital Print Order Format) control format, are described in Section 13.7. Finally, a number of recent image formats are reviewed in Section 13.8. The chapter concludes with Section 13.9.

TABLE 13.1
Memory card formats.

Name	*Date*	*Size (mm)*	*Type / Contacts*	*URL*
PC Card	1990	$85.6 \times 54.0 \times 3.3$	Socket / 68	www.pcmcia.org
Compact Flash	1994	$36.0 \times 43.0 \times 3.3$	Socket / 50	www.compactflash.org
Smart Media	1996	$45.0 \times 37.0 \times 0.76$	Surface / 22	www.ssfdc.or.jp
MMC	1997	$32.0 \times 24.0 \times 1.4$	Surface / 7	www.mmca.org
Memory Stick	1998	$21.5 \times 50 \times 2.8$	Surface / 10	www.memorystick.com
SD	1999	$32.0 \times 24.0 \times 1.4$	Surface / 9	www.sdcard.org
xD	2002	$20.0 \times 25.0 \times 1.7$	Surface / 18	www.xd-picture.com

13.2 Image Formats, Memory Formats, and Metadata

The conventional wisdom, when consumer digital cameras were first developed, was that all cameras would need to use the same standard memory card format in order to be successful. This thinking was based on the fact that film photography was a mass-market success as a result of film format standards. The interoperability provided by 35 mm film standards allowed many different companies throughout the world to provide compatible cameras, films, and photofinishing services. These standards ensured that the images captured on 35 mm format film could be developed and printed anywhere in the world. In other words, having standard physical properties and photofinishing processes for the film media were necessary to enable interoperability.

However, consumer digital cameras have used many different memory card formats, as shown in Table 13.1. Over the years, these card formats evolved from credit-card sized PC Cards, to smaller CompactFlash and Smart Media cards, and eventually to Memory Stick, MMC (MultiMedia Card), SD (Secure Digital) and xD cards. Miniature versions of some of the card formats listed in the table, such as miniSD cards, are used in mobile phones that incorporate digital cameras. All these cards, except for Smart Media cards, are still used today, and it is likely that new card formats will be introduced in the coming years.

Nevertheless, consumer digital photography became a mass-market success. This happened because consumer digital cameras adopted the same standard image file format. So the image file and directory formats, not the memory card format, are most analogous to the film format in conventional photography.

Today's consumer digital cameras store images using the industry standard Exif-compressed image format, which uses the JPEG image compression standard [4]. This enables the images from digital cameras to be used by many other devices, such as home computers, appliance printers, retail kiosks, and on-line printing services. The image files can be used by many different software applications, and posted on web pages or emailed so that they can be accessed anywhere in the world. Many different companies provide these digital photography products, software, and services. This provides benefits to both consumers and manufacturers.

But the compatibility provided by the Exif-JPEG format has some drawbacks, too. The format uses baseline JPEG compression, which is limited to storing 24-bit color images,

		
Camera Model	Kodak DX4330 Camera	Kodak V705 Dual Lens Camera
Image Size	1726 × 1150 pixels	3072 × 2304 pixels
Date / Time Original	July 10, 2004 10:59:43 am	April 19, 2007 5:38:33 pm
Lens Focal Length	63 mm (35mm equiv.)	94 mm (35mm equiv.)
Flash	Flash did not fire, auto mode	Flash fired, always flash mode
Exposure Time	1/700 second	1/100 second
F-number	7.0	4.3
Metering Mode	Average	Pattern

FIGURE 13.1

Images and metadata.

using 8 bits per component for the luminance (Y) component, and the two color components, red minus luminance (Cr), and blue minus luminance (Cb). Furthermore, baseline JPEG uses a lossy compression algorithm that can create image artifacts [5]. Professional photography applications often require higher quality. Therefore, most professional cameras, and many advanced amateur cameras, include a *raw* image format setting. The raw file stores digital data that directly relates to the single-sensor color information captured by the camera, normally using a single sample, with 12 bits or 16 bits, per pixel. The color demosaicking and other processing are performed after the image is transferred to a host computer. This allows more sophisticated image processing to be used, and enables the user to adjust various image-processing settings.

Because the raw image file stores the data directly from the sensor, the characteristics of the raw image data, such as the color it encodes and the type of noise it includes, are specific to the type of digital camera that created the file. In order to produce the finished image, the host device must be able to perform the image processing required for the specific camera. In many cases, the companies that make digital cameras or raw image processing software keep the details of their image processing algorithms proprietary, for competitive reasons. Therefore, while the format of the raw data can be standardized, the images provided by standard raw files will vary because of differences in the image-processing algorithms used to transform the raw data into standard color image data.

In addition to the digital values used to encode the image, image files store so-called *metadata* [6]. Metadata is any type of information that relates to the image. Figure 13.1 shows two different captured images and some of the metadata provided by the camera when the images were captured. The metadata specifies the model of the camera that captured the image and the size of the image. The metadata also includes the date and time the image was captured. This is very useful in retrieving specific images captured during a particular period of time, or on a specific date, such as a birthday or holiday. The

metadata also includes camera settings, such as the lens focal length and whether or not the camera flash fired. This metadata can help to automatically locate specific types of images, such as close-up scenes or indoor images. The camera settings also normally include the exposure time and f-number of the lens. A long exposure time means that the image is more likely to include motion blur, and a low f-number means that the image may have a relatively narrow depth-of-field. Other metadata provide camera mode settings, such as the metering mode. All of this metadata is stored using standardized tags within the Exif-JPEG file. Each metadata tag includes a data field which identifies the particular type of metadata stored by the tag, along with data fields that provide the metadata values and define how these values are encoded.

Some digital cameras provide other types of metadata, such as the ambient light level, lens focus distance, and the position of the main subject in the image. The metadata can also indicate the copyright owner, and other types of tags or labels selected by the photographer. In some cameras and camera phones, the metadata includes GPS (Global Positioning System) co-ordinates indicating the location of the camera when the picture was taken. In addition, proprietary metadata can be included in the image file. To remain useful, the metadata must be properly managed when an image is edited.

13.3 History of Image Formats for Digital Cameras

The image formats used in today's digital cameras are built on the TIFF image format and the JPEG compression standard. Figure 13.2 is a timeline showing when various image formats were developed and revised. The JPEG standard, ISO/IEC 10918-1, specifies a baseline compression algorithm using the discrete cosine transform (DCT). The JPEG standard was developed in the late 1980's and published as an approved international standard in 1994. It defines a minimal image file format for the exchange of compressed data, but without color space specifications. Instead, it defines only the metadata needed to decompress the JPEG compressed image data, such as the quantization and Huffman tables. While the JPEG standard was being approved, the JPEG file interchange format (JFIF) [7] was published by Eric Hamilton of C-Cube Microsystems. JFIF specifies a standard image orientation and color encoding. It uses an APP0 *application marker* at the beginning of the JPEG file to store a thumbnail image and metadata indicating the pixel aspect ratio and density. Several years later, the JPEG group developed a complete Still Picture Interchange File Format (SPIFF), which was published in 1997 as part of ISO/IEC 10918-3, JPEG extensions [8]. But this was too late, and SPIFF never achieved significant adoption.

TIFF was developed by Aldus and Microsoft in the 1980's for storing digital images from scanners and computer graphics. It remains one of the most popular and flexible raster file formats. TIFF is now controlled by Adobe, which published the current version 6.0 in 1992 [9]. TIFF defines a very flexible, tag-based file structure for storing image data and metadata. TIFF files can store many different types of image data. Standard TIFF tags are used to indicate, for example, the number of samples per line and the number of bits per sample. TIFF defines some common metadata, such as the make and model of the device

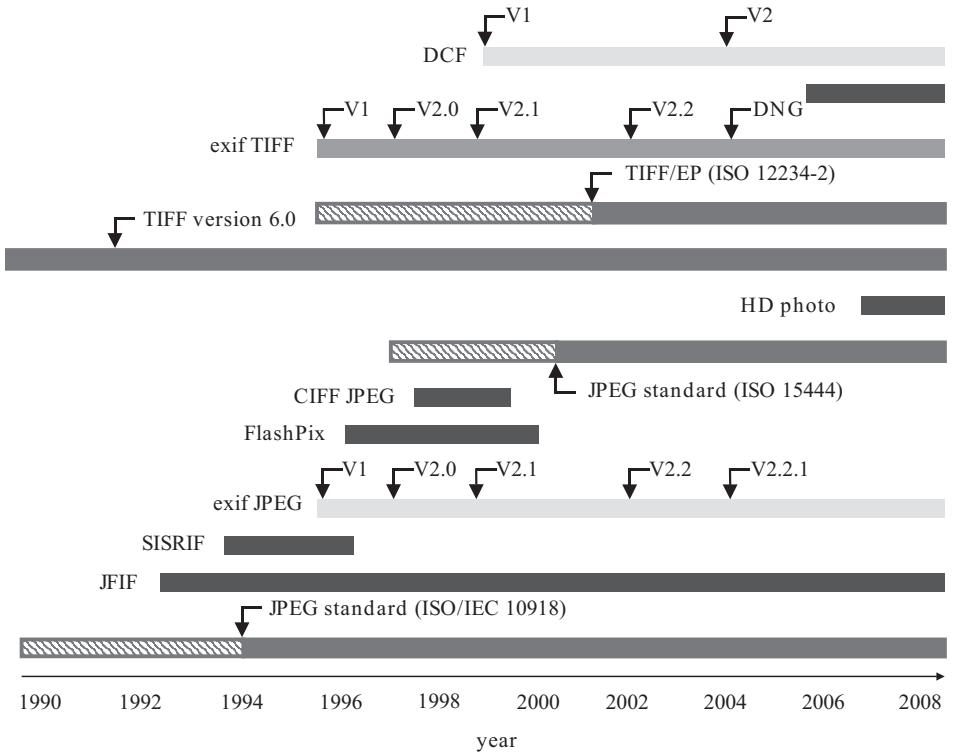


FIGURE 13.2

Image format timeline.

that created the file, and the date and time the file was last modified. TIFF also allows users to register proprietary tags to store new types of metadata. However, baseline TIFF 6.0 implementations are only required to read 8 bits per channel uncompressed image data and can ignore most of the metadata in the file. In 1994, Adobe published a draft TIFF technical note describing how JPEG compressed data should be embedded in TIFF files, which was later supported by their software applications [10]. However, baseline TIFF readers are not required to support these embedded JPEG files, and so even today very few software applications provide this capability.

The first digital cameras used proprietary image formats. In 1992, the ISO technical committee on photography standards, known as technical committee 42 (TC 42), created working group 18 (WG 18) in order to develop standards for electronic still picture imaging. In 1993, the Electronic Still Camera Working Group of the Japan Electronic Industry Development Association (JEIDA) adopted a standard image data format, which could store both uncompressed and baseline JPEG compressed image data on PC Cards. The development of this standard, later called SISRIF (Still Image, Sound and Related Information Format) [11], was led by Motokazu Ohkawa of Toshiba. SISRIF used *tuples* to store metadata, including the date and time, image size, and audio annotation. SISRIF was proposed to WG18 to become the ISO standard for storing digital camera images.

In 1994, two alternative image formats were also proposed to ISO, TC42, and WG18. One alternative was the TIFF/EP format [12], led by George Lathrop of Eastman Kodak

Company. Another alternative was the Exif format [13], led by Makio Watanabe of Fuji Photo Film. TIFF/EP supported three different types of image data, including uncompressed RGB (using baseline TIFF 6.0), JPEG compressed data, and raw data. The advantage of TIFF/EP was that all three types of camera image data could be stored using a common TIFF wrapper. The disadvantage was that few existing software applications could read the JPEG compressed data file that was embedded in the TIFF/EP file. The Exif image format solved this problem by using two different types of files for compressed and uncompressed image data. A TIFF file and TIFF tags were used for uncompressed data. Compressed data was stored in a JPEG file, with the metadata stored as TIFF tags within a TIFF file embedded within an APP1 application segment at the beginning of the JPEG file. This enabled existing JPEG software applications to read the Exif-JPEG image data while ignoring the newly defined Exif metadata in the APP1 application segment. It was this backward compatibility with existing JPEG software that made Exif successful. Version 1 of Exif was approved by JEIDA in 1996. ISO/TC42/WG18 worked to standardize, to the extent possible, the metadata tags used in TIFF/EP and Exif. This ISO group developed the ISO 12234-1 standard for storing digital camera images on memory cards [14], which was published in 2001. The standard allowed either Exif or TIFF/EP to be used as the image format for new digital cameras.

Several other image formats were developed during the mid-1990s. The FlashPix format [15], developed by Kodak, Microsoft, Hewlett Packard, and Live Picture was made public in 1996. FlashPix used a hierarchical image representation, with 64×64 pixel image tiles, in a structured storage file. Each tile could be uncompressed or use baseline JPEG compression. FlashPix adopted most of the metadata definitions from TIFF/EP, but stored the metadata in property sets rather than TIFF tags. FlashPix files had a standard-sized thumbnail, and optionally included metadata extensions such as audio annotations and proprietary metadata extensions. FlashPix supported two color spaces: the wide-gamut KODAK PHOTOYCC Color Interchange Space [16], first used in Kodak's Photo CD system, and the NIF RGB color space. NIF RGB matched the typical indoor viewing conditions of CRT monitors, and led to the sRGB color space later standardized by the IEC [17].

In 1997, Exif was updated to version 2.0, in order to enable easy conversion between Exif files and FlashPix files. A new TIFF tag was defined to indicate whether or not the file contained sRGB color data. Some additional camera metadata tags, originally developed for TIFF/EP, were also added. An extension mechanism was defined to allow Flashpix extension streams to be stored in APP2 application segments within Exif files. The FlashPix extensions can include a standard audio annotation extension as well as proprietary extensions. While FlashPix never achieved significant adoption, these new features remain part of the current Exif-JPEG format.

The Camera Image File Format (CIFF) [18] was introduced by Canon in 1997 and was used by a number of camera makers for several years. CIFF stored images using JFIF files, but kept the metadata for all the images together in a so-called *hierarchical heap*. This approach enabled rapid searching of the metadata. CIFF provided specifications for naming image files and for arranging the image files in a standard directory structure. It also specified how audio annotations were stored in separate WAV (Waveform audio) format files associated with particular images.

In order to harmonize the competing Exif and CIFF formats, JEIDA approved version 1.0 of the DCF standard [19] in 1998. DCF uses modified versions of the file and directory-naming conventions from CIFF, while adopting version 2.1 of the Exif format. This revision defined an *Interoperability* tag, which indicates whether the Exif file meets the requirements for *DCF basic* files. These requirements include using sRGB color image data, as well as requiring that the Exif-JPEG file include a 160×120 pixel compressed thumbnail. The JEIDA DCF specification successfully unified how digital cameras stored JPEG-compressed images. Surveys conducted by the author's company confirmed that for many years, all popular digital cameras sold worldwide have produced JPEG files that conform to the requirements for DCF basic files.

In 2001, Epson introduced *Print Image Matching* (PIM) [20] which supported the sYCC color space [21] having a wider gamut than sRGB. PIM also included proprietary metadata intended to improve printing. In response, Exif was updated to version 2.2 in 2002 in order to support sYCC and to provide new, standardized metadata useful for printing. Examples of this new metadata include the camera's white balance, contrast, saturation, and sharpness settings. In 2004, DCF was updated by the Japan Electronics and Information Technology Industries Association (JEITA) to version 2.0 [22], and Exif was updated to version 2.2.1, in order to support an optional extended-gamut color space known as Adobe RGB.

13.4 Exif-JPEG Image Format Structure

Essentially all current consumer digital cameras store fully processed, JPEG compressed images that meet the requirements for *DCF basic files*, using the JPG extension. DCF defines how the files are named and organized in directories, as described later in this chapter. It also sets restrictions and metadata requirements for the Exif-JPEG files. Figure 13.3 shows the structure of an Exif-JPEG image file. Exif-JPEG files are JPEG files that contain metadata in one or more Application Marker Segments (APPn) at the beginning of the JPEG file, as shown in the left side of the figure. Following the APPn segments, the file holds the quantization table (DQT), Huffman table (DHT), start of frame (SOF), and the compressed main image data.

The main image data is compressed using baseline JPEG image compression, which performs a DCT on 8×8 pixel blocks of Y (luminance) and color differential Cr and Cb signals. For DCF basic files, these color differential signals must be created from red, green, and blue signals provided in the sRGB color space. An optional color space, which corresponds to the Adobe RGB color specification, can be used in the *DCF optional files*, as defined in DCF version 2.0. Normally, so-called *4:2:0 color differential subsampling* is used for the main image. This means that there are twice as many rows and columns of Y samples as Cb or Cr samples. However, DCF basic files can also use so-called *4:2:2 subsampling*. This means that there are an equal number of rows of Y, Cb, and Cr samples, and twice as many columns of Y samples as Cb or Cr samples. The DCT coefficient values of the Y, Cb, and Cr signals are then quantized, entropy coded, and stored as the main image data in the Exif-JPEG file.

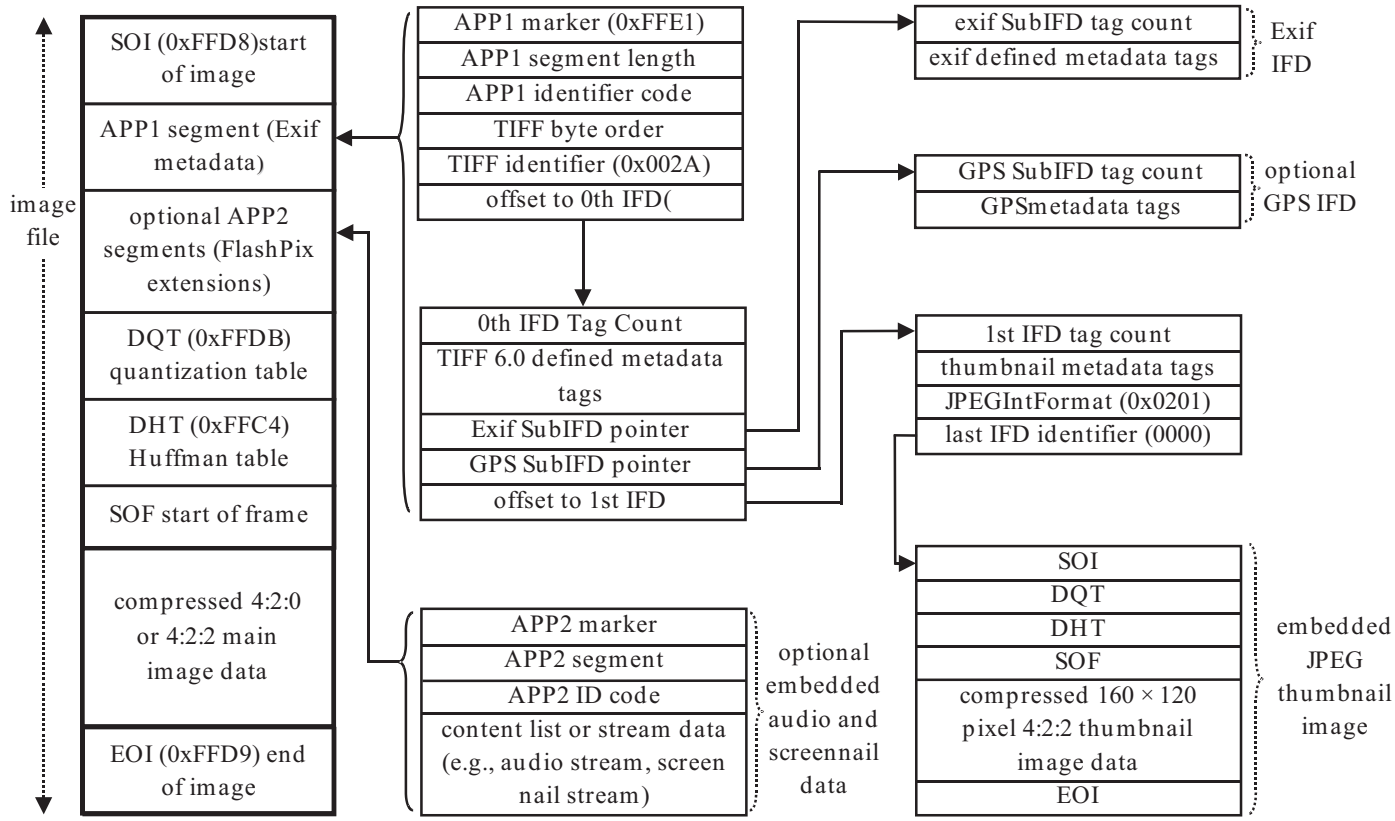


FIGURE 13.3

Exif/JPEG image file structure.

TABLE 13.2
Structure of Exif APP1 segment header.

Field	*Bytes*	*Field Definition*	*Hexadecimal value*
APP1	2	APP1 Marker.	FF, E1
Length	2	Total APP1 field byte count, including the 2-byte count value, but excluding the 2-byte APP1 marker itself.	
Identifier	6	This zero terminated and padded string (*Exif*) uniquely identifies this APP1 segment.	45, 78, 69, 66, 00, 00
TIFF Byte Order	2	Byte Order value is either: II (0×49, 0×49) (little endian) or MM (0×4D, 0×4D) (big endian)	49, 49
TIFF ID	2	TIFF Identifier (decimal 42)	2A, 00
Offset	4	Offset to 0th Image File Directory (IFD)	08, 00, 00, 00 if the IFD immediately follows, with II byte order.

The format of the APPn segments, which precede the main image data, is specified in Annex B of the JPEG standard, ISO/IEC 10918-1. Application segments can have different numbers (e.g., APP1 and APP2) that correspond to the different application marker values defined in the JPEG standard. The JPEG standard does not limit the number or sequence of application segments that can be included in a JPEG file. However, the Exif specification provides some restrictions on application segments. Exif requires that the JPEG file contain a specific APP1 segment immediately after the *Start of Image* (SOI) marker at the beginning of the file. This APP1 segment must have an identifier code, or label, with a value of *Exif*. The remainder of the APP1 segment is a TIFF file containing metadata and thumbnail image data. Following this APP1 segment, the file can optionally include a number of APP2 segments labelled *FPXR* (FlashPix-Ready). Storing metadata in these APP1 and APP2 segments yields image files with excellent backward compatibility because the JPEG standard requires that readers ignore unknown application segments.

Exif-JPEG files contain a TIFF file stored in an APP1 application segment immediately following the two-byte JPEG SOI marker. Table 13.2 shows the structure of the beginning portion of this APP1 segment. The first two bytes (FFE1 in hexadecimal notation, written as 0×FFE1) define the segment as an APP1 segment. The next two bytes are the length of the segment. Each application segment must be less than 64 Kbytes in length. The next six bytes provide ASCII characters of the identifier code field, which must be *Exif* followed by two zeros for termination and padding.

The TIFF Byte Order field follows the Exif Identifier Field. This Byte Order field is the first two bytes of the TIFF file that is embedded in the APP1 segment. It is followed by the decimal value 42, sometimes known as the TIFF *magic number*. Finally, the last entry in the table is the address offset to the Image File Directory (IFD0) structure that contains TIFF metadata tags. As a result, the last three entries in Table 13.2 contain the 8-byte header required for TIFF files.

Each IFD contains a group of metadata tags. The first two bytes of an IFD is a count field that indicates the number of tags present in the IFD. Each tag has a tag number. Tag numbers greater than 0×8000 are called *private* tags. Private tags are issued by Adobe, the registration authority for TIFF tags, to a particular company or group. The TIFF specification requires that all of the tags in an IFD be arranged in ascending order. Each TIFF tag is a 12-byte record with the following format:

1. Bytes 0-1: The Tag ID Field, which is a number that identifies the particular tag.
2. Bytes 2-3: The Data Type Field, which is one of the following values for the tags used in Exif-JPEG files:
   - 1 = BYTE, an 8-bit unsigned integer.
   - 2 = ASCII, a string of 8-bit bytes containing 7-bit ASCII character codes terminated with a NULL code.
   - 3 = SHORT, a 16-bit (2-byte) unsigned integer.
   - 4 = LONG, a 32-bit (4-byte) unsigned integer.
   - 5 = RATIONAL, is two LONGs. The first LONG is the numerator and the second LONG is the denominator.
   - 7 = UNDEFINED, indicates an 8-bit byte that can take any value depending on the field definition.
3. Bytes 4-7: The Count Field, which indicates the number of Data Type values (not the number of bytes) stored by the tag.
4. Bytes 8-11: The Value or Offset Field, which indicates the tag value(s) if they can be stored in these 4 bytes, or the offset to the data if more room is required. The offset value is the relative offset from the Byte Order field in the TIFF header.

The last 4 bytes of the IFD structure are either the offset to the next IFD, or are four zero bytes indicating that this is the last IFD.

In order to meet the requirements for DCF basic files, the APP1 segment in an Exif file contains two required IFDs. IFD0 provides metadata relating to the main JPEG compressed image, but no image data. IFD1 contains JPEG compressed data for the 160×120 pixel thumbnail. IFD0 includes some TIFF metadata tags that are defined in the TIFF 6.0 specification, such as the Make and Model. These tags must be used as specified in the TIFF 6.0 document. IFD0 also includes the Exif IFD Pointer, 0×8769. This tag uses a LONG Data Type with a count of one, where the value is the offset to the Exif IFD. This private tag number was registered for use in the Exif specification. Therefore, all of the tags included in the Exif IFD are private tags, which must be used as specified in the Exif standard. The Exif IFD stores tags that describe the capture conditions used when capturing the digital image stored in the file. An Interoperability IFD Pointer tag is located within the Exif IFD, with a tag ID of $0\times$A005. As shown in Table 13.3, the Interoperability IFD contains tags indicating whether the Exif image has followed the DCF interoperability rules, and thus whether the Exif file is a DCF basic file or a DCF optional file.

IFD0 optionally includes the GPS Info IFD Pointer, with a tag ID of 0×8825. The GPS IFD stores tags that provide Global Positioning Satellite information indicating the location of the camera when the image was captured. It is used by a number of digital cameras and camera phones that include GPS receivers, either integrated in the camera or attached to the camera.

TABLE 13.3
Interoperability IFD metadata tags.

Tag Field Name	*Number*	*T*	*U*	*Description*
InteroperabilityIndex	0×0001 1	M	N	A value of R98 indicates a DCF *basic file* that has used sRGB color encoding. A value of R03 indicates a DCF *optional file* that has used Adobe RGB color encoding. These values are NULL terminated.
InteroperabilityVersion	0×0002 2	M	N	This tag records the version of the InteroperabilityIndex value. The value is the 4-byte ASCII 0100, meaning Version 1.00. This is not terminated by NULL, as the tag Type is UNDEFINED.

The Exif specifications allow a thumbnail image in an Exif-JPEG file to be stored either as uncompressed TIFF image data or as a compressed JPEG image stream. However, for DCF basic files, compressed thumbnails are required. This has made the use of uncompressed thumbnails obsolete. The JPEG compressed thumbnail must be 160×120 pixels, and must use 4:2:2 color differential sampling. Because the size and orientation of the thumbnail is fixed, the left and right edges of the thumbnail image must be *padded* when a main image having a portrait aspect ratio is stored. Black pixels are recommended, but not required, for padding. The top and bottom of the thumbnail image are padded when a landscape image having an aspect ratio wider than 4:3 is used for the main image.

The compressed thumbnail image data is stored as a JPEG stream within IFD1 of the TIFF file contained in APP1 of the Exif-JPEG file, as shown at the bottom right of Figure 13.3. The stored thumbnail data is itself a JPEG file, beginning with a Start of Image

TABLE 13.4
Thumbnail IFD tags.

Tag Field Name	*Number*	*Description*
Compression	0×0103 259	A value of 6 must be used for DCF basic files, which the Exif specification defines as the value for JPEG compressed thumbnail data.
Xresolution	0×011A 282	The number of pixels in the image width direction per ResolutionUnit to use when reproducing the image.
Yresolution	0×011B 283	The number of pixels in the image height direction per ResolutionUnit to use when reproducing the image.
ResolutionUnit	0×0128 296	Used to define the scale of the Xresolution and Yresolution values. Typically, inches are used.
JPEGInterchangeFormat	0×0201 513	This tag provides the offset to the location of the thumbnail image data.
JPEGInterchangeFormatLength	0×0202 514	This tag provides the size of the JPEG compressed bit stream.

(SOI) marker and ending with an End of Image (EOI) marker, but is not allowed to contain Application Markers or Restart Markers. The tags listed in Table 13.4 are also required to define the thumbnail in IFD1.

13.5 Exif-JPEG Digital Camera Metadata

The metadata used in Exif-JPEG files is defined in the Exif specifications. However, this information is not always well understood by English readers, for several reasons. For example, the Exif specifications assume that the reader is familiar with both the JPEG standard (ISO 10918) and the TIFF 6.0 image file format. Furthermore, because the English versions of Exif are translations from the original Japanese language documents, some of the descriptions are brief and ambiguous. Table 13.5 and Table 13.6 provide a more detailed explanation of some of the key metadata, provided by digital cameras, that is stored in Exif-JPEG image files. Table 13.5 lists metadata found in IFD0, and Table 13.6 lists metadata found in the Exif IFD. The tags are listed in numerical order. The third column of these tables uses the following key to indicate the type (T) of metadata. M indicates a mandatory tag that must be recorded in DCF basic files, R indicates a Recommended tag that should be recorded, if possible, and O indicates an Optional tag that may be recorded. The Exif specifications provide guidelines describing what metadata tags need to be updated when an Exif file is edited by a software application. The fourth column of Table 13.5 and Table 13.6 indicates which metadata may need to be updated (U) when the file is edited.

The Exif specification does not explicitly describe how to add new, vendor-defined metadata to Exif files. However, because it uses TIFF tags to store metadata, it is possible to register new TIFF tags and to include these tags within IFD0 of the TIFF file stored in the APP1 segment of the Exif image file. The *Private Fields and Values* section of the TIFF 6.0 specification describes the process for registering private TIFF tags. The remainder of the fields that comprise the 12-byte TIFF tag can be set as required by the registrant of the tag. The newly registered TIFF tag can then be placed in the 0th IFD of an Exif JPEG image. However, these tags should not conflict with, or duplicate, tags that are already defined in the Exif specifications. For example, the use of an ICC profile in an Exif file is not allowed, because the information in the ICC profile will either conflict with, or duplicate, the information contained in some of the color related tags defined in the Exif specification.

Instead of using TIFF tags, metadata can be stored using other types of encodings, such as XML (extensible markup language). The DIG35 specification created XML metadata definitions for digital photography. These definitions were later included in the JPEG 2000 image format, which is discussed later in this chapter. In 2001, Adobe defined XMP (extensible metadata platform), an XML based metadata model that can be used with various defined sets of metadata items. XMP defines schemas for recording how an image has been captured, edited, and assembled into a final image. XMP metadata can be added to TIFF files images using the XMP tag (700), and to JPEG images using an APP1 segment with the identifier code http://ns.adobe.com/xap/1.0/. However, this XMP APP1 segment should not be used in an Exif-JPEG file, since the Exif specification allows only APP1 segments that use the Exif identifier shown in Table 13.2.

TABLE 13.5
TIFF metadata tags in Oth IFD.

Tag Field Name	*Number*	*T*	*U*	*Description*
ImageDescription	0×10E 270	R	N	A character string with the title of the image, using a one-byte (ASCII) character code. If Unicode is required for the title, the UserComment tag in the Exif IFD is typically used.
Make	0×010F 271	M	N	A character string with the name of the manufacturer of the digital camera.
Model	0×0110 272	M	N	A character string with the model name / number of the digital camera.
Orientation	0×112 274	R	Y	The orientation of the stored main image data, indicating whether the reader should rotate the image before display: If the value is 1, the image is in the normal orientation and should not be rotated when displayed. If the value is 6, the image should be rotated 90 degrees clockwise; and if the value is 8, the image should be rotated 90 degrees counter-clockwise. Note that if the rotated image is saved, the orientation tag value should be set equal to 1.
XResolution	0×011A 282	M	N	The number of pixels in the image width direction per ResolutionUnit, to use when reproducing the image.
YResolution	0×011B 283	M	N	The number of pixels in the image height (Y) direction per ResolutionUnit, to use when reproducing the image.
ResolutionUnit	0×0128 296	M	N	The unit for measuring both XResolution and Yresolution, typically set to inches.
Software	0×131 305	O	Y	The name and version number of the firmware used by the digital camera, written as an ASCII string.
DateTime	0×132 306	R	Y	The date and time the file was last modified. The format is YYYY:MM:DD HH:MM:SS with time shown in 24-hour format and the date and time separated by one blank character.
Artist	0×13B 315	R	N	The name of the camera owner or the photographer, written as an ASCII string.
YCbCrPositioning	0×213 531	M	N	The positioning of chrominance components in relation to the luminance component. A value of 2, co-sited, is normally used for Y:Cb:Cr = 4:2:2. A value of 1, centered, is normally used for Y:Cb:Cr = 4:2:0.
Copyright	0×8298 33432	O	N	A copyright notice for the photographer or editor, written as an ASCII string.
ExifIFDPointer	0×8769 34665	M	N	The value of this tag is the offset from the start of the TIFF header to the position where the Exif IFD is stored.
GPSInfoIFDPointer	0×8825 34853	O	N	This tag is only recorded when global position information for the image is recorded. The value is the offset to the position where the GPSInfoIFD is stored.

TABLE 13.6
Camera capture metadata tags in Exif IFD.

Tag Field Name	*Number*	*T*	*U*	*Description*
ExposureTime	0×829A 33434	R	N	The time, in seconds, that the image was exposed on the image sensor.
Fnumber	0×829D 33437	O	N	The lens f-number, which is the ratio of the lens aperture to its focal length, which was used when the image was captured.
ExposureProgram	0×8822 34850	O	N	The mode used by the camera to set the exposure time and f-number. Allowed values include 1 for manual mode (user selects all settings), 2 for the normal automatic exposure mode, 3 for aperture priority mode (user sets the f-number), 4 for shutter priority (user sets the exposure time), 5 for creative mode (biased toward depth of field), 6 for action mode (biased toward fast shutter speed), 7 for portrait mode (background out of focus), and 8 for landscape mode (background in focus).
SpectralSensitivity	0×8824 34852	O	N	An ASCII string that gives the spectral sensitivity of each color channel, using an ASTM defined format.
ISOSpeedRatings	0×8827 34855	O	N	Indicates the ISO Speed and ISO Latitude of the camera or input device as specified in ISO 12232. The first value is the ISO saturation speed rating and the last two optional values are the minimum and maximum ISO Speed Latitude value.
OECF	0×8828 34856	O	N	A table of values that describe the Opto-Electronic Conversion Function of the camera, measured as specified in ISO 14524.
ExifVersion	0×9000 36864	M	N	The version of the Exif standard used for the image file. An Exif version 2.22 image file has a value of 0222 stored as 4-byte ASCII. Because the type is UNDEFINED, there is no NULL for termination.
DateTimeOriginal	0×9003 36867	M	N	The date and time the picture was taken by the camera. The format is YYYY:MM:DD HH:MM:SS with time shown in 24-hour format.
DateTimeDigitized	0×9004 36868	M	N	The date and time when the image was stored as digital data, which is normally identical to the DateTimeOriginal.
Components-Configuration	0×9101 37121	M	N	The order of the compressed data, which has a value of 1230 for Y, Cb, Cr data.
CompressedBitsPerPixel	0×9102 37122	O	Y	The compression setting used when the image was compressed by the camera, using a bits per pixel value.

TABLE 13.6
Camera capture metadata tags in Exif IFD. (*cont.*)

Tag Field Name	*Number*	*T*	*U*	*Description*
ShutterSpeedValue	0×9201 37377	O	N	The APEX (Additive Systems of Photographic Exposure) time value of the shutter speed. The Exposure Time = 1/(2ShutterSpeedValue).
ApertureValue	0×9202 37378	O	N	The APEX value of the lens aperture used when the image was captured. For example, an APEX value of 6.0 indicates f/8.0.
BrightnessValue	0×9203 37379	O	N	The luminance of the scene, as measured by the camera. The value is expressed in APEX units, given by the equation Bv = log2 L/.3K, where L is the luminance (brightness) in candelas/m2 and K is the reflected light metering constant 11.4 candelas/m2. For example, a BV of 5 corresponds to 109 candelas/m2.
ExposureBiasValue	0×9204 37380	O	N	Indicates the amount of intentional over or under exposure, using an APEX value. For example, 2.0 means 2 photographic stops overexposure that has brightened the image, and –.5 indicates 1 stop underexposure that has darkened the image.
MaxApertureValue	0×9205 37381	O	N	The smallest f-number that the camera lens be set to, expressed as an APEX value.
SubjectDistance	0×9206 37382	O	N	The distance between the camera and the main subject of the image, given in meters.
MeteringMode	0×9207 37383	O	N	The type of exposure metering used when capturing the image. Some of the allowed values include 1 for average, 2 for center-weighted average, 3 for spot, 4 for multi-spot, and 5 for pattern.
LightSource	0×9208 37384	O	N	The type or color temperature of the light source that illuminated the scene. Some of the allowed values include 0 for unknown, 1 for daylight, 2 for fluorescent, 3 for incandescent, and 4 for flash.
Flash	0×9209 37385	R	N	Provides information on whether camera flash was used when the image was captured. Bit 0 indicates if the flash fired. Bits 1 and 2 indicate if the flash was quenched. Bits 3 and 4 indicate the flash mode. Bit 5 indicates whether or not the camera included a flash, and bit 6 indicates if the red eye flash mode was used.
FocalLength	0×920A 37386	O	N	The true focal length of the lens, in mm. As the size of the image sensor in a digital camera is normally much smaller than the 35 mm film frame, the value is normally much less than the 35 mm equivalent focal length.
SubjectArea	0×9214 37396	O	Y	The location and area of the main subject in the captured image.

TABLE 13.6
Camera capture metadata tags in Exif IFD. (*cont.*)

Tag Field Name	*Number*	*T*	*U*	*Description*
MakerNote	0×927C 37500	O	N	Different camera makers store a variety of different information.
UserComment	0×9286 37510	O	Y	This tag is used to write keywords or comments concerning the image. It can use either ASCII one-byte character codes, or two byte JIS or Unicode character codes.
SubsecTime-Original	0×9291 37521	O	N	This tag value is used with the DateTimeOriginal tag value to resolve the sub-second time the picture was captured, which is useful when a burst of images is captured.
FlashPixVersion	0×A000 40960	M	N	This mandatory tag has a value of 0100 recorded as 4-byte ASCII, to indicate FlashPix format Version 1.0. Because the type is UNDEFINED, there is no NULL for termination.
ColorSpace	0×A001 40961	M	N	For DCF basic files, a value of 1 is used to indicate sRGB. For DCF optional files, the uncalibrated value (0xFFFF) is used. The InteroperabilityIndex tag is then checked to see if it is set to R03 which means that the optional (e.g., Adobe RGB) color space is used. In this case, the file names also need to begin with the underscore character.
PixelXDimension	0×A002 40962	M	Y	The tag value is the Image Width of the main image, without including the padding pixels that may be present in the JPEG image stream.
PixelYDimension	0×A003 40963	M	Y	The tag value is the Image Height of the main image. As vertical data padding is unnecessary, this is the same value as that stored in the SOF (Start of Frame) JPEG marker.
RelatedSoundFile	0×A004 40964	O	N	This tag is used to store the name of an audio file that is related to the image data. The audio file must have the same DCF name as the Exif file, and use the WAV extension.
InterOperability-IFD Pointer	0×A005 40965	M	N	The value of this tag is the offset from the start of the TIFF header byte order field to the position where the Interoperability IFD is stored.
SpatialFrequence-Response	0×A20C 41484	O	N	A table that provides the spatial frequency response (SFR) of the digital camera in the horizontal, vertical, and diagonal directions, as measured using ISO 12233.
ExposureIndex	0×A215 41493	O	N	The exposure index used by the digital camera when the particular image was captured, as defined in ISO 12232.

TABLE 13.6
Camera capture metadata tags in Exif IFD. (*cont.*)

Tag Field Name	*Number*	*T*	*U*	*Description*
SensingMethod	0×A217 41495	O	N	The method for sensing the color image. Allowed values include 2 for cameras that use one-chip color area image sensors, 4 for three-chip color cameras, 5 for color-sequential cameras, and 7 for tri-linear color sensors.
FileSource	0×A300 41728	O	N	The type of device that created the file. Allowed values include 1 for film scanners, 2 for print scanners, and 3 for digital cameras.
CFAPattern	0×A302 41730	O	N	This tag indicates the geometric pattern of the color filter array used by the digital camera.
CustomRendered	0×A401 41985	O	Y	This tag indicates if the camera or computer has processed the image to produce a custom-rendered image, such as by intentionally coloring or overexposing the image. The value is 0 if the normal processing should be used when printing or displaying the image. The value is 1 if further processing, such as auto-correction, should be disabled or minimized when printing or displaying the image.
ExposureMode	0×A402 41986	R	N	The camera's exposure mode setting. Allowed values include 0 for automatic exposure modes, 1 for manual exposure modes, and 2 for automatic exposure bracketing modes, which capture an exposure series of the same scene.
WhiteBalance	0×A403 41987	R	N	The camera's white balance mode. Allowed values are 0 for automatic white balance, and 2 for manual white balance.
DigitalZoomRatio	0×A404 41988	O	N	The digital zoom ratio used to capture the image. For example, 2/1 indicates 2× digital zoom. If the numerator of the recorded value is 0, digital zoom was not used.
FocalLengthIn-35mmFilm	0×A405 41989	O	N	The focal length of the lens, in mm, for an equivalent field of view from a 35 mm film camera. A value of 0 means the focal length is unknown.
SceneCaptureType	0×A406 41990	R	N	The type of scene that was shot, normally determined by a camera mode setting. Allowed settings include 0 for standard scenes, 1 for landscapes, 2 for portraits, and 3 for night scenes.
GainControl	0×A407 41991	O	N	The gain setting of the camera, which increases the camera's exposure index. Allowed values are 0 for normal gain setting, 1 for low gain up settings, 2 for high gain up settings, 3 for low gain down settings, and 4 for high gain down settings.

TABLE 13.6
Camera capture metadata tags in Exif IFD. (*cont.*)

Tag Field Name	*Number*	*T*	*U*	*Description*
Contrast	0×A408 41992	O	N	The camera's contrast setting. Allowed values are 0 for the normal setting, 1 for low contrast (e.g., soft) settings, and 2 for high contrast (e.g., hard) settings.
Saturation	0×A409 41993	O	N	The camera's color saturation setting. Allowed values are 0 for the normal setting, 1 for low color saturation settings, and 2 for high color saturation settings.
Sharpness	0×A40A 41994	O	N	The camera's sharpness setting. Allowed values are 0 for the normal setting, 1 for low sharpness (e.g., soft) settings, and 2 for higher sharpness (e.g., hard) settings.
DeviceSettings-Description	0×A40B 41995	O	N	A table that can be used to store camera settings that are not included in other tags.
SubjectDistanceRange	0×A40C 41996	O	N	The approximate distance from the camera to the subject. Allowed values are 0 for unknown, 1 for macro setting, 2 for a relatively close subject, and 3 for a distant subject.
ImageUniqueID	0×A420 42016	O	N	This tag value is a globally unique identifier (GUID) for the image. It is recorded as an ASCII string of 32 hexadecimal values that encode the 128-bit GUID.

There are two options for storing audio annotations along with Exif-JPEG image files. The first option uses a separate WAV file, and the second option embeds the WAV file within APP2 segments in the Exif image file. The second approach ensures that the audio file gets transferred along with the image file, but it increases the size of the image file when a very long audio recording is made.

The first option uses associated image and file names along with metadata, and the requirements for this option are described in detail in the Exif specification. Exif defines the RelatedSoundFile tag (A004.H), which stores the name of the associated WAV file. The Exif specification also defines the details associated with the WAV RIFF file format, which stores the associated image file name within the WAV file. The DCF specification provides naming conventions for associating these image and audio files. Both files have the same DCF file number, but different file extensions: *JPG* for image files and *WAV* for audio annotation files. This makes the two files part of the same *image object*, and the DCF specification requires that the two files be copied or deleted as a pair.

The second option embeds the audio WAV file within APP2 segments, as a FlashPix-Ready data stream. Because there is no limit to the number of APP2 segments present in the Exif file, there are no limits to the length of the audio WAV file data. However, the APP2 segments must use the FPXR identifier code shown in Figure 13.3, since this is the only type of APP2 segment allowed in an Exif-JPEG file. The Exif specification defines the format of APP2 segments for storing FlashPix-Ready data streams in *Interoperability Structure of APP2 in Compressed Data* section. Audio is stored as defined in the *FlashPix audio*

extension specification [23]. There are two kinds of APP2 segments used for recording audio data as a FlashPix extension. The first is a *Contents List Segment* that records the list of data streams stored in subsequent APP2 segments. This contents list creates an important implied data element called the em Index to Contents List. The first stream in the contents list is assigned a zero value in the Index to Contents List and subsequent streams are assigned an ascending value. The APP2 segments that store the data streams, referred to as *Stream Data Segments*, use the Index to Contents List value to identify the data stream that is associated with the data stored in that APP2 segment.

The APP2 segments can also be used to store privately defined image data. For example, the thumbnail image size required for DCF basic files, 160×120 pixels, is too small to provide a sharp image for digital cameras that have a relatively large LCD display. Storing a larger *screen nail* image in the Exif-JPEG file provides a higher-resolution review image on the camera. Kodak has used the FlashPix-Ready APP2 segments defined in Exif to store JPEG compressed *screen nail* image data in some digital cameras to provide a sharper review images.

13.6 Raw Image Formats

A raw image file stores the data provided by the single-chip color sensor in a digital camera prior to most image processing. The raw image data is processed when the file is read, typically using a special software application on a host computer. This enables higher quality for several reasons. First, the image data is not degraded by the 8-bits per component, lossy baseline JPEG compression. Instead, the image data can be uncompressed data, typically using 12 or more bits per color channel. Since image processing is normally performed using a powerful processor on a host computer, the image processing algorithms can be more complex. Furthermore, the user can typically monitor the results of the processing and adjust the image processing parameters for specific images. Digital cameras that create raw files normally come with proprietary software that performs this image processing. For details on image processing in digital cameras refer to Chapters 1 and 3.

The image data in the raw file can be stored either uncompressed or using lossless compression. However, in order to properly interpret the color data, the meaning of the digital color values stored in the digital image file must be identified in the file and understood by the software application that processes the image. At a minimum, this requires that the color filter pattern, and the color responses of the color channels, be specified. In many cases, other parameters are also needed to enable the software application to perform the appropriate processing. These parameters can include user-adjustable camera settings, such as white balance and sharpness settings. They can also include camera characterization information, such as sensor noise data and lens correction data.

TIFF/EP, which is based on the TIFF 6.0 specification, was the first standard image format supporting raw image data. Almost all digital cameras that support raw files use TIFF version 6.0, and many use formats compatible with TIFF/EP. TIFF/EP uses the tags shown in Table 13.7 to define the uncompressed raw color filter array (CFA) image data. Chapters 1 and 5 discuss CFA patterns in detail.

TABLE 13.7
TIFF/EP raw image format tags.

Tag Field Name	*Number*	*Description*
ImageWidth	0×0100 256	Stores the width of the CFA image data.
ImageLength	0×0101 257	Stores the length of the CFA image data.
BitsPerSample	0×0102 258	Stores the number of bits per CFA sample, which is often 12 bits.
Compression	0×0103 259	Has a value of 1 if the CFA data is uncompressed or 7 if lossless JPEG compression is used. Other values can be used to signify proprietary compression.
PhotometricInterpretation	0×0106 262	A value of 32803 indicates that the color space is color filter array data.
StripOffsets	0×0111 273	Stores the offset to the image data.
SamplesPerPixel	0×0115 277	Has a value of 1, as each pixel has only one sample.
ICC Color Profile [24]	0×8773 34675	Used to define the RGB reference primaries, white point, and opto-electronic conversion function.
CFARepeatPatternDim	0×828D 33421	Used to encode the repeating dimensions of the color filter array pattern.
CFAPattern	0×828E 33422	Used to encode the color filter array pattern.

In order to improve some workflows, certain raw formats include a processed JPEG file inside the TIFF file. Figure 13.4 shows the structure of a raw file produced by the KODAK EASYSHARE P880 Zoom Digital Camera. This Kodak raw format uses the *KDC* file extension, and is based on TIFF-EP. The TIFF image file includes the JPEG Interchange Format tag, which is a pointer to a full resolution Exif-JPEG image that has been fully processed by the camera. This Exif-JPEG file, shown in the middle of Figure 13.4, can be extracted and used by applications and equipment that are not capable of performing raw image processing. The file also includes an Exif IFD Pointer to the camera settings tags normally stored in Exif-JPEG files.

The largest portion of the file is the Raw CFA sensor image data, which stores 12 bits per sample data from the Bayer pattern color sensor used in the KODAK EASYSHARE P880 Zoom Digital Camera. The file also includes a number of image reconstruction parameters, such as the white balance, sharpening, and noise cleaning settings. When the image is processed on the host computer, the stored image reconstruction parameters are used as the default image processing settings. The user can adjust these settings, in order to modify the processing used for particular images.

Digital cameras from other companies create raw files having slightly different arrangements. For example, the Nikon D70 Digital Camera produces *Nikon Electronic Format* raw files, using the *NEF* extension that is also based on TIFF/EP. These files include an uncompressed 160×120 thumbnail image in IFD0. SubIFDs tags point to the raw data for the main image and to a compressed rendered main image.

Canon digital cameras have used two different types of raw files. The Canon Rebel 10D camera produces raw files defined as part of CIFF, using the *CRW* extension. CIFF uses tags having a 10-byte structure, compared to the 12-byte tags used by TIFF. Another difference is that the offsets to the data in CIFF tags are relative to the start of the data block for each directory in a CIFF file. In a TIFF file, the offset to the data is relative to the first byte of the image file, which is the byte order field. Canon uses a raw format based on TIFF 6.0, with the *CR2* extension, in their high-end digital SLR cameras such as the Canon EOS-1Ds Mark II camera. Various-sized JPEG compressed processed images are stored in the 0th IFD, 1st IFD, and 2nd IFD. The 3rd IFD contains the compressed raw image data.

One problem with raw formats is that the processing used to convert the raw files to finished color images is often proprietary. Therefore, if the image processing software becomes unavailable in the future, the image file either may not be usable, or may be of unknown quality, if alternative image processing is used. While the TIFF/EP standard defines how raw image files can be stored, TIFF/EP support by compliant readers is optional.

In 2004, Adobe defined a raw format called *DNG* [25]. DNG is compatible with TIFF/EP, but defines additional tags used by some raw format cameras, and sets restrictions on the values of some TIFF tags. The ISO/TC42/WG18 group responsible for TIFF/EP began work on a revision to the TIFF/EP standard in 2006. In 2007, Adobe submitted their DNG specification for consideration by WG18. Therefore, the revised version of TIFF/EP may include a raw profile that adopts tags and restrictions from the DNG specification.

13.7 Directory and Control Formats

The DCF specification defines how image files are named and arranged into folders on a memory card. Figure 13.5 is an example of a DCF-compliant directory structure, which requires that the memory card be formatted using the DOS FAT file system. Images are stored in the DCIM (Digital Camera Images) directory, directly under the root directory of the removable media. The DCIM directory includes one or more DCF directories. The names of these DCF directories are eight characters in length. The first three characters provide the directory number, which must be between 100 and 999. The final five characters are *free characters*, chosen by the camera vendor. Only the capital letters A through Z, the numbers 0 through 9, and the underscore character may be used. The five free characters are often used to indicate the camera model number. In Figure 13.5, the `100KV705` and `102KV705` directories store images from a KODAK EASYSHARE V705 Dual Lens Digital Camera, and the `101KP880` directory stores images from a KODAK EASYSHARE P880 Camera. When a memory card is swapped between different model cameras, the camera normally creates a new DCF directory having the next directory number.

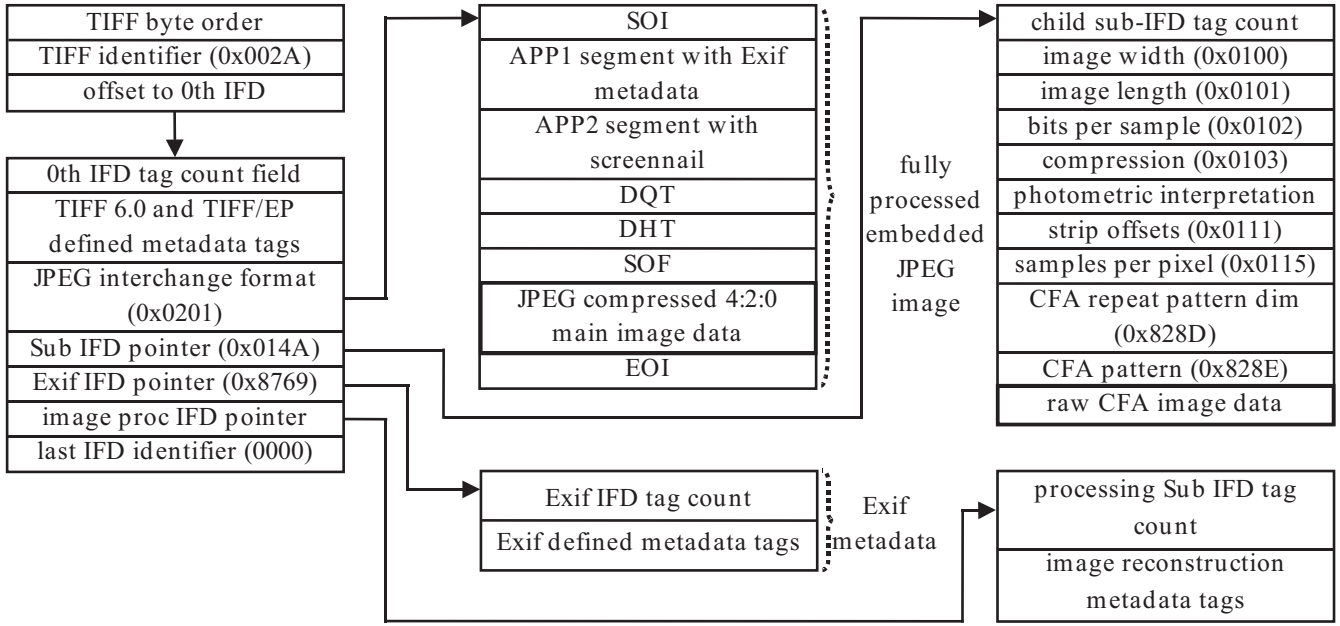


FIGURE 13.4

Example TIFF/EP raw image file structure.

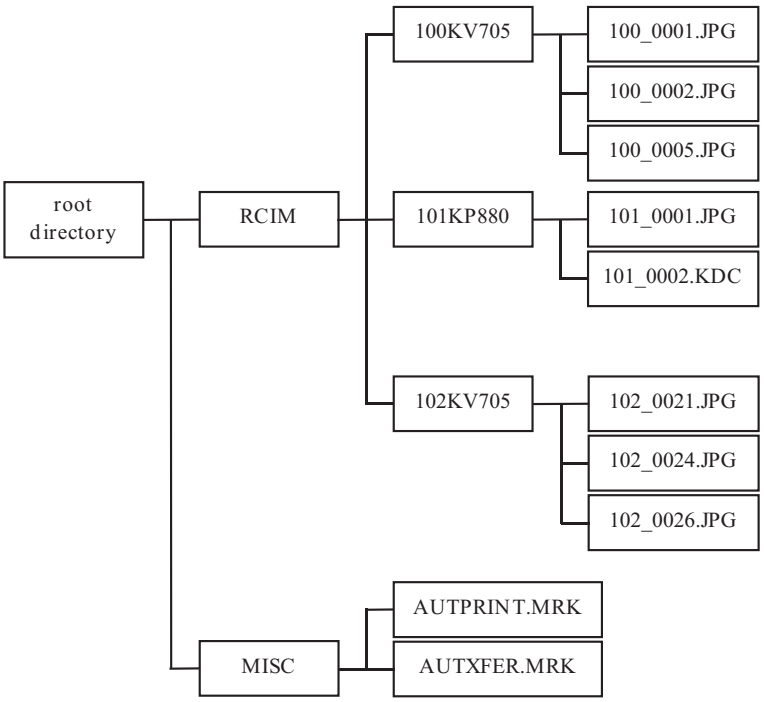


FIGURE 13.5

DCF file naming example.

Within each DCF directory, DCF basic image files are stored using an eight-character name followed by the *JPG* file extension. The first four characters are *free characters* chosen by the camera vendor. The last four characters are the file number, which ranges from 0001 to 9999. It is common to use the DCF directory number as part of the *free characters* to eliminate file name conflicts if images from multiple DCF directories are later transferred to the same directory. But this is not required.

It is possible to store raw image files within a DCF directory using a file extension other than JPG. For example, in Figure 13.5, the 101KP880 directory includes both a DCF basic Exif-JPEG file named `101_0001.JPG`, and a raw file named `101_0002.KDC`. In order to view the subject of the raw file on a camera that does not support raw image processing, some cameras also store a DCF thumbnail file. This thumbnail file has the same DCF file number, but uses the THM extension. Audio annotations can be stored in a separate file using the WAV extension. Files that have the same DCF file number, but a different extension, are considered part of the same DCF object, and need to be deleted, moved, or copied together.

Many consumer digital cameras allow images to be selected for printing as they are reviewed on the camera. The image selections are recorded in a text file that conforms to the *Digital Print Order Format* (DPOF) specification [26]. The camera or memory card can then be directly connected to a printer, which reads the DPOF file and prints the selected images. The DPOF print order file must be named AUTPRINT.MRK and must be located in a folder named MISC under the media root directory, as shown in Figure 13.5.

```
[HDR]
GEN REV = 01.10
GEN CRT = "KODAK V705 DUAL LENS DIGITAL CAMERA"
GEN DTM = 2007:04:19:19:47:24
[JOB]
PRT PID = 001
PRT TYP = STD
PRT QTY = 001
IMG FMT = EXIF2 -J
IMG SRC = "../DCIM/100KV705/100_0002.JPG"
[JOB]
PRT PID = 002
PRT TYP = STD
PRT QTY = 002
IMG FMT = EXIF2 -J
IMG SRC = "../DCIM/101KP880/101_0001.JPG"
[JOB]
PRT PID = 003
PRT TYP = STD
PRT QTY = 001
IMG FMT = EXIF2 -J
IMG SRC = "../DCIM/102KV705/102_0026.JPG"
```

FIGURE 13.6

DPOF Auto Print file example.

The DPOF version 1.0 specification was developed by Eastman Kodak Company, Canon Inc., Fuji Photo Film Co., Ltd., and Matsushita Electric Industrial Co., Ltd. in 1998. DPOF was updated to version 1.10 in 2000. Version 1.10 includes optional parameters to enable several images to be printed on one page and to specify the size of the print. Part 2 of DPOF version 1.10 defines an *Auto Transfer* (AUTXFER.MRK) file, which specifies images to be emailed, as well as *Auto Play* (AUTPLAYn.MRK) files, which define the order of still images, video images, and audio to be used in slide shows.

Figure 13.6 is an example of a DPOF print order file. The HDR header section indicates that the file conforms to DPOF version 1.10, and was created by the KODAK EASYSHARE V705 Dual Lens Digital Camera on April 19, 2007 at 7:47 PM. Optional parameters can be used to provide the name, address, and phone number of the camera owner. Vendor-unique parameters can also be included in the file to provide other types of information.

The DPOF print order file shown in Figure 13.6 contains three print jobs, identified by sequential ID numbers. The first print job is for a quantity of one standard sized print of the Exif-JPEG version 2 image file having the pathname `../DCIM/100KV705/100_0002.JPG`. In other words, one print should be made using the file named `100_0002.JPG` in the DCF directory named `100KV705` shown in Figure 13.5. The second print job is for two copies of a standard sized print of the file named `101_0001.JPG` in the DCF directory named `101KP880`, and the third print job is for one copy of the file named `102_0026.JPG`. DPOF print files can also contain optional parameters that specify other print sizes and index prints, or indicate how the image is to be cropped or overlaid with text. But because DPOF-compliant printers are not required to support these optional parameters, few consumer digital cameras currently support them.

```
[HDR]
GEN REV = 01.10
GEN CRT = "KODAK V705 DUAL LENS DIGITAL CAMERA"
GEN DTM = 2007:04:19:19:48:34
[JOB]
PMT PID = 001
DST EML = "ken@kodak.com"
IMG FMT = EXIF2 -J
IMG SRC = "../DCIM/100KV705/100_0005.JPG"
[JOB]
PMT PID = 002
DST EML = "ken@kodak.com"
IMG FMT = EXIF2 -J
IMG SRC = "../DCIM/100KV705/102_0021.JPG"
[JOB]
PMT PID = 003
DST EML = "rob@kodak.com"
IMG FMT = UNDEF
IMG SRC = "../DCIM/101KP880/101_0002.KDC"
```

FIGURE 13.7

DPOF Auto Transfer file example.

Figure 13.7 is an example of a DPOF auto transfer file. The HDR header section indicates that the file conforms to DPOF version 1.10, and was created by the KODAK EASYSHARE V705 Dual Lens Digital Camera on April 19, 2007 at 7:48 PM. The file contains three transfer jobs, identified by sequential ID numbers. The first job is to transfer the Exif-JPEG version 2 image file having the pathname `../DCIM/100KV705/100_0005.JPG` to the email address `ken@kodak.com`. In other words, the image file named `100_0005.JPG` in the DCF directory named `100KV705` shown in Figure 13.5 should be used. The second job requests that the file named `102_0021.JPG` be sent to the same email address. The third job requests that the file named `101_0002.KDC` be sent to the email address `rob@kodak.com`.

13.8 Advanced Image Formats

The image file formats used by future digital cameras will continue to advance. This may happen by continuing to make small, compatible enhancements to the current Exif and TIFF formats, or by replacing these formats with new, incompatible formats. One example of a possible replacement is the JPEG 2000 image format [27] that was standardized in the late 1990's. JPEG 2000 uses the discrete wavelet transform, rather than the discrete cosine transform. A JPEG 2000 file provides a multi-resolution representation of an image, using bit-plane encoding [28]. This allows progressive decoding that supports both lossy and lossless compression. Metadata is stored using an XML schema, rather than using TIFF tags. Refer to Chapters 14 and 15 for detailed discussions on image compression.

The JPEG 2000 compressed bit stream is organized in a flexible binary container using a rich syntax. The JPEG 2000 code stream can be fragmented and ordered into different boxes, and each box can be edited separately. The syntax provides improved metadata support, compared with the application segment approach used by Exif-JPEG. It also supports many color-encoding standards, including both sRGB and extended gamut color RGB encodings, such as ROMM RGB [29], RIMM RGB [30], and scRGB [31]. These color spaces are supported by storing a restricted ICC profile in the file, which essentially provides instructions to allow the color image data to be easily converted into sRGB if needed.

In 2004, a profile for using JPEG 2000 in digital still cameras was approved as an American national standard [32]. This standard provides a way for cameras to write JPEG 2000 files with a full set of digital camera metadata in a way that can be correctly read and interpreted by other devices. It also defines file naming and directory rules that are consistent with DCF. While JPEG 2000 has been used in various government and medical applications, it is not yet supported by consumer digital cameras.

In 2006, Microsoft introduced a new image file format called HD Photo [33], also known as Windows Photo Media. HD Photo uses the same TIFF header described earlier, except that *little-endian* byte order is required and a different *magic number* is used. HD Photo stores images and metadata using TIFF-like tags and IFDs. The file can include Exif metadata using the standard Exif IFD, as well as XMP metadata, which uses XML. The HD Photo file extension can be either HDP or WDP.

Images are compressed using a proprietary, reversible, lapped bi-orthogonal transform. HD Photo uses a PixelFormat tag with a Globally Unique Identifier (GUID) to indicate which of over 50 different pixel representations is used for the image data. These range from monochrome, up to 8-channel color data, using either 8, 16, or 32 bits per channel. However, baseline decoders are required to support only a few of these options, providing RGB or monochrome data at 8 or 16 bits per channel. The RGB color encoding is defined either implicitly as sRGB or scRGB, depending on the GUID, or by using an embedded ICC profile. In 2007, HD Photo was submitted to the JPEG group (formally known as ISO/IEC JTC 1/SC 29/WG 1) for standardization. It is likely that this format, to be known as *JPEG XR*, will be approved and published as ISO 29199-2 in 2009.

13.9 Conclusion

Digital cameras store images using standard image file formats. This enables readers, including computers and photo printers, to make use of the image data. Most consumer cameras store fully-processed images using the *Exif compressed* image format. These Exif-JPEG files are named and organized into folders according to the DCF requirements, and can be read by baseline JPEG readers. Metadata created by the camera is stored using TIFF tags in an application segment near the beginning of the file. These tags store required metadata, such as the make and model of the camera, the capture date and time, and a 160×120 pixel *thumbnail* image. The tags can also store recommended and optional metadata, such as lens settings, camera parameters, and GPS coordinates. DPOF control files can be used to indicate which images will be automatically printed or emailed.

Most SLR and *prosumer* digital cameras feature a *raw* mode that stores the sensor image data prior to demosaicking. This provides higher image quality, but requires that the reader performs demosaicking and other camera image processing. The TIFF/EP format is used to store raw image data in some cameras. TIFF/EP defines metadata, such as the color filter pattern and an ICC profile, which describe the raw data values. Some digital cameras also use the Exif-uncompressed or TIFF/EP formats to store processed, uncompressed, images that can be read by baseline TIFF readers.

The use of metadata in image files is playing an increasingly important role in the management and organization of consumer image collections. The size of these collections grows every year. A typical family now has many thousands of images stored on the hard drive of their home computer. While almost all current digital cameras create files that fully comply with the Exif and DCF specifications, some popular imaging applications produce edited files that do not. Compliance problems often arise when metadata is added or updated. To create compliant software, the developers must fully understand the requirements contained in several different documents, some of which were translated from the Japanese language. Because certain prohibitions are not fully described, some developers have made design choices that cause interoperability problems. This chapter has attempted to clarify some of the key requirements, in order to improve compliance.

The Exif-JPEG image format has been widely used in digital cameras for more than a decade. Because it was designed to be flexible and extensible, new capabilities have been added, while maintaining backward compatibility. New image formats, including JPEG2000 and HD Photo, provide higher image quality and more capabilities. But they are not backward compatible with the existing Exif-JPEG format. This makes it difficult for any new format to gain acceptance. It is possible that one of these new formats will replace Exif-JPEG as the primary image format for consumer digital cameras. But it is more likely that Exif-JPEG will continue to be enhanced and used to store images in digital cameras for many years to come.

References

[1] Technical Standardization Committee on AV & IT Storage Systems and Equipment, Exchangeable image file format for digital still cameras: Exif Version 2.2, Japan Electronics and Information Technology Industries Association, JEITA CP-3451-1, April 2002. Available online: http://www.jeita.or.jp.

[2] Amendment 1 exchangeable image file format for digital image still cameras; Exif Version 2.21 (Amendment to Version 2.2), Japan Electronics and Information Technology Industries Association, JEITA CP-3451, 2002. Available online: http://www.jeita.or.jp.

[3] 2001 Electronic still picture imaging removable memory Part 2: TIFF/EP image data format, ISO 12234-2. Available online: http://www.iso.ch.

[4] Information technology - Digital compression and coding of continuous-tone still images: Requirements and guidelines, ISO/IEC 10918-1, 1994. Available online: http://www.iso.ch.

[5] W.B. Pennebaker and J.L. Mitchell, *JPEG Still image data compression standard*. New York: Van Nostrand Reinhold, 1993.

[6] J. Milch and K. Parulski, "Using metadata to simplify digital photography," in *Proceedings of the IS&T Conference on PIC*, Savannah, GA, USA, April 1999, pp. 26–30.

[7] C-Cube microsystems, JPEG file interchange format, Version 1.02, September 1992. Available online: http://www.jpeg.org/jpeg/index.html.

[8] Information technology - digital compression and coding of continuous-tone still images: Extensions, ISO/IEC 10918-3, 1997. Available online: http://www.iso.ch.

[9] Adobe Systems Incorporated, TIFF (Tag Image File Format), Revision 6.0, June 1992. Available online: http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf.

[10] Adobe Photoshop, TIFF technical notes JPEG compression, March 2002. Available online: http://partners.adobe.com/public/developer/en/tiff/TIFFphotoshop.pdf.

[11] PCMCIA, JEIDA specific extensions,Section 3, SISRIF (Still Image, Sound and Related Information Format), PC Card Standard, no. 12, February 1995.

[12] K. Parulski and G. Lathrop, "TIFF/EP, A flexible image format for electronic still cameras," in *Proceedings of the IS&T 48th Annual Conference*, Washington, D.C., USA, May 1995, pp. 425–428.

[13] M. Watanabe, F. Funazaki, Y. Shigyo, and S. Nishi, "An image data file format for digital still camera," in *Proceedings of the International Symposium on Electronic Photography*, Washington, D.C., USA, May 1995, pp. 421–424.

[14] Electronic still picture imaging Removable memory Part 1: Basic removable-memory module, ISO 12234-1, 2001.

[15] Flashpix format specification, Version 1.0, Eastman Kodak Company, September 1996. Available online: http://www.i3a.org.

[16] PhotoYCC color encoding and compression schemes, Eastman Kodak Company, KODAK Publication PCD-045, April 1994.

[17] Multimedia systems and equipment Colour measurement and management Part 2-1: Colour management default RGB colour space sRGB, IEC 61966-2-1, 1999. Available online: http://www.iec.ch.

[18] CIFF specification on image data file, Version 1.0, Revision 4, Canon Inc., December 1997.

[19] Design rule for camera file system, Version 1.0, Japan Electronic Industry Development Association (JEIDA), December 1998.

[20] Print image matching white paper, Epson America, Inc., 2001. Available online: http://www.printimagematching.com/what_is_pim.php.

[21] IEC 61966-2-1-amendment 1, January 2003. Available online: http://www.iec.ch.

[22] Design rule for camera file system: DCF Version 2.0, JEITA CP-3461, September 2003. Available online: http://www.jeita.or.jp.

[23] Flashpix audio extension specification, International Imaging Industry Association (I3A). Available online: http://www.i3a.org/i_Flashpix.html.

[24] Image technology colour management Architecture, profile format, and data structure, ICC.1:2004-10 (Profile version 4.2.0.0). Available online: http://www.color.org.

[25] Adobe digital negative (DNG) specification, Version 1.1.0.0, Adobe Systems Inc., February 2005. Available online: http://www.adobe.com/products/dng/.

[26] Summary of DPOF. Available online: http://panasonic.jp/dc/dpof_110/white_e.htm.

[27] Information technology JPEG 2000 image coding system; Information technology JPEG 2000 image coding system extensions; Information technology JPEG 2000 image coding system conformance, ITUT Rec. T.800 — IS0 154441:2002; ITUT Rec. T.801 — IS0 154442:2002; ITUT Rec. T.803 — IS0 154444:2002. Available online: http://www.iso.ch.

[28] P.N. Topiwala, *Wavelet image and video compression*. Dordrecht, the Netherlands: Kluwer Academic Publishers, 1998.

[29] Photography and graphic technology Extended colour encodings for digital image storage, manipulation and interchange Part 2: Reference output medium metric RGB colour image encoding (ROMM RGB), ISO/TS 22028-2, 2006. Available online: http://www.iso.ch.

[30] Photography and graphic technology Extended colour encodings for digital image storage, manipulation and interchange Part 3: Reference input medium metric RGB colour image encoding (RIMM RGB), ISO/TS 22028-3, 2006. Available online: http://www.iso.ch.

[31] Multimedia systems and equipment - Colour measurement and management - Part 2-2: Colour management - extended RGB colour space scRGB, IEC 61966-2-2, 2003. Available online: http://www.iec.ch.

[32] Photography digital still cameras JPEG 2000 DSC profile, ANSI/I3A IT10.2000-2004, 2000-2004. Available online: http://www.ansi.org.

[33] HD photo, photographic still image file format, Version 1.0, Microsoft Corporation, November 2006. Available online: http://www.microsoft.com/whdc/xps/wmphoto.mspx.

14

Modelling of Image Processing Pipelines in Single-Sensor Digital Cameras

Nai-Xiang Lian, Vitali Zagorodnov, and Yap-Peng Tan

14.1 Introduction

The main concerns in the design of digital still cameras (DSCs) are cost, size, image quality, and operational/power efficiency [1]. To reduce cost and size, most DSCs capture images using an electronic sensor overlaid with a color filter array (CFA), sampling one of three color primaries at each pixel location. The two missing color components are then estimated to restore full-color information [2], [3], [4], [5], [6]. The process is referred to as demosaicking [7], [8], [9], [10], [11]. For additional information on the fundamentals on single-sensor color imaging refer to Chapter 1.

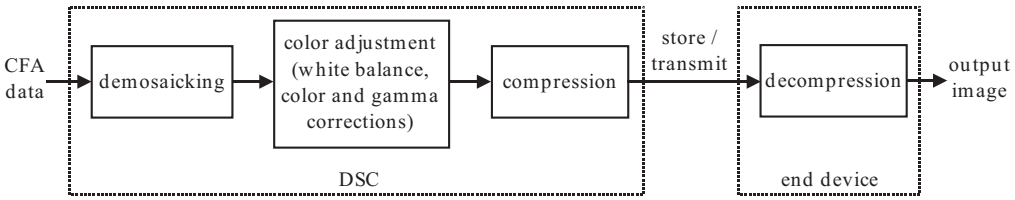


FIGURE 14.1

Block diagram of conventional processing chain.

Figure 14.1 shows the conventional DSC image processing pipeline. The CFA data from the sensor is first demosaicked to estimate full-color image. Then, the color values of the resultant image are modified by adjusting white balance, and performing color and gamma correction [12], [13], to match the colors of the original scene when displayed on a computer monitor. White balancing removes the color tint of an image to make white objects appear white. Color correction transforms the CFA sensor color space to a suitable display color space (e.g., sRGB [14]). Gamma correction adjusts the image intensity to compensate the non-linearity of cathode-ray-tube (CRT) displays. Some DSC models may also include noise reduction or image sharpening, but these processes are optional and can be deferred to a later stage if necessary. Finally, the image is compressed for storage or transmission [12], [13]. Detailed discussion on typical camera image processing pipelines can be found in Chapters 1 and 3.

The current development of DSC processing is typically guided by empirical performance evaluations [15], [16], [17], [18], [19], [20]. However, results provided by empirical evaluations are usually limited to a training set and are often inconclusive. For example, experiments may show one method to perform better on some types of images but worse on others, without explaining why this happens. The main goal of this chapter is to show the advantages of mathematical models that link the performance of the DSC processing to image content and algorithm settings. We also show how these models can be developed in practice through the examples of demosaicking, compression, color and gamma correction steps, as well as their interactions. Namely, we review the elements of DSC image processing pipelines in Section 14.2, and the alternative DSC processing chains in Section 14.3. Motivation for performance modelling is introduced in Section 14.4. Then, we show how to model the individual DSC processing elements and their interactions in Sections 14.5 and 14.6, respectively. Finally, an application example is presented in Section 14.7 and a summary of the chapter is given in Section 14.8.

14.2 Elements of Digital Still Camera Image Processing Pipeline

14.2.1 Demosaicking

Bilinear interpolation (or bilinear demosaicking) uses a linear function of two or four samples of the same color type located in the spatial proximity of the missing color

value [21]. Bicubic interpolation can improve the performance moderately at a substantial computational cost. However, the performance of these interpolation techniques can be rather poor, especially in regions rich in image details and edges.

Using anisotropic interpolation [5], [11], [22], [23] and exploiting spectral correlations [5], [7], [9], [10], [11] can yield additional performance improvements. Various ideas have been proposed here, such as interpolation along the edge direction (Hamilton method) [5] and substituting detail wavelet coefficients of green values into missing red/blue values by using an alternating projection (AP) method [9]. A more commonly used technique to incorporate spectral correlations is to take color ratios [7] or transform the image into a suitable color-difference space, as in the so-called effective color interpolation (ECI) [10] and improved effective color interpolation (IECI) [11] methods. Exploiting spectral correlation leads to incorporating samples from different color channels into an estimation of a missing color value. This requires higher computational complexity but produces better performance when compared to bilinear or bicubic methods that use the same color samples for prediction. Chapters 6 to 9 discuss demosaicking issues in more detail.

14.2.2 Color Adjustments

The appearance of scenes captured by DSCs depends on the color temperature of the light source. The goal of the white-balancing step is to adjust the colors of captured images to have a more natural appearance. In general, white balance can be achieved by multiplying the color channels with appropriate gain factors that depend on the color temperature of the light source.

Color correction aims to relate the sensor output values to colorimetry of the original scene [12]. Generally, the transformation depends on the digital camera, reflecting the differences in the sensor's spectral sensitivities as well as any nonlinear function used to encode the sensor output [12]. A typical example of color correction matrix for Canon EOS 300D is:

$$\mathbf{T} = \begin{bmatrix} 1.5915 & -0.6456 & 0.0541 \\ -0.0838 & 1.4794 & -0.3956 \\ 0.0697 & -0.4739 & 1.4042 \end{bmatrix} \tag{14.1}$$

Gamma correction is basically a power function of the form $ax^b + c$, where $b < 1$ and x is a color value in the range $[0,1]$. However, such a function has an infinite slope at zero, which may increase noise in dark regions. Hence, the function is usually modified with a linear portion near zero. A typical expression of gamma correction is [24]:

$$\gamma(x) = \begin{cases} 4.5x & \text{for } x \leq 0.018 \\ 1.099x^{0.45} - 0.099 & \text{for } x > 0.018 \end{cases} \tag{14.2}$$

14.2.3 Compression

Some cameras store the captured images in Tagged Image File Format for Electronic Photography (TIFF/EP) [25] format, which uses no or lossless compression. We can compress the image using standard compression techniques for DSC image storage since the

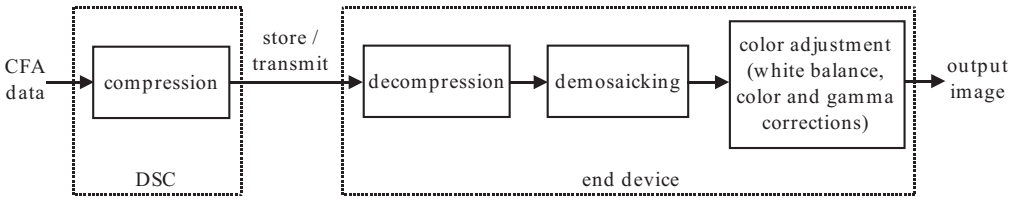


FIGURE 14.2

Block diagram of alternative processing chain.

demosaicking produces full color images. JPEG compression, which is based on discrete cosine transform [26], is used in Exchangeable Image Format (EXIF) storage format [27], a standard for most current DSCs [12]. Recently, wavelet-based compression (now formalized in the JPEG2000 compression standard [28]) has been shown to yield better performance than JPEG [29], by requiring fewer coefficients to represent image discontinuities. Because of its superiority over JPEG, JPEG2000 compression is likely to be incorporated into DSCs in the near future [12]. Detailed discussions on camera image storage formats can be found in Chapter 13.

14.3 Alternative Image Processing Pipelines

Demosaicking does not increase the information content (entropy) of the original image, so the additional pixels produced are mainly redundant [16], consuming substantial resources of the camera. To avoid this drawback, alternative processing pipelines often reverse the demosaicking and compression processes [15], [16], [17], [18], [19], [20]. Refer to Chapters 1 and 3 for detailed discussions on various processing configurations. Figure 14.2 depicts the case where the output of the CFA sensor is compressed directly before converting it to a full-color image. This removes the demosaicking and color adjustments (white balance, color correction, and gamma correction) from the camera to the end device.

Because the number of available color components in the CFA image is only 1/3 of that in the full-color image, this alternative processing chain requires less computational resources, storage capacity, and/or transmission bandwidth. The processing time (and hence the power consumption) is reduced by off-loading the demosaicking process from the camera to the end device, such as a personal computer (PC), personal digital assistant (PDA) or printer. This simplifies the hardware architecture and can reduce the cost and power consumption of DSCs, which is especially beneficial to cameras situated on mobile phones and PDAs [30]. Furthermore, recent work [15], [16], [17], [18], [19], [20] has indicated that the alternative processing chain can achieve better image quality than the conventional one under low compression ratios. It also allows different demosaicking methods to be applied to suit the needs of applications, for example, simpler demosaicking methods for a PDA image viewer, more advanced and complex demosaicking methods for high resolution PC displays [31].

All elements of the conventional processing chain can be reused in the alternative chain, except for compression, which has to be revised to work on CFA samples directly rather than on a full-color image. Several techniques have been proposed in the literature, such as applying standard grayscale compression techniques by discarding the pixel color labels [16], using Mallat packet wavelet transform and JPEG2000 compression to achieve better performance in the presence of artificial discontinuities [17], [32]. It has also been suggested to compress CFA color components separately [15], [16], [33], [34], which effectively discards the spectral correlations.

Color transformation is used in existing image compression standards to decorrelate the color components [26], [28]. A similar transformation can be used to remove artificial discontinuities in CFA images. For example, the popular Bayer CFA [35] consists of two green, one blue and one red sample arranged in a 2×2 square block. Therefore, Xie et al. [36] downsampled the green plane to the same size as the red and blue planes to simplify the use of color transformation for CFA mosaic images. Lee and Ortega [18], Parrein et al. [19], and Koh et al. [20] converted the four color values in each 2×2 Bayer unit to two luminance values (Y_1 and Y_2) and two chrominance values (U and V). The resultant chrominance values reside in a rectangular lattice with size four times smaller than the size of CFA, while the luminance values populate a quincunx lattice with half the size of CFA. The chrominance components (U or V) reside on a standard rectangular grid and hence can be compressed using any standard grayscale approach. For the quincunx lattice of the luminance plane, it has been suggested to either split it into two rectangular lattices Y_1 and Y_2, convolve it with a low-pass filter followed by columnwise downsampling [20], or rotate it by 45°, converting it into a diamond shaped rectangular lattice [18], [19]. For detailed information on CFA structure conversions and lossless compression of single-sensor mosaic images refer to Chapter 15.

14.4 Performance Modelling

14.4.1 Modelling Pipeline Elements

First, we must define what we mean by performance modelling. A performance model is a mathematical expression that links an algorithm's performance with a parametric description of the data content and/or algorithm's settings. For example, the data content parameters may include spatial (i.e., interpixel) or spectral (i.e., interchannel) correlations, noise variance; the possible algorithm's settings are bitrate, block size (for compression), and interpolation window size (for demosaicking). Availability of a mathematical model provides a better understanding of the algorithm as well as important cues to its application and further development.

The current research efforts in analysis of the DSC elements tend to use empirical performance evaluation. This, however, often produces inconclusive results where some algorithms perform better on some images and worse on others. For example, Table 14.1 shows the CPSNR (color peak signal-to-noise ratio) performance of three demosaicking methods for 24 test images shown in Figure 14.3. Namely, demosaicking methods under

FIGURE 14.3

Test images referred to as Image 1 to Image 24, enumerated from left-to-right and top-to-bottom.

TABLE 14.1
CPSNR performance (in dB) of three demosaicking methods. The best CPSNR of each row is shown in bold.

image	FDM	VOCD	AFD	image	FDM	VOCD	AFD
1	38.08	**38.53**	37.60	13	**35.18**	34.97	33.90
2	38.81	40.00	**40.30**	14	36.11	37.26	**37.39**
3	41.68	42.54	**42.77**	15	39.19	39.40	**39.54**
4	40.23	40.31	**40.40**	16	**43.75**	43.73	41.33
5	37.91	38.02	**38.12**	17	41.61	**41.61**	41.60
6	39.97	**40.01**	38.13	18	**36.74**	36.65	36.68
7	41.89	41.74	**42.74**	19	40.10	**40.83**	40.14
8	35.32	**36.43**	35.38	20	39.82	**40.38**	40.29
9	42.26	**43.18**	42.99	21	38.81	**39.26**	38.71
10	42.06	42.31	**42.65**	22	37.99	38.01	**38.55**
11	39.85	**39.95**	39.49	23	40.30	40.41	**41.09**
12	42.92	**43.03**	42.83	24	**35.13**	34.89	34.66

consideration are the frequency-domain method (FDM) [37], the variance of color difference (VOCD) method [38], and the adaptive filtering demosaicking (AFD) method [39]. Here CPSNR is defined as the average mean square error (MSE) of the reconstructed color components for RGB values normalized to [0,1]:

$$\text{CPSNR} = -10\log\left(\frac{1}{3}\sum_{c\in\{R,G,B\}}\text{MSE}_c\right) \tag{14.3}$$

Note that FDM performs the best for images 13, 16, 18, and 24; VOCD excels on images 1, 6, 8, 9, 11, 12, 17, and 19 to 21; and AFD is best for the remaining ten images. Without a mathematical performance model it is difficult to predict beforehand which method will achieve better performance on a given image.

TABLE 14.2
Performance comparison (SNR, in dB) of demosaicking on image data after and before gamma and color corrections (on average of test images).

method	bilinear	Hamilton [5]	AP [9]	IECI [11]
after	13.39	21.18	23.34	23.91
before	12.46	18.22	19.41	21.25

TABLE 14.3
Performance comparison (SNR, in dB) of compression on image data after and before gamma and color corrections (on average of test images).

compression ratio		5:1	10:1	15:1	30:1
JPEG	after	27.13	22.58	19.86	15.78
	before	21.30	18.41	16.65	13.75
JPEG2000	after	31.47	26.23	23.22	18.79
	before	24.57	21.04	18.99	15.82

Another example of the use of performance models comes from our recent work on error inhomogeneity in wavelet-based compression [40]. In this work we extend the well-known Kato-Yatsuda compression error model [41] to design a balanced wavelet, which sacrifices some compression performance for reduction in error inhomogeneity. The same compression model has also been used in many recent publications [42], [43], [44], [45], [46], [47] on wavelet evaluation and design for compression.

14.4.2 Modelling Interactions

Interactions between various elements of the DSC pipeline are rarely mentioned in the literature. Their existence and importance can be demonstrated using the following simple experiments.

Table 14.2 and Table 14.3 show a surprising result that applying demosaicking and compression after gamma and color corrections[1] can improve SNR performance by one to four dB for demosaicking and up to seven dB for compression at high bitrates, compared with the opposite order. Here SNR is defined as the ratio between the signal variance σ_I^2 and average MSE of the reconstructed image:

$$\mathrm{SNR} = 10\log \frac{\sigma_I^2}{\frac{1}{3}\sum_{c\in\{R,G,B\}} \mathrm{MSE}_c} \tag{14.4}$$

If there were no interaction between these elements, we would expect to see exactly the same performance regardless of the order.

[1]Here we adopt the color correction matrix from Equation 14.1 used in the Canon EOS300D camera and gamma correction defined as in Equation 14.2 according to Rec.709 [24]. Signal-to-noise ratios (SNRs) are computed on the gamma and color-corrected data.

FIGURE 14.4 (See color insert.)

Cropped region of Window image: (a) after bilinear demosaicking; (b) after bilinear demosaicking and JPEG (10:1) compression.

TABLE 14.4
CPSNR performance (in dB) of the conventional processing chain with bilinear demosaicking and JPEG/JPEG2000 compression.

	after	after demosaicking and compression			
image	demosaicking	JPEG(10:1)	JPEG(30:1)	JPEG2000(10:1)	JPEG2000(30:1)
window	32.37	33.86	32.40	32.08	31.71
lighthouse	28.01	30.04	29.17	27.87	27.65
statue	31.91	33.66	32.41	31.67	31.27
sails	31.69	33.59	32.97	31.58	31.38

That compression and demosaicking also interact with each other follows from the fact that applying JPEG compression after bilinear demosaicking leads to overall error reduction, as shown in Table 14.4 and illustrated in Figure 14.4. This would not happen if the compression and demosaicking errors were independent, since lossy compression always leads to decrease in image quality. Moreover, this interaction may depend on the demosaicking and compression methods used. For example, the CPSNR improvement after compression does not happen if we use JPEG2000 compression; see Table 14.4.

The importance of modelling interactions follows from a recent survey of publications on the comparison between conventional and alternative chains [18], [20], [19]. One of the goals of these studies was to provide support for viability of alternative processing chains by discovering conditions under which alternative processing chains have their performance comparable to or better than that of the conventional processing chain. It appears that the main factors affecting the relative performance of the two chains are the compression ratio and the choice of compression algorithm. For example, Koh et al. [20] used JPEG compression in their alternative processing chain and found that it can outperform the conventional one at low (10:1) and high (80:1) compression ratios. Lee and Ortega [18] observed superiority of alternative processing with JPEG compression at low compression ratios. Parrein et al. [19] used JPEG2000 and reached the same conclusion for compression ratios 15:1. All of these conclusions are based on empirical studies, making their results relevant to only existing compression and demosaicking algorithms. Without a mathematical performance model it is difficult to explain why the alternative processing chain is better than conventional ones at low compression ratios, or predict the break-point compression ratio (where the relative performance changes sign) for a given pair of compression and demosaicking algorithms. Furthermore, it is also unclear which chain will be superior when more advanced demosaicking and compression algorithms become available in the future.

14.4.3 Modelling via Taylor Series Expansion

Demosaicking and compression methods usually exploit spatial and spectral image correlations. In general, lower correlations yield larger reconstruction and compression errors, while close to 1 correlations yield smaller errors. For perfect spatial and spectral correlations $\rho = 1$, the error should be zero. This suggests the following performance model, where the error variance $e(\rho)$ is treated as a function of correlation and is expanded in terms of Taylor series near $\rho = 1$:

$$e(\rho) = \sum_{k=1}^{\infty} e^{(k)}(1) \frac{(\rho-1)^k}{k!} \sigma_I^2 \tag{14.5}$$

where σ_I^2 is the signal variance for the color values x. In reality, an image exhibits different types of correlations (e.g., spatial correlation in horizontal and diagonal direction, and spectral correlation) and hence the error variance should be treated as a function of several variables. The following expression shows a model example where we preserved only linear terms and included spatial correlations in various directions $\rho_{i,j}$ and spectral correlations ν_{S_k,S_l}:

$$e(\rho,\nu) = \left[\sum_{i,j} e'_{i,j}(1)(\rho_{i,j}-1) + \sum_{S_k \neq S_l} e'_{S_k,S_l}(1)(\nu_{S_k,S_l}-1) \right] \sigma_I^2 \tag{14.6}$$

Here S_l and S_k, both from $\{R,G,B\}$, designate the color components. To complete the modelling, the unknown coefficients are estimated empirically.

Taylor series expansion can also be used to model interactions. Let D and F be two consecutive DSC processing elements. Let Δx be the error of processing element D, $D(x) = x + \Delta x$, and $\zeta^2 \triangleq E(\Delta^2 x)$ be its variance. If the the processing element F does not produce

errors, the total error is $F[D(x)] - F(x) = F(x_i + \Delta x) - F(x)$. The variance of the total error is a monotonic function of ζ^2, $f(\zeta^2) = E\{[F(x+\Delta x) - F(x)]^2\}$, such that $f(0) = 0$. Assuming that ζ^2 is small, we can expand $f(\zeta^2)$ using a Taylor series, preserving only the linear term:

$$f(\zeta^2) = E\{[F(x+\Delta x) - F(x)]^2\} \approx c^2\zeta^2 \tag{14.7}$$

where (in general) parameter $c^2 = E\{[\frac{\partial F(x)}{\partial x}]^2\}$ can depend on F.

If the processing element F produces an error with variance ξ^2, the total error becomes

$$F[D(x)] - x = F[D(x)] - F(x) + F(x) - x \tag{14.8}$$

The total error variance can be shown as [48],

$$E\left\{\{F[D(x)] - x\}^2\right\} = c^2\zeta^2 + \xi^2 + 2c\rho_c\zeta\xi \tag{14.9}$$

where ρ_c is the correlation between two processing errors.

In the following sections we provide several practical examples of modelling the DSC processing elements (Section 14.5) and their interactions (Section 14.6).

14.5 Modelling of Individual Digital Still Camera Processing Elements

In DSC processing, color adjustments (white balance, color and gamma corrections) transfer images from one space to another, which does not produce their own errors. Hence, in this section we concentrate on modelling the demosaicking and compression errors.

14.5.1 Modelling of Demosaicking Errors

Demosaicking methods use pixels in a local neighborhood for interpolation. Hence we restrict Equation 14.6 to correlations corresponding to $|i| + |j| \leq 1$. Furthermore, to reduce the number of parameters, we assume e'_{S_k,S_l} to be the same for any $S_k \neq S_l$. As non-coinciding pixels in different color components are used for demosaicking, we add a new term corresponding to the spectral correlations of non-coinciding pixels $\nu_{S_k,S_l}\rho_0$. The resultant model then has the form of

$$\zeta^2 = \{\chi_0 + \chi_1\rho_0 + \chi_2 \sum_{S_k \neq S_l} \nu_{S_k,S_l} + \chi_3 \sum_{S_k \neq S_l} \nu_{S_k,S_l}\rho_0\}\sigma_I^2 = \chi' \mathscr{P} \sigma_I^2 \tag{14.10}$$

where the corresponding sums $e'_{i,j}$ and e'_{S_k,S_l} are now abbreviated as coefficients χ_i, and $\mathscr{P}$ is the vector of image correlations.

To estimate the unknown parameters for each demosaicking method we fit the actual demosaicking errors from 15 test images shown in Figure 14.3 to the model as shown in Equation 14.10. Table 14.5 shows the modelling misfit, defined via the ratio between the predicted and the actual demosaicking errors $|1 - \frac{\zeta^2_{predicted}}{\zeta^2_{actual}}|$. To check the validity of our model we also applied it to four test images that were excluded from the training set; see the bottom four rows of Table 14.5. As we can see, the misfit is less than 10% in most cases, which implies the sufficient accuracy of the linear model.

TABLE 14.5
Modelling misfit of demosaicking error variance.

method	bilinear	Freeman	Hamilton	ECI	AP	ECI3
prediction misfit for training images	0.5%	2.7%	9.6%	2.1%	9.4%	13.0%
test image	modelling misfit					
16	4.5%	14.1%	2.6%	14.3%	0.2%	1.2%
17	2.4%	0.1%	17.5%	0.0%	1.2%	1.9%
18	0.4%	0.2%	12.8%	0.0%	8.6%	15.3%
19	0.0%	1.2%	19.9%	0.7%	5.1%	9.5%

14.5.2 Modelling of Compression Errors

Unlike demosaicking, the model for compression error needs to link the performance with both the image content (correlations) and compression bitrate. The previously discussed method based on Taylor series expansion (Equation 14.6) is effective in capturing the former but not the latter. Most existing compression models are based on the work of Katto and Yasuda [41] on modelling the compression error variance ξ_I^2 using K-subbands decomposition[2] and optimal bitrate allocation, given as

$$\xi_I^2 = \varepsilon^2 \sigma_I^2 2^{-2R_I} \cdot \prod_{k=0}^{K-1} (A_k B_k)^{\alpha_k} \tag{14.11}$$

where R_I is the total compression bitrate, α_k is the relative proportion of the number of kth subband wavelet coefficients, and ε is a constant depending on the signal. The term A_k represents the relative proportion of signal energy in the kth subband, that is, $\sigma_k^2 = A_k \sigma_I^2$. The term B_k governs the contribution of the kth subband quantization error variance σ_{qk}^2 to the total quantization error, that is, $\xi_I^2 = \sum_{k=0}^{K-1} B_k \sigma_{qk}^2$. For one-dimensional (1D) signals modelled using Markov chains, A_k and B_k are related to the analysis h_k and synthesis g_k filters and spatial correlation [42], [43] as follows:

$$A_k = \sum_i \sum_j h_k(i) h_k(j) \rho_{|i-j|} \tag{14.12}$$

$$B_k = \sum_i g_k^2(i) \tag{14.13}$$

Since the filters are known by construction, the resultant model depends only on bitrate R_I, spatial correlations ρ_i, and signal variance σ_I^2. The model presented in Equation 14.11 can be extended to images modelled by two-dimensional (2D) Markov random processes [42], [43], [44], [45], [46], [47].

[2]The theory can be straightforwardly extended to JPEG compression by grouping DCT coefficients and treating them as subbands. For example, in JPEG compression all zero frequency coefficients are assigned the same bitrate, and can be considered as one subband.

It is important to note that the product $A_k B_k$ in Equation 14.11 is closely related to the coding gain [42], [43] defined as

$$\mathscr{G}_I = \frac{1}{\prod_{k=0}^{K-1}(A_k B_k)^{\alpha_k}} \tag{14.14}$$

The derivation of Equation 14.11 is based on modelling the kth subband quantization error as $\sigma_{qk}^2 = \varepsilon^2 \sigma_k^2 2^{-2R_k}$ [49], where R_k is the subband bitrate. This quantization model is valid only at high bitrates R_k, [50]. We can generalize the model in Equation 14.11 for lower bitrates by allowing β to be a function of bitrate [48]:

$$\xi_I^2 = \varepsilon^2 \sigma_I^2 2^{-2\beta(R_I)R_I} \mathscr{G}_I^{-1} \tag{14.15}$$

Based on the generalized error model from Equation 14.15 it is possible to derive the compression error models for full-color and CFA images.

14.5.2.1 Full-Color Image Compression

Components of color images (R, G, and B) can be compressed independently by treating each component as a grayscale image. Let $\mathscr{C} \in \{R,G,B\}$ be an arbitrary color component. Using the error model (14.15), the compression error of $\mathscr{C}$ is given by $\xi_c^2 = \varepsilon^2 \sigma_c^2 2^{-2\beta(R_c)R_c} \mathscr{G}_I^{-1}$. Here σ_c^2 and R_c are the pixel variance and the bitrate assigned to component $\mathscr{C}$, respectively, and $\sum_{\mathscr{C}} R_c = R$. Hence, the average compression error of all color components is

$$\xi_s^2 = \frac{1}{3}\varepsilon^2 \mathscr{G}_I^{-1} \sum_{\mathscr{C}} \sigma_c^2 2^{-2\beta(R_c)R_c} \tag{14.16}$$

A color image is typically compressed in an appropriate color space, such as YUV or YCbCr space [26], [28]. We use the error model from Equation 14.15 by treating the color transformation T_s as a 3-subband decomposition [48]. Then the analysis and synthesis filters h_k and g_k are rows of T_s and T_s^{-1}, respectively, and the spatial correlations in Equation 14.15 are replaced by spectral correlations. Furthermore, parameters $\mathscr{A}_c$ and $\mathscr{B}_c$ for $\mathscr{C} \in \{Y,U,V\}$ can be derived from Equations 14.12 and 14.13. As detailed in Reference [48], the overall compression error becomes

$$\begin{aligned}\xi_s^2 &= \varepsilon^2 \sigma_I^2 2^{-2\beta(\frac{R}{3})\frac{R}{3}} \prod_{\mathscr{C}\in\{Y,U,V\}} (\mathscr{A}_c \mathscr{B}_c)^{\frac{1}{3}} \prod_{k=0}^{K-1} (A_k B_k)^{\alpha_k} \\ &= \varepsilon^2 \sigma_I^2 2^{-2\beta(\frac{R}{3})\frac{R}{3}} \mathscr{G}_s^{-1}\end{aligned} \tag{14.17}$$

where $\mathscr{G}_s$ is the new coding gain.

14.5.2.2 CFA Image Compression

Without color transformation, the model for CFA compression is similar to that for the grayscale image compression as shown in (14.15). We obtain [48]:

$$\begin{aligned}\xi_o^2 &= \varepsilon^2 \sigma_I^2 2^{-2\beta(R)R} \prod_{k=0}^{K-1} (A_{ok} B_k)^{\alpha_k} \\ &= \varepsilon^2 \sigma_I^2 2^{-2\beta(R)R} \mathscr{G}_o^{-1}\end{aligned} \tag{14.18}$$

The only difference is the new expression for the coding gain $\mathscr{G}_o$, which is computed according to the new spatial correlations.

Performance can be improved by using a luminance-color difference transformation. The red (R), blue (B), and two green samples (G_1 and G_2) in a 2×2 Bayer unit are converted into Y_1, Y_2, U, and V samples using a linear transformation defined by a 4×4 matrix. With this transformation, the compression error variance becomes [48]:

$$\begin{aligned}\xi_t^2 &= \varepsilon^2 \sigma_I^2 2^{-2\beta(R)R} \prod_{\mathscr{C} \in \{Y_1, Y_2, U, V\}} \left(\mathscr{A}_c \mathscr{B}_c \prod_{k=0}^{K-1} (A_{ck} B_k)^{\alpha_k} \right)^{\frac{1}{4}} \\ &= \varepsilon^2 \sigma_I^2 2^{-2\beta(R)R} \mathscr{G}_t^{-1} \end{aligned} \tag{14.19}$$

where $\mathscr{G}_t$ is the corresponding coding gain.

The correlated components Y_1 and Y_2 can be more efficiently compressed as one image Y, [18], [19], [20]. It has also been suggested to rotate Y by 45°, converting a quincunx lattice to a rectangular lattice with diamond-shaped support [18], [19]. Equation 14.19 is still applicable if we recalculate parameter A_{ck} based on the spatial correlations in the horizontal direction in a quincunx lattice.

14.5.2.3 Evaluation of the Compression Error Models

It should be noted that compression error models can accurately predict the relative performance of two compression algorithms on the same data but not the absolute value of the error [48]. Here we demonstrate this on an example comparison between full-color and CFA image compression performances. From Equations 14.17 and 14.18, we obtain

$$\frac{\xi_s^2}{\xi_o^2} = 2^{2\beta_r(R)R} \frac{\mathscr{G}_o}{\mathscr{G}_s} \tag{14.20}$$

where $\beta_r(R) = \beta(R) - \frac{\beta(\frac{R}{3})}{3}$, while $\mathscr{G}_s$ and $\mathscr{G}_o$ are the respective coding gains. The ratio $\frac{\xi_s^2}{\xi_o^2}$ depends on the ratio of the coding gains $\frac{\mathscr{G}_o}{\mathscr{G}_s}$ and the bitrate via $2^{2\beta_r(R)R}$. Full-color images have higher spatial and spectral correlations compared with the corresponding CFA images, leading to larger coding gains $\mathscr{G}_s > \mathscr{G}_o$. However, full-color images also have three times as many pixel values as the CFA images. This is captured by the term $2^{2\beta_r(R)R}$. At high bitrate, $\beta(R) = 1$ and then $\beta_r(R) = 1 - \frac{1}{3} = \frac{2}{3}$. Hence this term $2^{2\beta_r(R)R} \gg 1$ dominates at high bitrates leading to $\frac{\xi_s^2}{\xi_o^2} = 2^{2\beta_r(R)R} \frac{\mathscr{G}_o}{\mathscr{G}_s} > 1$ and brings about the superiority of the alternative processing chain at the low compression ratio.

Experimental results are generally consistent with our analysis. In particular, the results show that the log-ratio is approximately linear for bitrate $R > 1.5$ bpp (bits per pixel), and the break-point bitrate, corresponding to $\frac{\xi_s^2}{\xi_o^2} = 1$, generally occurs in the linear portion of the curves. Hence, we can predict the break-point bitrate by treating $\beta_r(R)$ in Equation 14.20 as a constant, which can be estimated by fitting Equation 14.20 to actual data at high bitrates. Our predicted break-point for JPEG2000 is $R_b = 2.2$ bpp, which practically coincides with the actual value. For JPEG, the predicted value is $R_b = 3.6$ bpp, which is slightly smaller than the actual $R_b = 3.8$ bpp.

14.6 Modelling of Interactions

14.6.1 Color Adjustments and Demosaicking / Compression

Since color and gamma corrections do not produce their own errors, we model their interaction with demosaicking/compression based on the result in Equation 14.7. If demosaicking precedes color and gamma correction, the total error variance is

$$\Upsilon^2 = c_f^2 \zeta_0^2 \tag{14.21}$$

where c_f^2 depends on the derivative of the color adjustment function. When color and gamma corrections precede demosaicking, $\Upsilon^2 = \zeta_1^2$, where $\zeta_0^2 \approx \zeta_1^2$ since the total strength of spatial and spectral correlations is not substantially affected by color and gamma corrections.

Since color correction is a linear operation, its derivative does not depend on image content. Assuming that the errors of the preceding operation have equal variance and are independent in the three color channels, we have,

$$c_f^2 = \frac{1}{3} Tr(\mathbf{T}\mathbf{T}'), \tag{14.22}$$

where Tr is the trace of the matrix. For the Canon EOS300D camera, $c_f^2 = 2.5$. Other cameras have c_f^2 around 2. This means that color correction could enlarge the errors of the preceding operations, such as demosaicking and compression, conforming to our earlier observation in Table 14.2 and Table 14.3.

The derivative of gamma correction, Equation 14.2, can be calculated as

$$\frac{d\gamma}{dx}(x) = \begin{cases} 4.5 & x \leq 0.018 \\ 0.4945 x^{-0.55} & x > 0.018 \end{cases} \tag{14.23}$$

Then we can obtain the value of c_f^2 using the distribution of image color values $p(x)$,

$$c_f^2 = \int_0^1 \frac{d\gamma}{dx}(x)^2 p(x) dx \tag{14.24}$$

We plot the derivative $\frac{d\gamma}{dx}(x)$ as a function of image intensity in Figure 14.5. The curve intersects the line $\frac{d\gamma}{dx} = 1$ at point $x = 0.278$, indicating that gamma correction enlarges the error for $x < 0.278$ (darker regions) and reduces the error for $x > 0.278$ (brighter regions). Hence, darker images will have larger c_f^2 and brighter images will have smaller c_f^2. For example, $c_f^2 = 1.32$ and $c_f^2 = 11.13$ for test images 12 and 17, whose distribution of color values are shown in Figure 14.6a and Figure 14.6b, respectively. Note that $c_f^2 > 1$ even in the brighter image, which is due to much larger multiplication factors in darker regions. While it is theoretically possible to have $c_f^2 < 1$ for very bright images, for the overwhelming majority of cases $c_f^2 > 1$ and hence demosaicking and compression can achieve better performance on gamma-corrected images. This agrees with our earlier observations in Table 14.2 and Table 14.3.

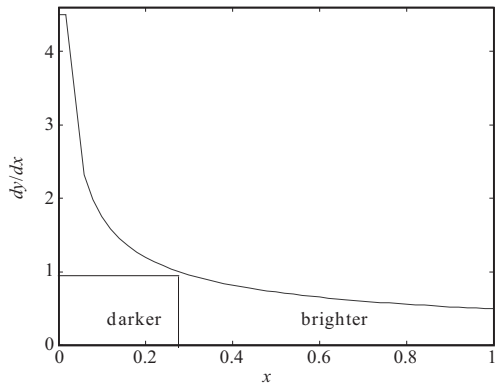


FIGURE 14.5

Relationship between intensity x and derivative value ($\frac{d\gamma}{dx}$) for gamma correction.

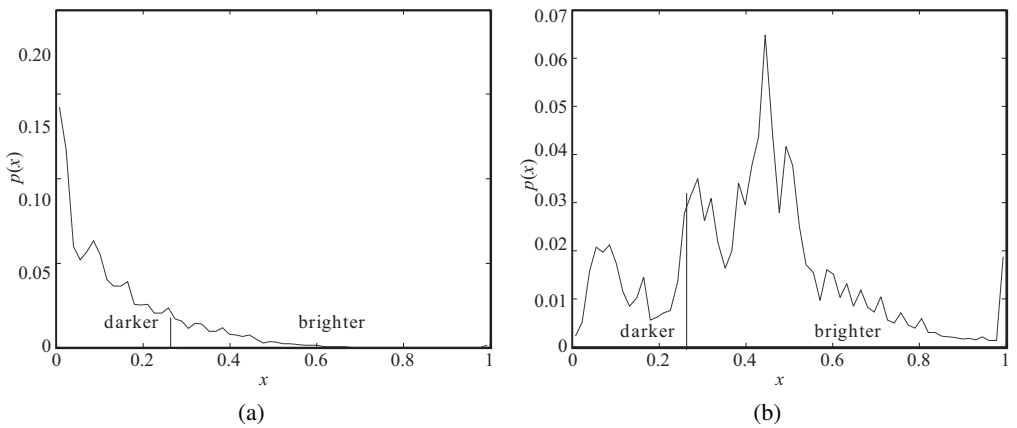


FIGURE 14.6

The distribution of color values for (a) test image 12 and (b) test image 17.

14.6.2 Demosaicking and Compression

The influence of interaction between demosaicking and compression on the overall performance can be modelled using a Taylor series expansion approach as shown in (14.9). If demosaicking precedes compression, the error is [48]:

$$\Omega^2 = \xi_s^2 + a_c^2\zeta^2 + 2\rho_a a_c \xi_s \zeta \approx \xi_s^2 + a_c^2\zeta^2 \tag{14.25}$$

The cross term can be discarded due to small correlation ρ_a between demosaicking and compression errors. Empirical tests show that $a_c^2 \approx 0.5$ for JPEG compression and $a_c^2 = 1$ for JPEG2000. Figure 14.7 shows that the model in Equation 14.25 provides an excellent fit to the experimental results.

The values of coefficient a_c^2 can be justified as follows. Demosaicking errors mostly occur near edges and hence contain a lot of high frequency information. JPEG compression is poor at preserving these high frequencies and hence makes the error variance of the preceding demosaicking step smaller ($a_c^2 < 1$). On the other hand, JPEG2000 can better

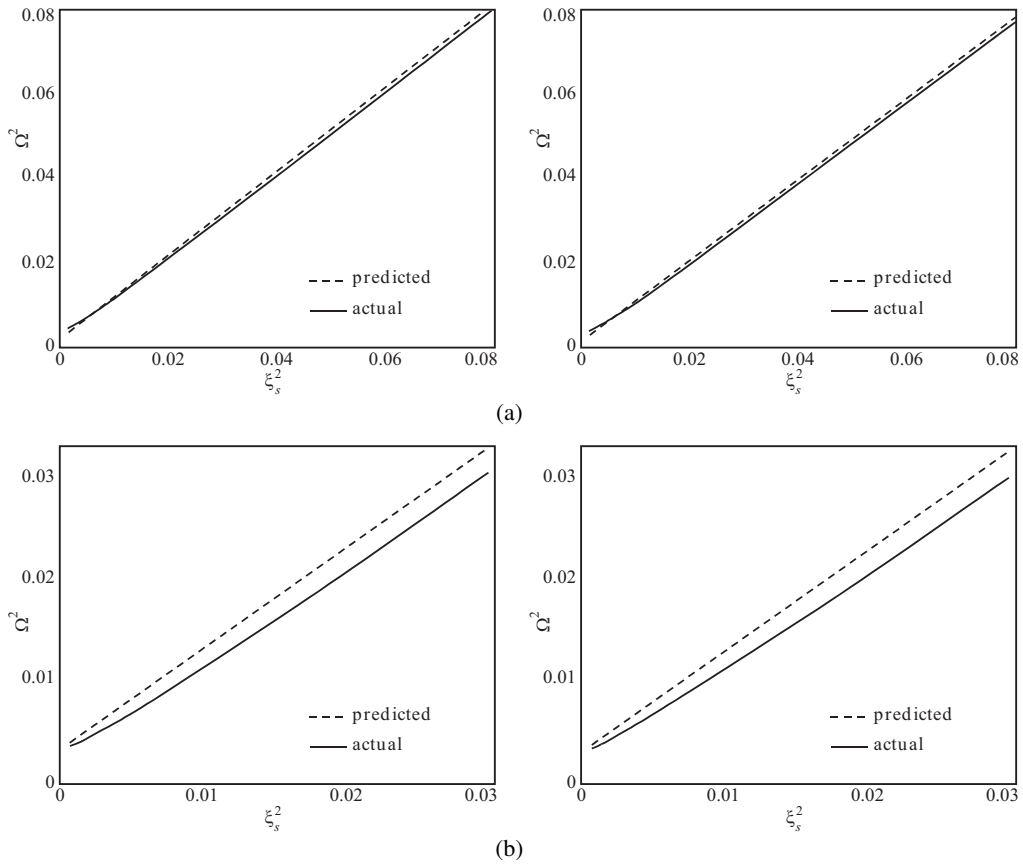


FIGURE 14.7

The error model from Equation 14.25 for demosaicking followed by (a) JPEG and (b) JPEG2000 compression. Used demosaicking methods: (left) Hamilton, and (right) AP.

preserve high-frequency content, leading to $a_c^2 \approx 1$. However, when the compression ratio is very large, both techniques will suppress high frequencies and hence $a_c^2 \to 0$.

The model in Equation 14.25 can also explain the phenomenon in Table 14.4, that the JPEG compression following bilinear demosaicking reduced the total error. For such a combination, $a_c^2 \approx 0.5$ and ζ^2 is large (due to poor performance of bilinear demosaicking). Hence it is possible that

$$\Omega^2 \approx \xi_s^2 + 0.5\zeta^2 < \zeta^2 \tag{14.26}$$

at least when the compression error is small (low compression ratios). The same phenomenon does not appear in JPEG2000 even at low compression ratios (Table 14.4), since $a_c^2 \approx 1$, [48].

If compression precedes demosaicking, then from Equation 14.9 we have [48]:

$$\Psi^2 = \zeta^2 + b_d^2 \xi_o^2 + 2\rho_b b_d \zeta \xi_o \approx \zeta^2 + \xi_o^2 \tag{14.27}$$

The cross term can be discarded due to small correlation ρ_b. It also follows that parameter $b_d^2 \approx 1$ for all tested combinations of demosaicking and compression. Figure 14.8 shows the excellent fit of the model in Equation 14.27 to the experimental results.

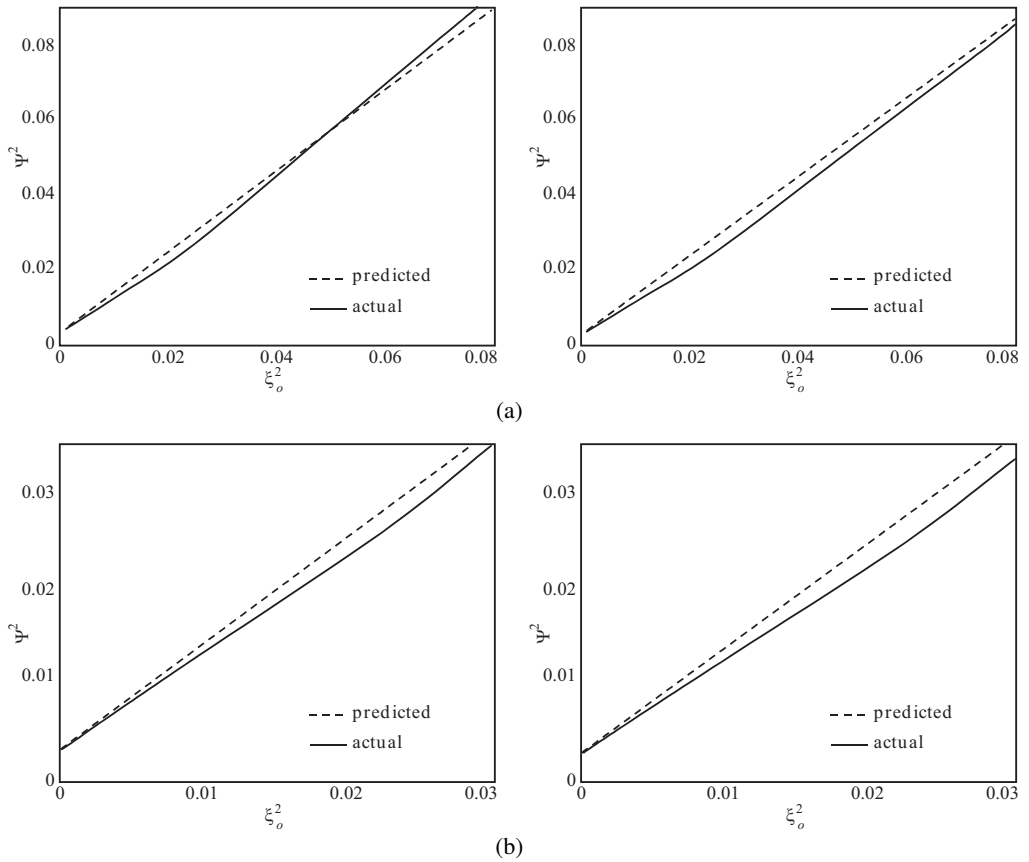


FIGURE 14.8

The error model from Equation 14.27 for (a) JPEG and (b) JPEG2000 compression followed by demosaicking. Used demosaicking methods: (left) Hamilton, and (right) AP.

14.7 Performance Evaluation of Digital Still Camera Processing Pipelines

In this section we show how the earlier results on modelling of DSC processing elements (Section 14.5) and interactions (Section 14.6) can be applied to a practical problem of comparison between conventional and alternative processing pipelines. The need for such a comparison arises because the alternative chain promises to offload some of the processing operations, such as demosaicking and color adjustments, from the camera to the end device, thus simplifying the DSC and making it more cost and power efficient. However, doing so will be justifiable only if alternative processing does not degrade image quality.

For the sake of simplicity we assume that gamma correction is applied directly to CFA data prior to demosaicking and compression, and that color correction can be neglected for both the conventional and alternative chains. Hence, for comparison we can consider the simplified processing pipelines shown in Figure 14.9.

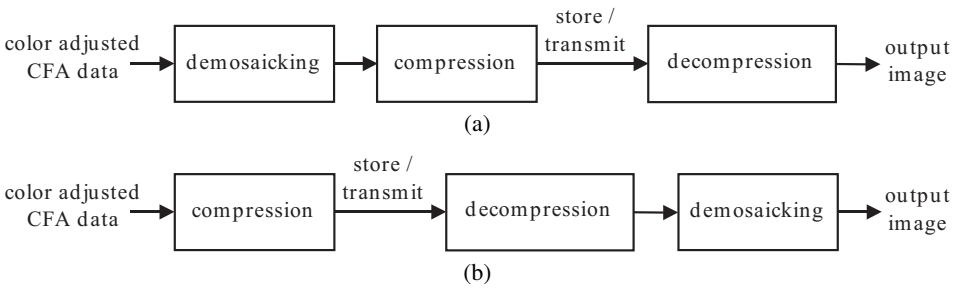


FIGURE 14.9

Simplified block diagrams of DSC image processing chains: (a) conventional, and (b) alternative.

Based on the models in Equations 14.25 and 14.27, we obtain the following expression for the difference of total error variances between the two chains:

$$\Omega^2 - \Psi^2 = (\xi_s^2 - \xi_o^2) + (a_c^2 - 1)\zeta^2 \tag{14.28}$$

At high compression ratios, the compression error is larger than the demosaicking error and the relative performance depends mainly on $\xi_s^2 - \xi_o^2$. At low compression ratios, even though $(\xi_s^2 - \xi_o^2) < \zeta^2$, both compression and demosaicking error terms should be preserved since a_c^2 can be close to one [48], leading to cancellation of the demosaicking error term.

The results of qualitative analysis of Equation 14.28 are summarized in Table 14.6. Here we split the demosaicking approaches into simple (e.g., bilinear - large error) and advanced (majority of recently proposed methods - small error). As follows from the table, the alternative chain can outperform the conventional one at low compression ratios, except when simple demosaicking is combined with JPEG compression.

Qualitative predictions in Table 14.6 agree very well with our experimental results shown in Figure 14.10 and Figure 14.11. In these figures we compare the signal-to-noise ratio (SNR) performance of the conventional and alternative processing chains, where the latter applied a simple grayscale compression for CFA images. As expected, the alternative processing chain outperforms the conventional one at low compression ratios (high bitrates) for JPEG2000 compression. The same is true for JPEG compression combined with the AP demosaicking method. However, in the combination of JPEG compression and bilinear demosaicking, the conventional chain performs better for all tested compression ratios, as shown in Figure 14.10 and Figure 14.11.

TABLE 14.6

Superiority of processing chains (C - conventional, A - alternative).

compression	JPEG		JPEG2000
bitrates	simple demosaicking	advanced demosaicking	
high	C	A	A
low	C	C	C

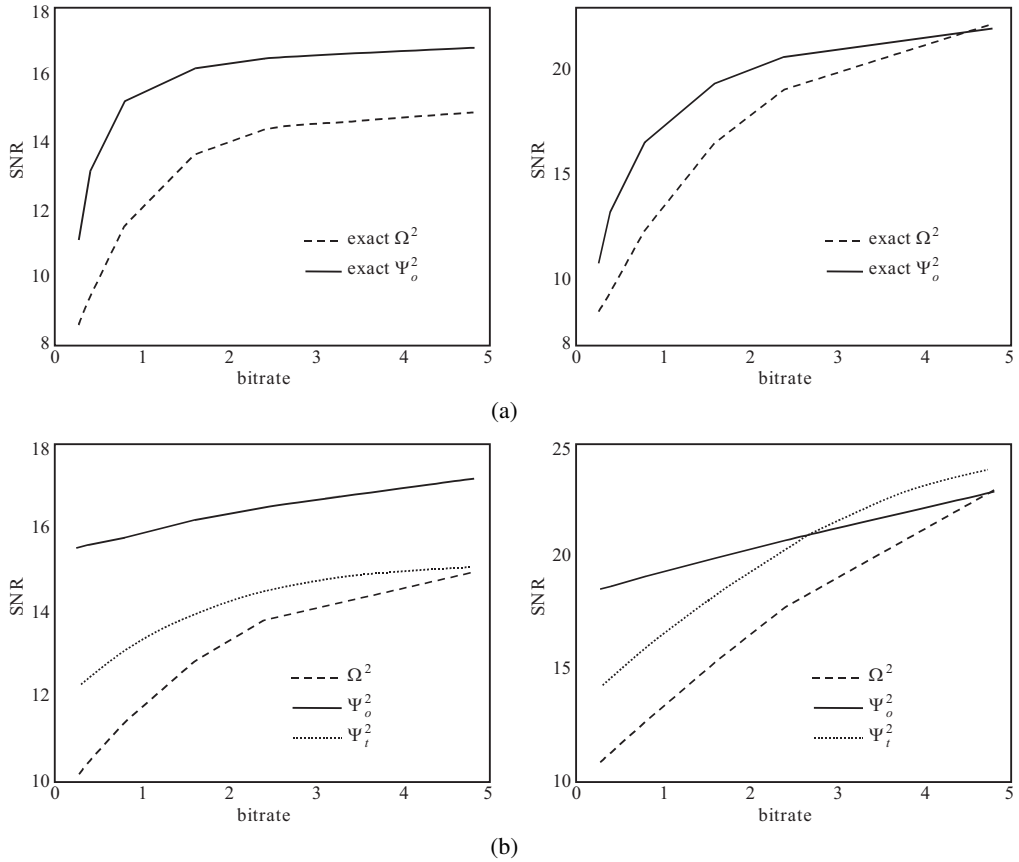


FIGURE 14.10

Performance comparison between conventional and alternative processing chains using JPEG compression: (a) actual and (b) predicted performance; (left) bilinear and (right) AP demosaicking.

The relative performance model in Equation 14.28 can also provide precise prediction of the break-point compression ratio t_b (or a break-point bitrate $R_b = \frac{24}{t_b}$) at which the processing chains have equal performance [48]. As shown in Figure 14.10, the actual break-point bitrate for JPEG-AP combination is $R_b \approx 4.5$ bpp, while predicted $R_b \approx 4.0$ bpp. For JPEG2000, both experimental and predicted results yield $R_b \approx 2.5$ bpp (Figure 14.11).

Prediction results can be extended to more advanced (color transformation based) CFA image compression methods by using Equation 14.19. The corresponding curves are shown in Figure 14.10 and Figure 14.11 (designated as Ψ_t^2) and indicate that the break-points occur at lower bitrates. This happens because color transformation removes discontinuities in CFA images caused by interlaced color components, leading to a larger coding gain $\mathscr{G}_t > \mathscr{G}_o$. As the image size remains unchanged, the larger coding gain leads to smaller compression error $\xi_t^2 < \xi_o^2$, moving the break-points to lower bitrates [48].

Figure 14.12 shows an example obtained by using a CFA image from an actual DSC Canon EOS300D. Besides some blurring effect due to the use of a simple (bilinear) demosaicking method, the reconstructed images obtained by the two processing chains have similar quality under a typical DSC compression ratio 10:1.

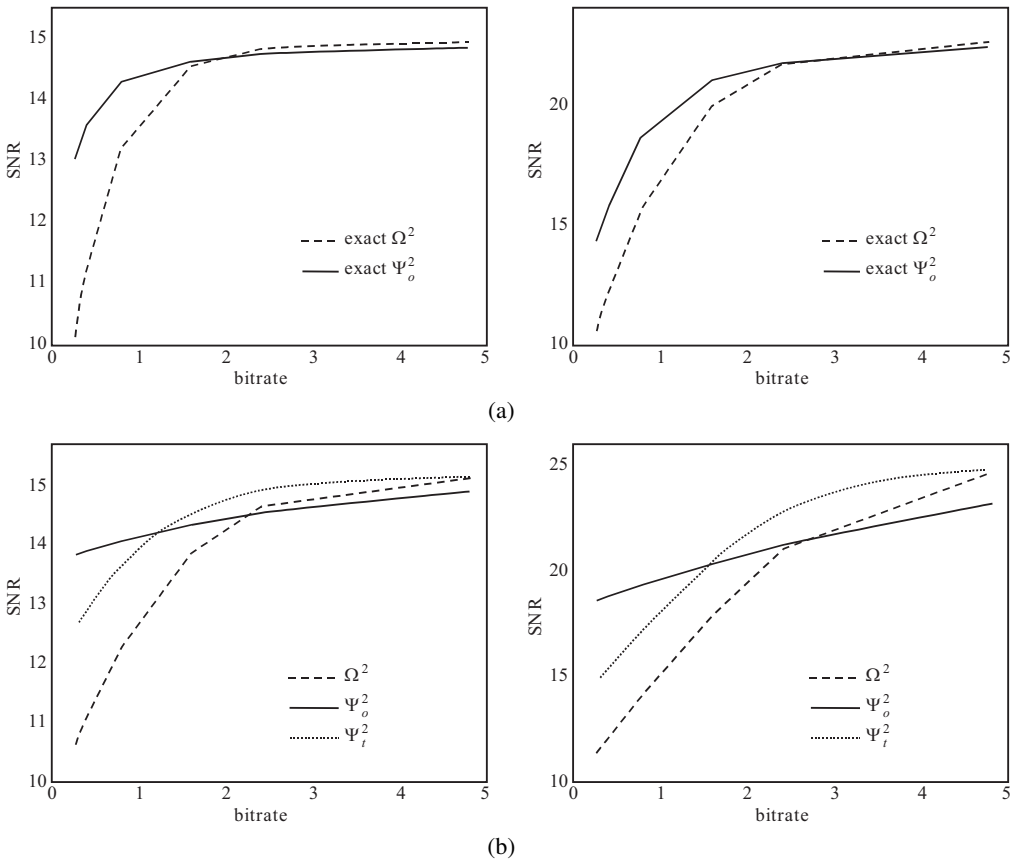


FIGURE 14.11

Performance comparison between conventional and alternative processing chains using JPEG-2000 compression: (a) actual and (b) predicted performance; (left) bilinear and (right) AP demosaicking.

The analysis suggests a bright future for alternative processing chains. With further development of compression methods the break-point bitrate can be expected to shift to even lower values, covering the most popular compression setting used in current DSCs. Note that such predictions can only be made on the basis of the model developed and are difficult to derive from empirical performance evaluations.

14.8 Summary

In this chapter we mathematically modelled behaviors of DSC processing elements and their interactions. The main goal of such a modelling is to optimize the pipeline's performance by picking suitable processing elements and their arrangements. For example, color and gamma corrections lead to smaller error when placed before demosaicking. Reversing the order of demosaicking and compression can simplify camera design and can even improve performance for low compression ratios.

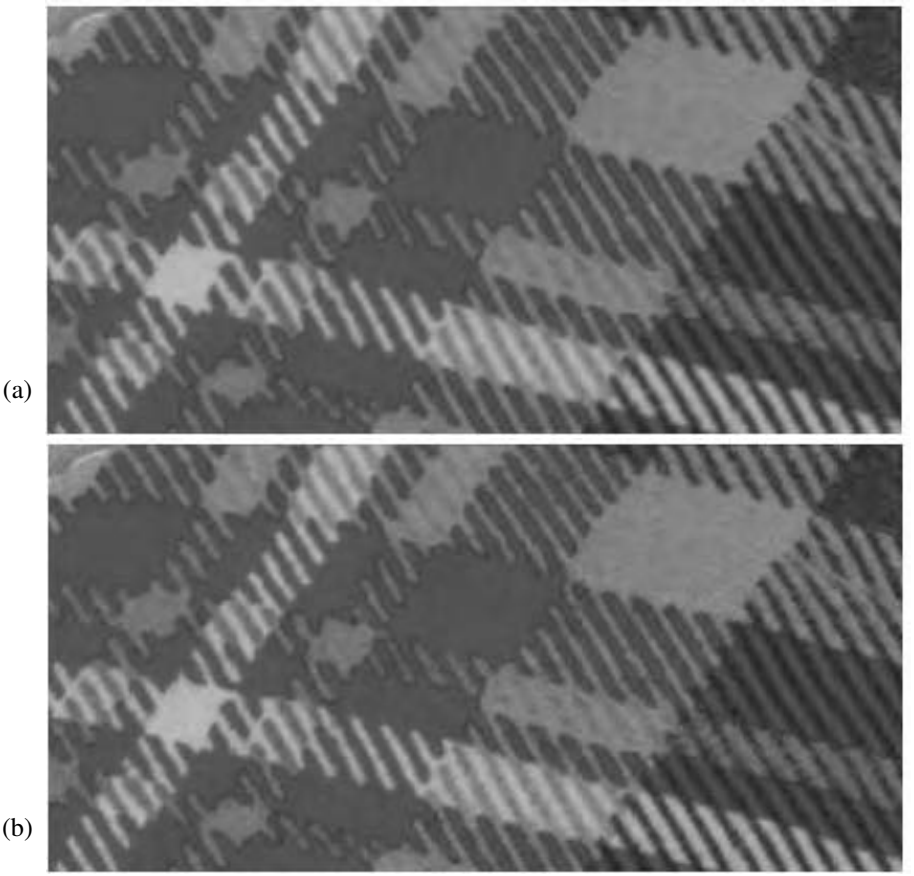


FIGURE 14.12 (See color insert.)

Comparison of the images reconstructed from a real CFA image by using bilinear demosaicking and JPEG2000 coding at 10:1 compression ratio: (a) conventional pipeline and (b) alternative pipeline.

We have provided several examples of how to model the performance of individual elements and their interactions using a Taylor series expansion (demosaicking, color and gamma corrections, and interactions) or by extending coding gain modelling (compression). Various experimental results (both qualitative and quantitative) are also provided to demonstrate the validity of the proposed models.

Other applications of the developed models are also possible, including, for example, optimization of the image processing pipeline components. Consider the demosaicking approaches that put different weights on the importance of spatial and spectral correlations (parameters χ_1 and χ_2 in Equation 14.10). This means that for smooth images with high spatial correlations, demosaicking methods with small χ_1 should be used. On the other hand, for images with high spectral correlations (e.g., images with little color content), demosaicking methods with small χ_2 can be expected to achieve better performance. The analysis and tools provided in this chapter could help the engineers and scientists to better understand these and similar issues arising in the development of DSC processing chains.

References

[1] M. Mancuso and S. Battiato, "An introduction to the digital still camera technology," *ST Journal of System Research*, vol. 2, no. 2, pp. 1–9, December 2001.

[2] J.E. Adams, "Interactions between color plane interpolation and other image processing functions in electronic photography," in *Proceedings of the SPIE Conference on Cameras and Systems for Electronic Photography and Scientific Imaging*, San Jose, CA, USA, March 1995, vol. 2416, pp. 144–151.

[3] E. Chang, S. Cheung, and D. Pan, "Color filter array recovery using a threshold-based variable number of gradients," in *Proceedings of the SPIE Conference on Sensors, Cameras, and Applications*, San Jose, CA, USA, January 1999, vol. 3650, pp. 36–43.

[4] D.R. Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal," U.S. Patent 4 642 678, February 1987.

[5] J.F. Hamilton and J.E. Adams, "Adaptive color plane interpolation in single sensor color electronic camera," U.S. Patent, 5 629 734, May 1997.

[6] C.A. Laroche and M.A. Prescott, "Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients," U.S Patent 5 373 322, December 1994.

[7] R. Kimmel, "Demosaicing: Image reconstruction from color CCD samples," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 1221–1228, March 1999.

[8] T.W. Freeman, "Median filter for reconstructing missing color samples," U.S. Patent 4 724 395, February 1988.

[9] B.K. Gunturk, Y. Altunbasak, and R.M. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transcations on Image Processing*, vol. 11, no. 9, pp. 997–1013, September 2002.

[10] S.C. Pei and I.K. Tam, "Effective color interpolation in CCD color filter arrays using signal correlation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 6, pp. 503–513, June 2003.

[11] L. Chang and Y.P. Tan, "Effective use of spatial and spectral correlations for color filter array demosaicking," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 355–365, February 2004.

[12] K. Parulski and K.E. Spaulding, *Digital Color Image Handbook*, ch. Color image processing for digital cameras. G. Sharma (ed.), Boca Raton, FL: CRC Press 2002, pp. 727–757.

[13] R. Ramanath, W.E. Snyder, Y. Yoo, and M.S. Drew, "Color image processing pipeline," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 34–43, January 2005.

[14] International Electrotechnical Commission, *Colour Measurement and Management in Multimedia Systems and Equipment - Part 2-1: Default RGB Colour Space - sRGB*. IEC 61966-2-1, 1999.

[15] T. Toi and M. Ohita, "A subband coding technique for image processing in single CCD cameras with Bayer color filter arrays," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 1, pp. 176–180, February 1999.

[16] Y.T. Tsai, "Color image compression for single-chip cameras," *IEEE Transaction on Electron Devices*, vol. 38, no. 5, pp. 1226–1232, May 1991.

[17] N. Zhang and X. Wu, "Lossless compression of color mosaic images," *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1379–1388, June 2006.

[18] S.Y. Lee and A. Ortega, "A novel approach of image compression in digital cameras with a Bayer color filter," in *Proceedings of the IEEE International Conference on Image Processing*, Thessaloniki, Greece, October 2001, vol. 3, pp. 482–485.

[19] B. Parrein, M. Tarin, and P. Horain, "Demosaicking and JPEG2000 compression of microscopy images," in *Proceedings of the IEEE International Conference on Image Processing*, Singapore, October 2004, vol. 1, pp. 521–524.

[20] C.C. Koh, J. Mukherjee, and S.K. Mitra, "New efficient methods of image compression in digital cameras with color filter array," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1448–1456, November 2003.

[21] T. Sakamoto, C. Nakanishi, and T. Hase, "Software pixel interpolation for digital still cameras suitable for a 32-bit MCU," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 4, pp. 1342–1352, November 1998.

[22] W. Lu and Y.P. Tan, "Color filter array demosaicking: New method and performance measures," *IEEE Transactions on Image Processing*, vol. 12, no. 10, pp. 1194–1210, October 2003.

[23] R. Lukac, K.N. Plataniotis, and D. Hatzinakos, "Color image zooming on the Bayer pattern," *IEEE Transactions on Circuit and Systems for Video Technology*, vol. 15, no. 11, pp. 1475–1492, November 2005.

[24] C.A. Poynton, *A Technical Introduction to Digital Video*. New York: John Wiley & Sons, 1996.

[25] "Electronic still picture imaging - removable memory, part 2: Image data format - TIFF/EP," *Technical Committee ISO/TC 42, Photography, ISO 12234-2*, January 2001.

[26] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards*. Norwell, MA: Kluwer Academic Publishers, 1995.

[27] "Exchangeable image file format for digital still cameras: Exif version 2.2," *Technical Standardization Committee on AV & IT Storage Systems and Equipment, JEITA CP-3451*, April 2002.

[28] D.S. Taubman and M.W. Marcellin, "JPEG2000: Standard for interactive imaging," *Proceedings of the IEEE*, vol. 90, no. 8, pp. 1336–1357, August 2002.

[29] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, July 1997.

[30] R. Lukac and K.N. Plataniotis, "Fast video demosaicking solution for mobile phone imaging application," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 675–681, May 2005.

[31] L. Zhang, X. Wu, and P. Bao, "Real-time lossless compression for mosaic video sequences," *Real-Time Imaging*, vol. 11, no. 5-6, pp. 370–377, October-December 2005.

[32] N. Zhang and X. Wu, "Lossless compression of color mosaic images," *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1379–1388, June 2006.

[33] A. Bazhyna, A. Gotchev, and K. Egiazarian, "Near-lossless compression algorithm for Bayer pattern color filter arrays," in *Proceedings of the SPIE-IS&T Conference on Electronic Imaging: Digital Photography*, San Jose, CA, USA, January 2005, vol. 5678, pp. 198–209.

[34] R. Lukac and K.N. Plataniotis, "Single-sensor camera image compression," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 299–307, May 2006.

[35] B.E. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 1976.

[36] X. Xie, G. Li, Z. Wang, C. Zhang, D. Li, and X. Li, "A novel method of lossy image compression for digital image sensors with Bayer color filter arrays," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005, pp. 4995–4998.

[37] E. Dubois, "Frequency-domain methods for demosaicking of Bayer-sampled color images," *IEEE Transactions on Signal Processing Letters*, vol. 12, no. 12, pp. 847–850, December 2005.

[38] K.H. Chung and Y.H. Chan, "Color demosaicking using variance of color differences," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 2944–2955, October 2006.

[39] N.X. Lian, L. Chang, Y.P. Tan, and V. Zagorodnov, "Adaptive filtering for color filter array demosaicking," *IEEE Transactions on Image Processing*, vol. 16, no. 10, pp. 2515–2525, October 2007.

[40] N.X. Lian, Y.P. Tan, and V. Zagorodnov, "Error inhomogeneity of wavelet based compression," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3659–3669, July 2007.

[41] J. Katto and Y. Yasuda, "Performance evaluation of subband coding and optimization," in *Proceedings of the SPIE Conference on Visual Communication and Image Processing*, Boston, USA, November, 1991, vol. 1605, pp. 95–106.

[42] D.B.H. Tay, "Rationalizing the coefficients of popular biothogonal wavelet filters," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 6, pp. 998–1005, September 2000.

[43] M. Lightstone, E. Majani, and S.K. Mitra, "Low bit-rate design considerations for wavelet-based image coding," *Multidimensional Systems and Signal Processing*, vol. 8, no. 1-2, pp. 111–128, January 1997.

[44] O. Egger, P. Fleury, T. Ebrahimi, and M. Kunt, "High-performance compression of visual information: A tutorial review – Part I: Still pictures," *Proceedings of the IEEE*, vol. 87, no. 6, pp. 976–1011, June 1999.

[45] T.D. Tran and T.Q. Nguyen, "A progressive transmission image coder using linear phase uniform filterbanks as block transforms," *IEEE Transactions on Image Processing*, vol. 8, no. 11, pp. 1493–1507, November 1999.

[46] T.D. Tran, R.L. de Queiroz, and T.Q. Nguyen, "Linear-phase perfect reconstruction filter bank: Lattice structure, design, and application in image coding," *IEEE Transactions on Signal Processing*, vol. 48, no. 1, pp. 133–147, January 2000.

[47] J. Liang and T.D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme," *IEEE Transactions on Signal Processing*, vol. 49, no. 12, pp. 3032–3044, December 2001.

[48] N.X. Lian, L. Chang, V. Zagorodnov, and Y.P. Tan, "Reversing demosaicking and compression in color filter array image processing: Modeling and analysis," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3261–3278, November 2006.

[49] N.S. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice-Hall Press, 1984.

[50] R.M. Gray and D.L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2382, October 1998.

15

Lossless Compression of Color Mosaic Images and Videos

Ning Zhang, Xiaolin Wu, and Lei Zhang

15.1 Introduction

Lossless compression of raw sensor data captured by a digital camera with a color filter array (CFA) is a problem in digital photography with many applications. For instance, it is desirable to preserve the original sensor data so that ultra-high quality joint spatiotemporal color demosaicking and/or superresolution reconstruction can be performed offline after the image capture. Lossless mosaic image compression is also required for archival purposes when the digital camera is used in scientific and medical fields. In this chapter we investigate the problem of lossless compression of CFA mosaic images, and present both intraframe mosaic compression techniques for still images and interframe mosaic compression technique for videos.

Technically, lossless compression of CFA mosaic images poses a unique challenge of spectral decorrelation of spatially interleaved samples of three or more primary colors. We study reversible lossless spectral-spatial transforms that can remove statistical redundancies in both spectral and spatial domains. The main result of this study is a particular wavelet decomposition scheme, called Mallat wavelet packet transform, that is ideally suited to the task of decorrelating color mosaic data.

We also design a practical, fast codec for lossless compression of a time sequence of CFA mosaic frames. This design strikes a balance between the compression performance and the codec throughput. To make the encoding throughput high enough for real-time lossless mosaic video compression, we propose a hybrid scheme of inter and intraframe coding. Interframe predictive coding is invoked only when the motion between adjacent frames is modest and a simple motion compensation operation can significantly improve compression performance. Otherwise, still frame compression is performed to keep the complexity low. Experimental results show that the proposed scheme achieves higher lossless video compression ratio than existing methods such as JPEG-LS and JPEG-2000.

The remaining sections are organized as follows. Section 15.2 discusses the benefit of putting lossless compression before the demosaicking process. Sections 15.3 and 15.4 present and evaluate some schemes of coding mosaic images by deinterleaving red (R), green (G), and blue (B) samples prior to compression. In Sections 15.5 and 15.6, we consider an alternative approach of compressing color mosaic images directly without deinterleaving the color bands. In Section 15.5, we study the strength and weakness of both DPCM and wavelet-based lossless coding methods in the direct compression approach. Section 15.6 offers a wavelet analysis of mosaic images. The analysis leads to a new wavelet decomposition scheme that is well suited for lossless coding of Bayer pattern [1] mosaic data directly without deinterleaving. This new wavelet decomposition, which resembles the SPACL mode of JPEG-2000 standard, has the nice property of decorrelating color samples both spatially and spectrally. In Section 15.7, we discuss lossless mosaic video coding, which utilizes the correlation between adjacent mosaic frames. Section 15.8 ends this chapter with a discussion of open issues to further improve lossless compression performance of mosaic images and videos.

15.2 Compression and Demosaicking in the Camera Imaging Pipeline

The system architecture of most digital cameras is the cascade of a demosaicking module followed by a compression module, as illustrated in Figure 15.1. For details refer to Chapters 1 and 3. In this architecture, the demosaicking module is made quite simple [2] in order to meet the design objectives of low cost, long battery life, and high frame rate. But due to the above complexity constraint the performance of demosaicking is suboptimal compared with more sophisticated demosaicking algorithms [3], [4], [5]. This limits the color reproduction quality of digital cameras of single sensor array, since how well the demosaicking process can recover the missing primary colors at each pixel holds the key to the output color fidelity.

One way to achieve the best possible quality of reproduced color images is to perform demosaicking off-camera so that it can be done on a powerful computer using an advanced demosaicking algorithm. This requires a reverse of the workflow of demosaicking and compression in Figure 15.1, and the camera has to compress the raw mosaic image directly rather than a demosaicked image. Furthermore, in order not to degrade the performance of subsequent demosaicking, the compression of raw mosaic image needs to be mathe-

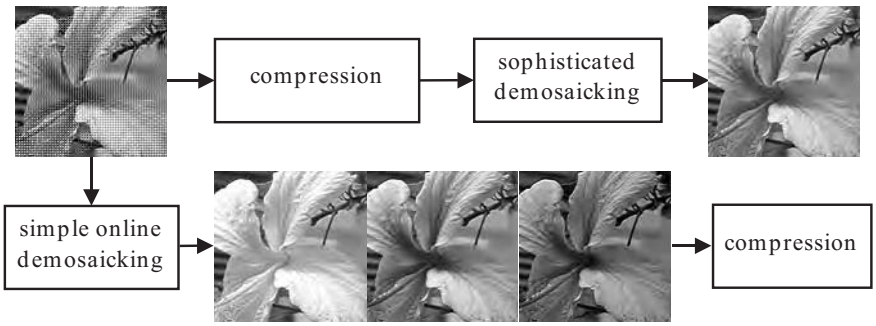


FIGURE 15.1

Two different orders of compression and demosaicking.

matically lossless or near-lossless, which is the subject of this chapter. Indeed, for many high-end digital photography applications, such as digital archiving of precious museum arts and relics, professional advertising, and digital cinema, users often demand the original mosaic data to be kept in lossless format. Other image/video applications such as superresolution imaging and motion analysis will also benefit from lossless compression of raw color mosaic data, because these operations are performed in even subpixel precision.

The workflow of demosaicking followed by compression of demosaicked image, which is adopted by most digital cameras, is largely motivated by the considerations of easy user interface, device compatibility, and standard compliance. However, industrial policy and standard compatibility aside, this popular workflow has some disadvantages. After demosaicking the amount of raw data will be tripled for color interpolation generates red, green, and blue bands in full spatial resolution. Ironically, the task of compression needs to decorrelate the three bands, which essentially attempts to reverse-engineer the color interpolation process of demosaicking. This unnecessarily increases algorithm complexity and burdens the on-camera I/O bandwidth. More detrimentally for some high-end applications, demosaicking before compression makes lossless reconstruction of raw mosaic data impossible because the underlying interpolation process is mathematically irreversible. Moreover, as shown by Lian et al. [6] and also as discussed in Chapter 14, the demosaicking error and compression distortion are not independent. They showed that the interaction of the two types of errors can degrade even lossy compression performance at high bit rates (e.g., near-lossless coding), if demosaicking is performed before compression.

For the above reasons we advocate the compression-first and demosaicking-later architecture for high-end professional cameras for which image quality is paramount and standard compliance is lesser an issue. As for the issue of complying to image coding standards, one can always transcode the final image to whatever required standard upon the completion of demosaicking process.

Seemingly lossless compression of raw mosaic image is a more challenging task than compression of continuous-tone images, and may have a lower compression ratio. But one should keep in mind that the total input image size of a raw mosaic image is only one-third of the continuous-tone counterpart in the first place. Furthermore, as argued in the previous paragraph, coding mosaic image directly relieves the camera from the tasks of

color demosaicking and color decorrelation. This can potentially reduce the requirements of on-camera computing power and processor I/O bandwidth.

Lossless compression of color mosaic images poses a unique and interesting problem of decorrelating spatially interleaved samples of different spectral bands. Take the ubiquitous Bayer pattern as an example. A color mosaic image consists of interlaced red, green, and blue samples. Existing decorrelation techniques such as DPCM, DCT and wavelets, although proven highly effectively on continuous-tone images, may not work properly on mosaic images. In this chapter we examine a number of interband coding techniques for lossless coding of color mosaic images. In particular, we are interested in reversible lossless spectral-spatial transforms that can remove statistical redundancies in both spectral and spatial domains. Our main result is a unique wavelet decomposition scheme, called Mallat packet transform, that is ideally suited to the task of decorrelating color mosaic data.

15.3 Deinterleaving

As already noted, existing decorrelation techniques may not work properly if applied directly to mosaic images which are not smooth signals. A simplistic approach to circumvent the difficulty is to deinterleave samples of different spectral bands and form down-sampled subimages of a given color, one per each spectral band. For the sake of concreteness and without loss of generality, let us focus our discussions on compression of mosaic images captured using the Bayer pattern. The following development can be easily generalized to other mosaic color sampling schemes. The Bayer pattern:

$$
\begin{array}{cccccccc}
G_{0,0} & R_{0,1} & G_{0,2} & R_{0,3} & G_{0,4} & R_{0,5} & G_{0,6} & R_{0,7} \\
B_{1,0} & G_{1,1} & B_{1,2} & G_{1,3} & B_{1,4} & G_{1,5} & B_{1,6} & G_{1,7} \\
G_{2,0} & R_{2,1} & G_{2,2} & R_{2,3} & G_{2,4} & R_{2,5} & G_{2,6} & R_{2,7} \\
B_{3,0} & G_{3,1} & B_{3,2} & G_{3,3} & B_{3,4} & G_{3,5} & B_{3,6} & G_{3,7} \\
G_{4,0} & R_{4,1} & G_{4,2} & R_{4,3} & G_{4,4} & R_{4,5} & G_{4,6} & R_{4,7} \\
B_{5,0} & G_{5,1} & B_{5,2} & G_{5,3} & B_{5,4} & G_{5,5} & B_{5,6} & G_{5,7} \\
G_{6,0} & R_{6,1} & G_{6,2} & R_{6,3} & G_{6,4} & R_{6,5} & G_{6,6} & R_{6,7} \\
B_{7,0} & G_{7,1} & B_{7,2} & G_{7,3} & B_{7,4} & G_{7,5} & B_{7,6} & G_{7,7}
\end{array}
$$

can be deinterleaved into the following three down-sampled subimages, each of which consists of samples in a single color band:

$$
\begin{array}{cccccccc}
G_{0,0} & & G_{0,2} & & G_{0,4} & & G_{0,6} & \\
 & G_{1,1} & & G_{1,3} & & G_{1,5} & & G_{1,7} \\
G_{2,0} & & G_{2,2} & & G_{2,4} & & G_{2,6} & \\
 & G_{3,1} & & G_{3,3} & & G_{3,5} & & G_{3,7} \\
G_{4,0} & & G_{4,2} & & G_{4,4} & & G_{4,6} & \\
 & G_{5,1} & & G_{5,3} & & G_{5,5} & & G_{5,7} \\
G_{6,0} & & G_{6,2} & & G_{6,4} & & G_{6,6} & \\
 & G_{7,1} & & G_{7,3} & & G_{7,5} & & G_{7,7}
\end{array}
$$

$$
\begin{array}{cccc}
R_{0,1} & R_{0,3} & R_{0,5} & R_{0,7} \\
R_{2,1} & R_{2,3} & R_{2,5} & R_{2,7} \\
R_{4,1} & R_{4,3} & R_{4,5} & R_{4,7} \\
R_{6,1} & R_{6,3} & R_{6,5} & R_{6,7}
\end{array}
\qquad
\begin{array}{cccc}
B_{1,0} & B_{1,2} & B_{1,4} & B_{1,6} \\
B_{3,0} & B_{3,2} & B_{3,4} & B_{3,6} \\
B_{5,0} & B_{5,2} & B_{5,4} & B_{5,6} \\
B_{7,0} & B_{7,2} & B_{7,4} & B_{7,6}
\end{array}
$$

The resulting green subimage forms a quincunx lattice, while the red and blue subimages are square lattice. To use any of the existing lossless image compression algorithms to compress the green quincunx image one has to transform it into a square lattice. Several ways of transforming or deinterleaving a quincunx lattice into a square lattice were proposed in the literature [7], which are summarized below.

1. *Simple shift:* The quincunx lattice of the green channel is converted to square lattice by shifting all odd columns one pixel to the left and forming the following realignment:

$$\begin{matrix} G_{0,0} & G_{0,2} & G_{0,4} & G_{0,6} \\ G_{1,1} & G_{1,3} & G_{1,5} & G_{1,7} \\ G_{2,0} & G_{2,2} & G_{2,4} & G_{2,6} \\ G_{3,1} & G_{3,3} & G_{3,5} & G_{3,7} \\ G_{4,0} & G_{4,2} & G_{4,4} & G_{4,6} \\ G_{5,1} & G_{5,3} & G_{5,5} & G_{5,7} \\ G_{6,0} & G_{6,2} & G_{6,4} & G_{6,6} \\ G_{7,1} & G_{7,3} & G_{7,5} & G_{7,7} \end{matrix}$$

2. *Quincunx separation:* The quincunx lattice of green pixels is further split into two square sublattices.

$$\begin{matrix} G_{0,0} & G_{0,2} & G_{0,4} & G_{0,6} \\ G_{2,0} & G_{2,2} & G_{2,4} & G_{2,6} \\ G_{4,0} & G_{4,2} & G_{4,4} & G_{4,6} \\ G_{6,0} & G_{6,2} & G_{6,4} & G_{6,6} \end{matrix} \qquad \begin{matrix} G_{1,1} & G_{1,3} & G_{1,5} & G_{1,7} \\ G_{3,1} & G_{3,3} & G_{3,5} & G_{3,7} \\ G_{5,1} & G_{5,3} & G_{5,5} & G_{5,7} \\ G_{7,1} & G_{7,3} & G_{7,5} & G_{7,7} \end{matrix}$$

3. *Interpolation deinterlacer:* The simple shift scheme distorts the two-dimensional (2D) spatial correlation, while the separation scheme destroys the correlation between the two subimages. A better alternative is an interpolation transform [7], [8]:

$$\begin{matrix} G_{0,0} & G_{0,2} & G_{0,4} & G_{0,6} \\ g_{1,1} & g_{1,3} & g_{1,5} & g_{1,7} \\ G_{2,0} & G_{2,2} & G_{2,4} & G_{2,6} \\ g_{3,1} & g_{3,3} & g_{3,5} & g_{3,7} \\ G_{4,0} & G_{4,2} & G_{4,4} & G_{4,6} \\ g_{5,1} & g_{5,3} & g_{5,5} & g_{5,7} \\ G_{6,0} & G_{6,2} & G_{6,4} & G_{6,6} \\ g_{7,1} & g_{7,3} & g_{7,5} & g_{7,7} \end{matrix}$$

in which the green samples in odd columns are passed through a vertical low pass filter and then aligned with even columns, as follows:

$$g_{(x,y)} = \frac{1}{4}\left[G_{(x-1,y-1)} + 2G_{(x,y)} + G_{(x-1,y+1)}\right] \tag{15.1}$$

Since $g_{(x,y)}$ increases the dynamic range of G by one bit, the above transform becomes inefficient for lossless coding. In other words, if the binary representation of $G_{(x,y)}$ has N bits, N+1 bits are required to represent $g_{(x,y)}$ in order to have lossless inverse transform. Although $g_{(x,y)}$ and $G_{(x,y)}$ have the same dynamic range in Equation 15.1, an extra bit is needed to resolve the parity of the sum for lossless reconstruction. For this reason the interpolation transform is more suitable for lossy compression of Bayer pattern mosaic data.

4. *Rotation:* The original quincunx lattice can be made an upright square lattice by a 45-degree rotation:

$$
\begin{array}{ccccccc}
 & & & G_{0,0} & & & \\
 & & G_{2,0} & G_{1,1} & G_{0,2} & & \\
 & G_{4,0} & G_{3,1} & G_{2,2} & G_{1,3} & G_{0,4} & \\
G_{6,0} & G_{5,1} & G_{4,2} & G_{3,3} & G_{2,4} & G_{1,5} & G_{0,6} \\
G_{7,1} & G_{6,2} & G_{5,3} & G_{4,4} & G_{3,5} & G_{2,6} & G_{1,7} \\
 & G_{7,3} & G_{6,4} & G_{5,5} & G_{4,6} & G_{3,7} & \\
 & & G_{7,5} & G_{6,6} & G_{5,7} & & \\
 & & & G_{7,7} & & &
\end{array}
$$

   Unfortunately, resulting image has a diamond boundary.

After any of the above deinterleaving transforms, the green channel can be coded using any of the existing lossless image codecs. To maintain an adequate frame rate of the camera the lossless mosaic image compression has to be performed in real time. The task is most suitable for a hardware implemented industrial standard, such as JPEG-LS and JPEG-2000. For this reason we evaluate the performance of JPEG-LS [9] and JPEG-2000 [10] (lossless mode) on the outputs of three of the above deinterleaving transforms. The lossless bit rates of mosaic images with different deinterleaving transforms and different lossless compression algorithms are listed in Table 15.1.

The comparison study above declares no clear winner among the deinterleaving methods over all the test images. Not surprisingly, the separation transform performs the worst on average because it disregards the correlation between the two resulting sub-images of green samples. The compression results of the merge and rotation transforms are very close for a given lossless image codec. In our comparison study JPEG-LS achieves better lossless compression than JPEG-2000 on all test images for the deinterleaving methods of separation and merge. For the rotation method we present only the results of JPEG-LS not those of JPEG-2000, because it is relatively easy to modify JPEG-LS to code the rotated image but very difficult to do the same with JPEG-2000.

TABLE 15.1
Lossless bit rates of Green channel under different deinterleaving transforms when compressed by JPEG-LS and JPEG-2000. The bold face numbers indicate the best results for the given test image.

	Separation		Shift		Rotation
Image	J2K	JPEG-LS	J2K	JPEG-LS	JPEG-LS
Woman	5.251	5.207	5.167	5.086	**5.004**
Bike	5.374	5.029	5.237	4.944	**4.919**
Monarch	4.699	4.518	4.496	4.318	**4.166**
Wall	6.197	**5.943**	6.147	5.985	6.068
Boat	5.364	5.176	5.200	**5.031**	5.131
Windows	6.418	**6.060**	6.389	6.320	6.285
Landscape	6.815	6.585	6.635	**6.401**	6.434
Fence	5.201	5.078	5.094	5.075	**5.054**
Lighthouse	5.234	5.050	5.063	**4.896**	4.976
Average	5.617	5.405	5.492	5.340	**5.337**

15.4 Interchannel Coding

The approach of compressing deinterleaved color bands separately is clearly suboptimal in terms of compression performance, because it fails to remove the statistical redundancies in the form of the spectral correlation. To overcome this drawback, we develop a technique of exploiting interchannel correlations in deinterleaved compression of mosaic images. The idea is to first code the quincunx green image and then code the red and blue subimages of CFA using the interchannel correlations. The coded quincunx green image is chosen to be used as an anchor for subsequent interchannel compression, because the Bayer CFA produces twice as many green samples as red and blue samples. Consequently, the sample correlation in the green channel is much higher than in the red and blue channels.

Once the compressed quincunx green image is decoded and made known to the decoder, the green samples of CFA can be used as causal context to predict the red and blue samples. Specifically, at the red and blue sample positions of CFA where the green samples are missing, we can predict the missing green samples from the existing CFA green samples. Denote the predicted green values by $g_{(x,y)}$, where the value of $x-y$ is odd. The green samples of CFA are denoted by $G_{(x,y)}$, where the value of $x-y$ is even. Likewise, the red and blue samples of CFA are denoted by $R_{(x,y)}$ and $B_{(x,y)}$, with $x-y$ being odd. We do not compress the red and blue subimages $R_{(x,y)}$ and $B_{(x,y)}$ separately as in the previous section, but instead we perform lossless coding of the two color difference images:

$$\alpha_{(x,y)} = R_{(x,y)} - g_{(x,y)} \quad \text{and} \quad \beta_{(x,y)} = B_{(x,y)} - g_{(x,y)} \tag{15.2}$$

Since the decoder can make the same prediction $g_{(x,y)}$ as the encoder, it can reconstruct the original samples $R_{(x,y)}$ and $B_{(x,y)}$ from $\alpha_{(x,y)}$ and $\beta_{(x,y)}$ without any loss. The difference images $\alpha_{(x,y)}$ and $\beta_{(x,y)}$ can be regarded as two chrominance components of the original true color image signal. For typical nature scenes, $\alpha_{(x,y)}$ and $\beta_{(x,y)}$ are low-pass signals due to high spectral correlation, and hence they are more compressible than $R_{(x,y)}$ or $B_{(x,y)}$.

TABLE 15.2
Lossless bit rates of red and blue mosaic samples using JPEG-LS.

	Intrachannel		Bilinear Interpolation		CubicSpline Interpolation	
Image	R	B	R-g	B-g	R-g	B-g
Woman	5.185	5.273	4.751	4.792	**4.698**	**4.653**
Bike	4.812	5.254	**4.571**	5.003	4.647	**4.954**
Monarch	4.528	4.625	**4.323**	**4.334**	4.397	4.391
Wall	5.962	5.937	5.566	5.526	**5.421**	**5.343**
Boat	5.199	5.349	4.833	5.023	**4.726**	**4.912**
Windows	6.062	6.040	5.688	5.664	**5.505**	**5.501**
Landscape	6.594	6.531	6.085	6.138	**6.039**	**6.122**
Fence	5.118	5.051	4.713	4.774	**4.645**	**4.773**
Lighthouse	5.053	5.154	4.778	4.881	**4.752**	**4.854**
Average	5.39	5.468	5.034	5.126	**4.981**	**5.056**

TABLE 15.3
Lossless bit rates of deinterleaved mosaic images by JPEG-LS and JPEG-2000.

	No interpolation		Bilinear interpolation	
Image	JPEG-LS	JPEG-2K	JPEG-LS	JPEG-2K
Woman	5.158	5.220	4.929	4.968
Bike	4.989	5.289	4.866	5.064
Monarch	4.447	4.613	4.323	4.448
Wall	5.967	6.152	5.766	5.855
Boat	5.153	5.323	4.980	5.085
Windows	6.186	6.388	5.998	6.086
Landscape	6.482	6.697	6.256	6.406
Fence	5.080	5.154	4.909	4.926
Lighthouse	5.000	5.161	4.863	4.960
Average	**5.384**	**5.555**	**5.210**	**5.311**

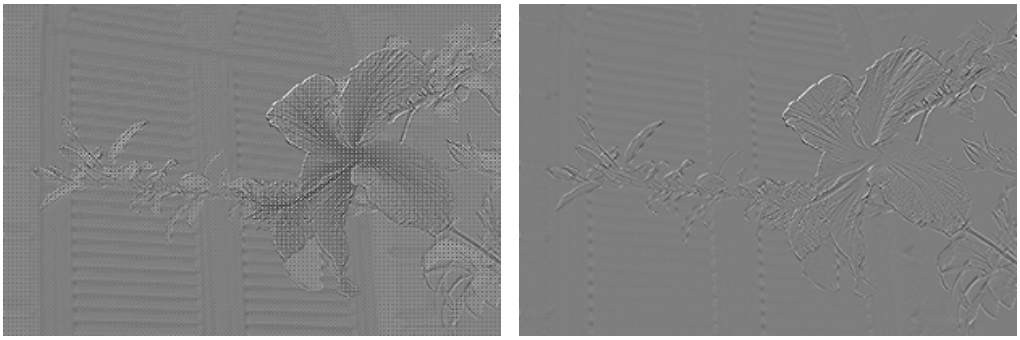
The prediction of the missing green samples $g_{(x,y)}$ is essentially an interpolation problem. Many image interpolation techniques can be applied here, such as bilinear interpolation, bicubic B-Spline [11] and some nonlinear methods [12], [13]. Table 15.2 lists the lossless bit rates of the red and blue mosaic samples obtained by three compression schemes: i) intrachannel coding of red and blue subimages, ii) color difference coding with bilinear green interpolation, and iii) color difference coding with bicubic B-spline green interpolation. The results given in Table 15.2 clearly demonstrate that coding color differences is more effective than coding the red and blue channels individually. The coding gain is more than 7.5% on average, which is a significant margin by the standard of lossless image coding. More sophisticated interpolation algorithms for the missing green samples of CFA can improve the lossless compression performance of color mosaic images, but only marginally. Simple bilinear interpolation works satisfactorily in practice.

Table 15.3 presents the overall lossless bit rates of JPEG-LS and JPEG-2000 lossless mode on the deinterleaved green channel (using the merge deinterleaving transform) and the two color difference images α and β with $\mathbf{g}$ being estimated by bilinear interpolation. For comparison purposes we also give the results of coding red and blue channels directly without interband decorrelation.

The interchannel compression approach of coding color difference images $\alpha_{(x,y)}$ and $\beta_{(x,y)}$ has a beneficial side effect for the digital camera workflow. The signals $\alpha_{(x,y)}$ and $\beta_{(x,y)}$ provide vital information in many color demosaicking algorithms [2], [3], [4], [5]. Therefore, lossless coding of $\alpha_{(x,y)}$ and $\beta_{(x,y)}$ serves dual purposes of lossless compression of color mosaic data and demosaicking.

15.5 Towards Direct Compression

Most investigators of mosaic image compression chose to compress deinterleaved color channels of CFA rather than dealing with a CFA image as a whole. This is apparently



(a) (b)

FIGURE 15.2

Residual images of the median predictor of JPEG-LS: (a) Bayer mosaic, and (b) original green channel. The mid-grey represents zero.

because mosaic images are not smooth like continuous-tone images, and hence unsuitable for existing image compression techniques. However, from the point of view of camera architecture, it is advantageous to compress the CFA mosaic image directly without deinterleaving the color channels. This will simplify the hardware design and reduce the system cost versus the approach of lossless compression after deinterleaving.

For continuous-tone images the best lossless compression performance is achieved by adaptive predictive coding, or the approach of differential pulse coded modulation (DPCM). The popular lossless image codecs of DPCM type are JPEG-LS [9] and CALIC [14]. But what happens if we apply a DPCM lossless image coding algorithm directly to raw color mosaic images without any preprocessing?

The design objective of DPCM is to remove long term memory of a smooth signal in the spatial domain via predictive coding. Given that the DPCM predictor is a low-pass filter in nature, it is ineffective to remove statistical redundancy hidden in the periodic patterns of CFA mosaic images. The handicap of DPCM in compressing CFA mosaic images can be clearly seen by examining prediction residual images. Figure 15.2 compares the prediction residual images of JPEG-LS, in which the median predictor is used, on a continuous-tone image (given in Figure 15.6a) and the corresponding mosaic image of Bayer CFA (given in Figure 15.6b). The DPCM residual signal of the latter image has much greater energy than that of the former. Moreover, one can see that the DPCM residual signal of the mosaic image is far from being i.i.d., exhibiting the two-dimensional CFA mosaic structure. This means that predictive coding in spatial domain cannot effectively decorrelate samples of CFA mosaic images.

The problem is also exposed by comparing the statistics of the prediction residuals of a continuous-tone image and its mosaic counterpart. It is a well known fact that the DPCM residuals of a continuous-tone image obey a Laplacian distribution (see Figure 15.3b). But the residual distribution for the CFA mosaic image can be far more complex and image dependent. This difference in the residual distributions is exemplified in Figure 15.2, where the histograms of prediction residuals of JPEG-LS are plotted for the above mosaic image and the original image. Note that the distribution of Figure 15.3a is multimodal and asymmetric with respect to the origin. In general, the DPCM residuals of CFA mosaic images

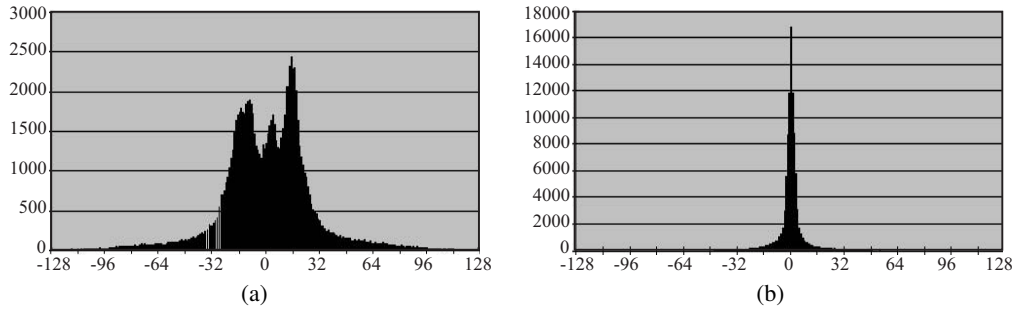


FIGURE 15.3

Histograms of JPEG-LS residuals: (a) mosaic image, and (b) original green channel. Histograms correspond to images shown in Figure 15.2.

cannot be satisfactorily modelled by a Laplacian distribution or a generalized Gaussian distribution, and have significantly higher empirical entropy than the DPCM residuals of continuous-tone images.

An alternative lossless image compression technique is reversible integer wavelet. The lossless mode of JPEG-2000 is such a wavelet-based lossless image codec. Now the question is how well a wavelet-based image compression method performs on a CFA mosaic image without deinterleaving. One might dismiss this idea quickly for the same reason that the CFA mosaic image is discontinuous. But surprisingly, experiments show that JPEG-2000 achieves on average more than 10% higher lossless compression ratio than JPEG-LS on raw CFA mosaic images. To understand the suitability and potential of wavelet compression on raw CFA images, we analyze in the next section the decorrelation mechanism of wavelet. This analysis will lead us to a particular wavelet decomposition, called Mallat packet transform, that can pack the energy of a CFA mosaic image in spatial-frequency domain far more efficiently than in the spatial domain. Finally, our development will result in a practical wavelet method for lossless compression of CFA mosaic images directly without deinterleaving.

15.6 Wavelet Compression

Unlike the spatial decorrelation method of DPCM, the wavelet can compactly describe an image signal in a joint frequency-spatial domain. Specifically, we will show that the lifting scheme of reversible integer wavelet is particularly suited for coding periodic CFA mosaic signals. We start by examining an intrinsic interplay between the Bayer pattern and the 2D integer wavelet transform via separable one-dimensional lifting. One level of 2D wavelet transform produces four subbands that have clear interpretations of the attributes of a Bayer CFA mosaic image. As shown in Figure 15.4, a 2D wavelet transform (after decimation) and the Bayer CFA mosaic pattern both have a 2×2 periodical sampling structure. Such a correspondence makes the 2D wavelet transform a powerful tool to describe the Bayer CFA mosaic image in frequency-spatial domain. The effect of performing a 2D wavelet

G	R	G	R		LL	HL	LL	HL
B	G	B	G		LH	HH	LH	HH
G	R	G	R		LL	HL	LL	HL
B	G	B	G		LH	HH	LH	HH

FIGURE 15.4

The 2×2 periodical sampling pattern of: (left) Bayer mosaic data, and (right) 2D wavelet transform.

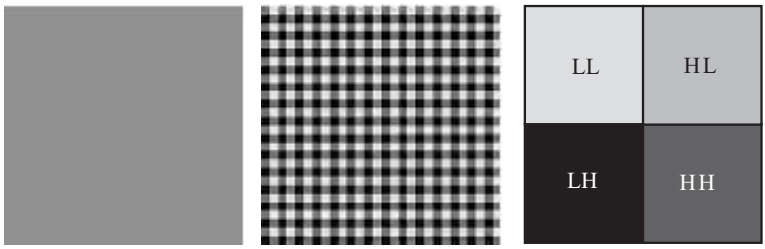


FIGURE 15.5

Efficiency of representing Bayer CFA mosaic image in wavelet domain: (a) input image with uniform coloration; (b) Bayer mosaic image; (c) 2D wavelet decomposition.

transform on a mosaic image can be understood by using an example in Figure 15.5. Here a uniform color produces a Bayer CFA mosaic image of high frequency, but the mosaic image is transformed into four constant subbands.

Let us analyze the effect of Figure 15.4 for the 5-3 integer wavelet. The analysis for other wavelets such as 9/7M, 5/11-C [10] is similar. The 5-3 integer wavelet uses the low-pass and high-pass filters with the following coefficients:

$$\mathbf{f}_L = \left(-\frac{1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, -\frac{1}{8}\right), \quad \mathbf{f}_H = \left(-\frac{1}{2}, 1, -\frac{1}{2}\right). \tag{15.3}$$

The 2D low pass filtering $\mathbf{F}_{LL} = \mathbf{f}_L^T \mathbf{f}_L$ of the Bayer mosaic image generates the LL subband that can be interpreted as the luminance component of the original color image. In a window of smooth color, where red, green and blue color components are approximately constants, (i.e., $R_{(x,y)} \cong R$, $G_{(x,y)} \cong G$, $B_{(x,y)} \cong B$), the coefficients in the LL subband are $\frac{1}{4}(R + 2G + B)$. This weighted average of R, G, and B values approximates the (luminance) component, Y, of the NTSC YUV color space. The same formula is also used by JPEG-2000 to generate the luminance component in a reversible color transform for lossless coding.

The three high frequency wavelet subbands of a Bayer mosaic image also have clear physical meanings using a model of mosaic images. The model was originally studied for the application of demosaicking [3], and it represents a Bayer mosaic image (see Figure 15.6b) as a sum of two component images:

$$\begin{matrix} G_{0,0} & R_{0,1} & G_{0,2} & R_{0,3} & G_{0,4} \\ B_{1,0} & G_{1,1} & B_{1,2} & G_{1,3} & B_{1,4} \\ G_{2,0} & R_{2,1} & G_{2,2} & R_{2,3} & G_{2,4} \\ B_{3,0} & G_{3,1} & B_{3,2} & G_{3,3} & B_{3,4} \\ G_{4,0} & R_{4,1} & G_{4,2} & R_{4,3} & G_{4,4} \end{matrix} = \begin{matrix} G_{0,0} & G_{0,1} & G_{0,2} & G_{0,3} & G_{0,4} \\ G_{1,0} & G_{1,1} & G_{1,2} & G_{1,3} & G_{1,4} \\ G_{2,0} & G_{2,1} & G_{2,2} & G_{2,3} & G_{2,4} \\ G_{3,0} & G_{3,1} & G_{3,2} & G_{3,3} & G_{3,4} \\ G_{4,0} & G_{4,1} & G_{4,2} & G_{4,3} & G_{4,4} \end{matrix} + \begin{matrix} 0 & \alpha_{0,1} & 0 & \alpha_{0,3} & 0 \\ \beta_{1,0} & 0 & \beta_{1,2} & 0 & \beta_{1,4} \\ 0 & \alpha_{2,1} & 0 & \alpha_{2,3} & 0 \\ \beta_{3,0} & 0 & \beta_{3,2} & 0 & \beta_{3,4} \\ 0 & \alpha_{4,1} & 0 & \alpha_{4,3} & 0 \end{matrix}$$

The first component image is the original green channel prior to CFA down sampling. This green image can be viewed as an approximation of the luminance component (see Figure 15.6c). The second component, with the terms $\alpha_{(x,y)}$ and $\beta_{(x,y)}$ defined as follows:

$$\begin{aligned} \alpha_{(x,y)} &= R_{(x,y)} - G_{(x,y)} \quad \text{for odd } x \text{ and odd } y, \\ \beta_{(x,y)} &= B_{(x,y)} - G_{(x,y)} \quad \text{for even } x \text{ and even } y, \end{aligned} \tag{15.4}$$

is a color difference image whose samples are arranged in a quincunx lattice. We can interpret the difference signals $\alpha_{(x,y)}$ and $\beta_{(x,y)}$ as two chrominance components of the original image.

Since most natural scenes are predominantly composed of pastoral (unsaturated) colors, there exist high correlations between the green and blue channels and between the green and red channels. Thus the two color difference images $\alpha = \mathbf{R} - \mathbf{G}$ and $\beta = \mathbf{B} - \mathbf{G}$ are low-pass. We consider an integer wavelet transform to be a linear operation by ignoring the rounding. The 2D wavelet transform of a mosaic image is thus equivalent to separately transforming the original green channel (resulting in Figure 15.6e) and the down sampled color difference image (resulting in Figure 15.6f), and then summing the results. The net effect is the image in Figure 15.6g, whose characteristics are very different from a wavelet transformed gray scale image.

Next, let us examine the effect of the two dimensional 5-3 HH high-pass filter:

$$\mathbf{F}_{HH} = \mathbf{f}_H^T \mathbf{f}_H = \frac{1}{4} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \tag{15.5}$$

Applying $\mathbf{F}_{HH}$ to a sample in Bayer CFA, say, position $G_{1,1}$, yields

$$\begin{aligned} x_{1,1} = &\left[G_{1,1} - \frac{1}{2} \sum_{\substack{(0,1),(1,2) \\ (1,0),(2,1)}} G(x,y) + \frac{1}{4} \sum_{\substack{(0,0),(0,2) \\ (2,0),(2,2)}} G(x,y) \right] \\ &- \frac{1}{2} \Big[(\alpha_{0,1} + \alpha_{2,1}) + (\beta_{1,0} + \beta_{1,2}) \Big] \end{aligned} \tag{15.6}$$

Equation 15.6 means that the HH subband is a composition of a luminance component and a chrominance component. The luminance component is the high-pass signal of the green channel, and chrominance component is the low-pass of the two color difference images $\alpha = \mathbf{R} - \mathbf{G}$ (averaged vertically) and $\beta = \mathbf{B} - \mathbf{G}$ (averaged horizontally). Since color difference images α and β are low passed signals in the first place, the 5-3 high-pass filter makes the HH responses of α and β even smoother, which is quite counter-intuitive. In other words, the 5-3 high-pass filter achieves spectral decorrelation. As a result, in the HH subband of a Bayer mosaic image (see the HH subband of Figure 15.6g), the details of the green channel (see the HH subband of Figure 15.6e) are superimposed on highly smoothed color difference signals (see the HH subband of Figure 15.6f). The above analysis shows that further decomposition of the HH subband by the 5-3 integer wavelet transform achieves a greater degree of separation of the green details from the smooth color difference signal. To the benefit of compression, the recursive packet 2D wavelet decomposition of Bayer

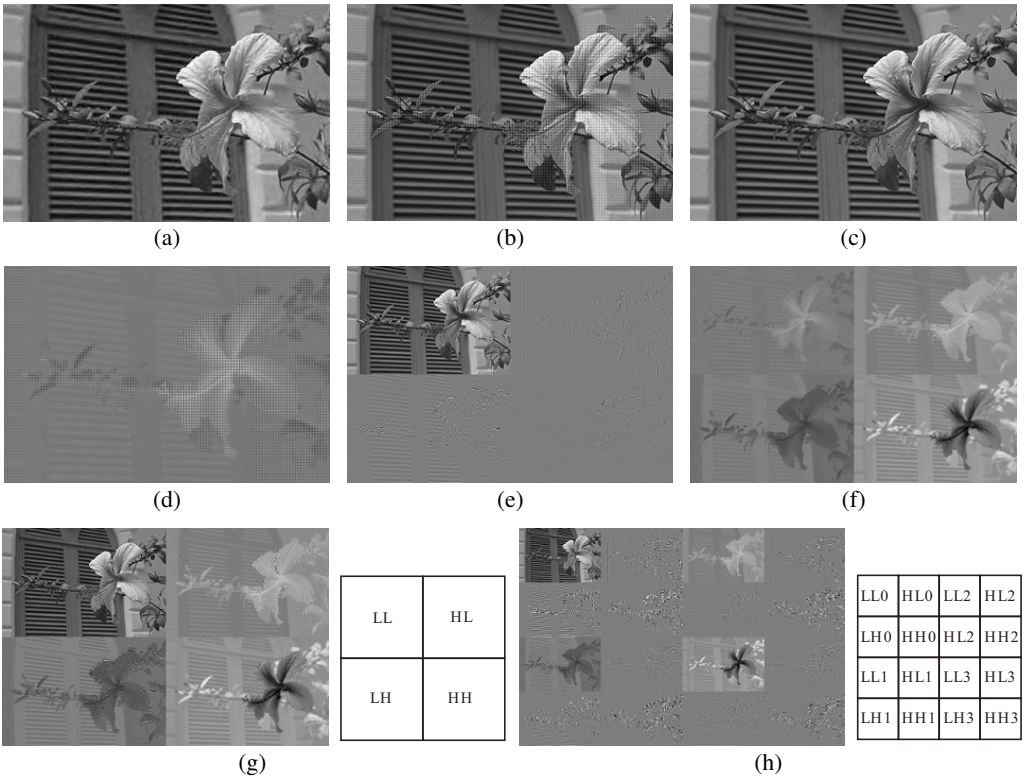


FIGURE 15.6

The mosaic image model and effects of wavelet transform on Bayer mosaic images: (a) original color image, (b) Bayer mosaic image, (c) original green channel, (d) checker-board color-difference signal corresponding to the subtraction of the original green channel from the mosaic image, and with mid-gray levels representing zero, (e) one-level 2D wavelet transform of the original green channel, (f) one-level 2D wavelet transform of the checker-board color-difference image, (g) one-level 2D wavelet transform of the mosaic image, which is approximately the sum of the previous two wavelet transformed images, and (h) two-level 2D Mallat packet decomposition of the mosaic image.

mosaic images exploits spatial and spectral correlations simultaneously. The packet like transform packs the energy of mosaic data into small areas in the frequency-spatial domain, and greatly facilitates subsequent entropy coding of wavelet coefficients. This effect can be visualized in Figure 15.6h.

The above derivations on the HH subband can be extended to the LH and HL subbands, and lead to similar conclusions. For instance, the output of the two dimensional 5-3 LH high-pass filter

$$\mathbf{F}_{LH} = \mathbf{f}_H^T \mathbf{f}_L = \frac{1}{16} \begin{bmatrix} 1 & -2 & -6 & -2 & 1 \\ -2 & 4 & 12 & 4 & -2 \\ 1 & -2 & -6 & -2 & 1 \end{bmatrix} \tag{15.7}$$

at position $B_{1,2}$ is

$$x_{1,2} = \left[\frac{3}{4} G_{1,2} - \frac{3}{8} \sum_{\left\{ \substack{(0,2) \\ (2,2)} \right\}} G_{(x,y)} + \frac{1}{4} \sum_{\left\{ \substack{(1,1) \\ (1,3)} \right\}} G_{(x,y)} - \frac{1}{8} \sum_{\left\{ \substack{(0,1),(0,3),(1,0) \\ (1,4),(2,1),(2,3)} \right\}} G_{(x,y)} \right.$$
$$\left. + \frac{1}{16} \sum_{\left\{ \substack{(0,0),(0,4) \\ (2,0),(2,4)} \right\}} G_{(x,y)} \right] + \left[-\frac{1}{8}\beta_{1,0} + \frac{3}{4}\beta_{1,2} - \frac{1}{8}\beta_{1,4} - \frac{1}{8} \sum_{\left\{ \substack{(0,1),(0,3) \\ (2,1),(2,3)} \right\}} \alpha_{(x,y)} \right] \quad (15.8)$$

Equation 15.8 shows that the LH subband consists of luminance and chrominance components just like the HH subband. The first term of Equation 15.8 corresponds to the luminance component, which is the detail signal of the usual LH subband of the green channel. The second term of Equation 15.8 corresponds to the chrominance component, which is further decomposed into two subcomponents: the smoothed color difference signal (the average of a four neighbor window):

$$\alpha = \mathbf{R} - \mathbf{G}$$

and a horizontally filtered signal:

$$\beta = \mathbf{B} - \mathbf{G}$$

by filter

$$H(f) = \frac{3}{4} - \frac{1}{4}\cos(2\pi f).$$

Note that there is a significant amount of attenuation to the chrominance component, and consequently the luminance component dominates in the LH subband.

Analogously to the LH subband, the HL subband can be shown to contain a luminance component and a chrominance component as well. The first is the detail signal of the usual HL subband of the green channel, and the second is a smoothed color difference signal β in a 2D window and a vertically filtered color difference signal α effected by $H(f)$. Summarizing the observations above, each of the LL, HL, LH, and HH subbands contains low frequency components of either chrominance or luminance signals of the original image. In order to more thoroughly remove both spatial and spectral statistical redundancies, we can perform recursive wavelet decompositions of these four subbands. This generates sixteen subbands as shown in Figure 15.6h. As shown in the figure, most of the energy of mosaic image is packed into four low frequency subbands LL0, LL1, LL2 and LL3. Specifically, the LL0 subband contains down sampled luminance information. The LL1, LL2 and LL3 subbands contain the energy of the chrominance signals, and hence they play the role of spectral decorrelation. The remaining twelve subbands represent much of the image details in both luminance and chrominance, and they are high frequency signals of low amplitude, just as the high frequency wavelet subbands of normal continuous-tone images. We do not decompose these high frequency subbands any further.

With the revealed properties of wavelet transform of Bayer CFA mosaic images, we propose a particular wavelet decomposition scheme that is ideally suited for mosaic image compression, called Mallat packet transform. The Mallat packet transform is defined by a packet of four Mallat decompositions of LL0, LL1, LL2, and LL3 subbands, as illustrated

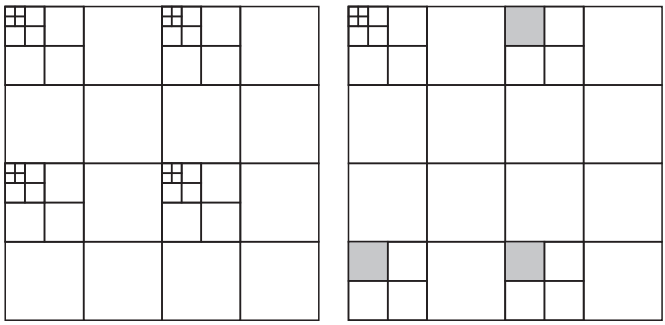


FIGURE 15.7

(a) Five-level Mallat packet decomposition, (b) the decomposition realizable by JPEG-2000 decomposition options and closest to the image on the left.

by Figure 15.7a. The proposed wavelet packet transform is a simple and effective way of decorrelating CFA mosaic data in both spatial and spectral domains. This technique is also attractive for camera hardware design because the compression codec operates directly on raw CFA mosaic images without any preprocessing.

Although the specific derivations above are with respect to the 5-3 integer wavelet, other reversible integer wavelet filters can also be used in the Mallat packet for lossless compression of CFA mosaic images. Table 15.4 lists the self-entropy results for some popular integer wavelets when the 5-level Mallat packet decomposition is performed on a set of test mosaic images. The experimental results show that the energy packing capabilities of the popular 5/3, 9/7M, 5/11-C [15] wavelets are about the same on a set of test mosaic images, while the simplest Haar wavelet is 4% inferior. The Mallat packet transform is a flexible framework that allows different wavelet filters to be used at different decomposition levels. The sixth column of Table 15.4 lists the self-entropy results obtained by applying 9/7M filter in first level decomposition (Figure 15.6g) and followed by the 5/3 filter in the other four

TABLE 15.4
Self-entropy of wavelet coefficients of the 5-level Mallat packet decomposition for different wavelet filters on mosaic images.

	Wavelet Transform					
Image	Haar	53	97M	5/11C	53+97M	53+5/11C
Woman	5.392	5.153	5.158	5.154	**5.121**	5.128
Bike	5.480	5.275	5.288	5.287	**5.255**	5.259
Monarch	5.120	4.697	4.672	**4.661**	4.694	4.689
Wall	6.003	5.858	5.895	5.883	**5.847**	5.850
Boat	5.561	5.355	5.370	5.364	**5.335**	5.338
Windows	6.189	5.956	5.985	5.983	**5.951**	5.955
Landscape	6.570	6.406	6.444	6.437	**6.400**	6.405
Fence	5.279	5.068	5.102	5.092	**5.067**	**5.067**
Lighthouse	5.396	5.213	5.254	5.240	5.215	**5.212**
Average	5.666	5.442	5.463	5.456	**5.432**	5.434

TABLE 15.5
Lossless bit rates of mosaic images by ECECOW for different integer wavelet transforms.

	Wavelet Transform					
Image	Haar	53	97M	5/11C	53+97M	53+5/11C
Woman	4.881	4.717	4.720	4.723	**4.699**	4.700
Bike	5.048	4.863	4.867	4.874	**4.847**	4.849
Monarch	4.627	4.285	4.255	**4.245**	4.283	4.277
Wall	5.632	**5.524**	5.560	5.561	5.527	5.525
Boat	5.075	4.898	4.907	4.903	4.880	**4.877**
Windows	5.887	5.678	5.699	5.703	**5.674**	**5.674**
Landscape	6.287	6.157	6.178	6.186	**6.154**	6.155
Fence	4.852	**4.678**	4.695	4.698	**4.678**	**4.678**
Lighthouse	4.905	**4.772**	4.803	4.795	4.777	**4.772**
Average	5.244	5.064	5.076	5.076	5.058	**5.056**

levels (Figure 15.7a). The results of a different combination of wavelet filters in different decomposition levels are given in column 7, where the 5/11C filter is used in first level and the 5/3 filter in subsequent four levels of the Mallat packet transform.

After the Mallat packet transform, we code the coefficients of Mallat packet transform using the high-order context-based entropy coding technique ECECOW [16]. The lossless bit rates of different integer wavelets when coded by ECECOW are listed in Table 15.5.

15.7 Interframe Lossless Video Compression

In this section our attention is turned to lossless compression of CFA mosaic video. The problem is important because most digital video cameras also adopt the design of single sensor array. Of course, one can always compress each frame of a raw mosaic video independently using one of the mosaic image compression methods described in the preceding sections. But significantly higher lossless compression ratio can be expected if an interframe coding technique is used in conjunction with motion estimation and compensation. Accurate motion estimation is, however, computationally intensive, and its real time implementation is difficult and costly. This is why motion-based video coding standard MPEG was primarily developed for stored video applications, where the encoding process is typically carried out offline on powerful computers. The MPEG design is asymmetrical in terms of complexity, with the encoder being much slower than the decoder, because only the decoder needs to play back the compressed video in real time. For lossless coding of mosaic data on video cameras, the system requirement is exactly opposite: encoder throughput has to be high enough to achieve real-time recording. For this reason we develop an interframe coding scheme that requires only a small fraction of the cost of full-search motion estimation. In addition, we switch from interframe coding mode to intraframe coding mode if the former does not offer significant coding gain.

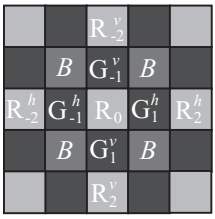


FIGURE 15.8

A row and a column of mosaic data that intersect at a red sampling position.

15.7.1 Switch of Coding Mode

The selection between intraframe and interframe coding modes depends on how fast and how well we can estimate the motions between adjacent frames. To estimate the motion between CFA mosaic frames, we first need to interpolate the missing green samples in frame F_{j-1}. This can be efficiently done using a linear directional filtering method; for example, the second order Laplacian correction filter proposed in Reference [2].

Let us examine the case depicted by Figure 15.8. In this figure, a column and a row of alternating green and red samples intersect at a red sampling position, where the missing green value needs to be estimated. The symmetric case of estimating the missing green values at the blue sampling positions of the Bayer pattern can be handled in the same way. Denote the red sample at the center of the window by R_0. Its interlaced red and green neighbors in horizontal direction are labelled as R_i^h, $i \in \{-2,2\}$, and G_i^h, $i \in \{-1,1\}$, respectively; similarly, the red and green neighbors of R_0 in vertical direction are R_j^v, $j \in \{-2,2\}$, and G_j^v, $j \in \{-1,1\}$. Let $\alpha_0 = G_0 - R_0$ be the unknown difference between green and red channels at the sample position of R_0. We first obtain an estimate of α_0, denoted by $\hat{\alpha}_0$, and then recover the missing green sample by $\hat{G}_0 \approx R_0 + \hat{\alpha}_0$. The reason for estimating $\alpha = G - R$ rather than the green value G directly is that the color difference signal α is low-pass for natural images. Referring to Figure 15.8, the horizontal and vertical differences between the green and red channels at R_0 can be estimated as

$$\begin{aligned} \alpha_0^h &= \frac{1}{2}\left(G_{-1}^h + G_1^h\right) - \frac{1}{4}\left(2R_0 + R_{-2}^h + R_2^h\right) \\ \alpha_0^v &= \frac{1}{2}\left(G_{-1}^v + G_1^v\right) - \frac{1}{4}\left(2R_0 + R_{-2}^v + R_2^v\right) \end{aligned} \tag{15.9}$$

Note that all the filter coefficients are of form 2^{-n}, $n \in Z$, which makes the hardware implementation very fast. In Reference [2], the second order gradient of red samples and the first order gradient of green samples are calculated respectively in horizontal and vertical directions. If the horizontal gradient is smaller than that of vertical, then $\hat{a}_0 = a_0^h$; otherwise $\hat{a}_0 = a_0^v$. This decision is to avoid the interpolation across image edges. For speed considerations we may omit these operations and simply let $\hat{a}_0 = (a_0^h + a_0^v)/2$. Our experiments show that this has almost no effect on compress performance. Denote the interpolated green channel of frame F_{j-1} as

$$\mathbf{G}_{j-1} = \begin{matrix} G_{0,0}^{j-1} & \hat{G}_{0,1}^{j-1} & G_{0,2}^{j-1} & \hat{G}_{0,3}^{j-1} & \cdots \\ \hat{G}_{1,0}^{j-1} & G_{1,1}^{j-1} & \hat{G}_{1,2}^{j-1} & G_{1,3}^{j-1} & \cdots \\ G_{2,0}^{j-1} & \hat{G}_{2,1}^{j-1} & G_{2,2}^{j-1} & \hat{G}_{2,3}^{j-1} & \cdots \\ \hat{G}_{3,0}^{j-1} & G_{3,1}^{j-1} & \hat{G}_{3,2}^{j-1} & G_{3,3}^{j-1} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{matrix} \tag{15.10}$$

Note that $\mathbf{G}_{j-1}$ consists of half the original green samples and half the interpolated samples. Denote the green channel of frame F_j as

$$\mathbf{G}_{j} = \begin{matrix} G_{0,0}^{j} & \times & G_{0,2}^{j} & \times & \cdots \\ \times & G_{1,1}^{j} & \times & G_{1,3}^{j} & \cdots \\ G_{2,0}^{j} & \times & G_{2,2}^{j} & \times & \cdots \\ \times & G_{3,1}^{j} & \times & G_{3,3}^{j-1} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{matrix} \tag{15.11}$$

Let (d_x, d_y) be the motion vector between the two frames $\mathbf{G}_{j-1}$ and $\mathbf{G}_j$. In video compression, the estimation accuracy of (d_x, d_y) is one of the most important factors that affect the final compression performance. However, the heavy computation load of accurate motion estimation makes it impractical for our purposes. In this paper, we restrict $d_x, d_y \in \{-1, 0, 1\}$. To further streamline computations we determine d_x and d_y separately in x and y search windows:

$$\begin{cases} d_x = \arg \min\limits_{d \in \{-1,0,1\}} \sum\sum\limits_{n+m=even} \left| G_{n,m}^{j} - G_{n,m+d}^{j-1} \right| \\ d_y = \arg \min\limits_{d \in \{-1,0,1\}} \sum\sum\limits_{n+m=even} \left| G_{n,m}^{j} - G_{n+d,m}^{j-1} \right| \end{cases} \tag{15.12}$$

After (d_x, d_y) is computed, a distance between $\mathbf{G}_j$ and $\mathbf{G}_{j-1}$

$$\mathrm{D}_1 = \sum\sum_{n+m=even} \left| G_{n,m}^{j} - G_{n+dy,m+dx}^{j-1} \right| \tag{15.13}$$

is computed to quickly evaluate the merit of interframe coding. A large D_1 value indicates that big relative motions or abrupt scene changes may occur in these frames. Consequently, interframe coding may be inefficient compared with intraframe coding. Therefore, an intraframe coding performance measure D_0 is needed, which will be compared against D_1 in the switch between intraframe and interframe coding modes. The measure D_0 is to predict, with minimum computation, the compressibility of the current green frame $\mathbf{G}_j$ by exploiting the intraframe redundancies only. Referring to Equation 15.13, for each pixel $G_{n,m}^{j}$ with the even value of $n+m$, we predict it by only its two nearest neighbors within frame F_j:

$$\hat{G}_{n,m}^{j} = (G_{n-1,m-1}^{j} + G_{n-1,m+1}^{j})/2 \tag{15.14}$$

The intraframe coding performance is then measured by

$$\mathrm{D}_0 = \sum\sum_{n+m=even} \left| G_{n,m}^{j} - \hat{G}_{n,m}^{j} \right| \tag{15.15}$$

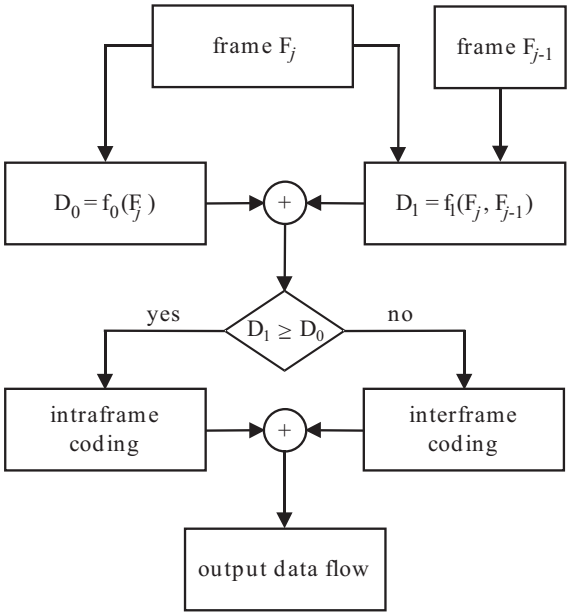


FIGURE 15.9

Illustration of the encoding procedure.

which reflects the prediction error. A small D_0 value implies that the intraframe correlation is high, and thus simple intraframe coding should suffice to compress $\mathbf{G}_j$. Finally, we have the following simple coding mode selection criterion: If $D_1 \geq D_0$, go to intraframe coding described in Section 15.6; otherwise, go to interframe coding described in Section 15.7.2.

15.7.2 Interframe Coding

Figure 15.9 illustrates the whole encoding procedure of the proposed scheme. When $D_1 \geq D_0$, an intraframe mosaic compression algorithm is used. If $D_1 < D_0$, $\mathbf{G}_j$ is to be compressed by using interframe coding with the help of $\mathbf{G}_{j-1}$. We define the difference image between $\mathbf{G}_j$ and $\mathbf{G}_{j-1}$ as

$$\tilde{\mathbf{G}} = \begin{matrix} \tilde{G}_{0,0} & \times & \tilde{G}^j_{0,2} & \times & \cdots \\ \times & \tilde{G}^j_{1,1} & \times & \tilde{G}^j_{1,3} & \cdots \\ \tilde{G}_{2,0} & \times & \tilde{G}^j_{2,2} & \times & \cdots \\ \times & \tilde{G}^j_{3,1} & \times & \tilde{G}^{j-1}_{3,3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{matrix} \tag{15.16}$$

where $\tilde{G}_{n,m}$, for d_x and d_y from Equation 15.12, is given by

$$\tilde{G}_{n,m} = G^j_{n,m} - G^{j-1}_{n+dy,m+dx}. \tag{15.17}$$

Now the lossless compression of $\mathbf{G}_j$ becomes the problem of entropy coding of $\tilde{\mathbf{G}}$. The residual terms in $\tilde{\mathbf{G}}$ are coded in raster scan order from left to right and top to bottom. After

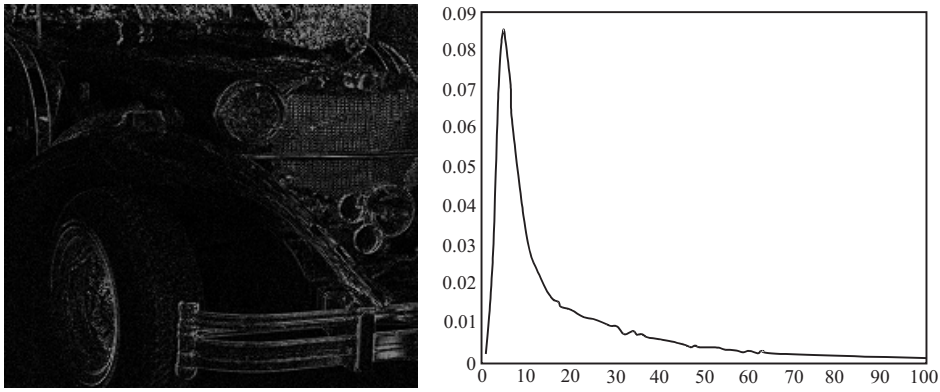


FIGURE 15.10

(a) A difference image $\tilde{\mathbf{G}}$ for the test sequence shown in Figure 15.7a; (b) the distribution of $P_{n,m}$ for the difference image shown on the left where the horizontal axis stands for the magnitude of pixels and the vertical axis denotes the probability density.

the interframe prediction, the samples of difference image $\tilde{\mathbf{G}}$ remain somewhat correlated. Large values of $\tilde{G}_{n,m}$ tend to register with edges, and the neighbors of $\tilde{G}_{n,m}$ with significant magnitude are also likely to have high magnitude, as shown by Figure 15.10a. This form of correlation can be exploited by applying context-based coding to $\tilde{\mathbf{G}}$, [17]. Specifically, the coding of $\tilde{G}_{n,m}$ is conditioned on the energy level in its neighborhood. Define the neighborhood energy of $\tilde{G}_{n,m}$ as

$$P_{n,m} = \left|\tilde{G}_{n-1,m-1}\right| + \left|\tilde{G}_{n-1,m+1}\right| + \left(\left|\tilde{G}_{n-2,m}\right| + \left|\tilde{G}_{n-2,m}\right|\right)/2 \tag{15.18}$$

We quantize $P_{n,m}$ into some conditioning contexts in which $\tilde{G}_{n,m}$ will be coded. Figure 15.10a shows a difference image $\tilde{\mathbf{G}}$ computed from a video sequence and Figure 15.10b plots the distribution of $P_{n,m}$ for this difference image. We quantize $\tilde{G}_{n,m}$ into K contexts Θ_κ, $\kappa = 1,2,...,\mathrm{K}$, by partitioning the range of $P_{n,m}$. The optimal partition can be obtained via dynamic programming. Empirically we found that the simple dyadic partition of $P_{n,m}$ works well in practice. Let

$$\begin{cases} \tilde{G}_{n,m} \in \Theta_K & \text{for } P_{n,m} \geq 2^{K-1} \\ \tilde{G}_{n,m} \in \Theta_\kappa & \text{for } 2^{\kappa-1} \leq P_{n,m} < 2^\kappa,\ \kappa = 2,3,...,K-1 \\ \tilde{G}_{n,m} \in \Theta_1 & \text{for } 0 \leq P_{n,m} < 2. \end{cases} \tag{15.19}$$

After the green channel $\mathbf{G}_j$ of frame F_j is coded, we proceed to code the red and blue channels of F_j, $\mathbf{R}_j$ and $\mathbf{B}_j$:

$$\mathbf{R}_j = \begin{matrix} \times & R^j_{0,1} & \times & R^j_{0,3} & \cdots \\ \times & \times & \times & \times & \cdots \\ \times & R^j_{2,1} & \times & R^j_{2,3} & \cdots \\ \times & \times & \times & \times & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{matrix}, \quad \mathbf{B}_j = \begin{matrix} \times & \times & \times & \times & \cdots \\ B^j_{1,0} & \times & B^j_{1,2} & \times & \cdots \\ \times & \times & \times & \times & \cdots \\ B^j_{3,0} & \times & B^j_{3,2} & \times & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{matrix} \tag{15.20}$$

The definition of $\mathbf{R}_{j-1}$ and $\mathbf{B}_{j-1}$ for frame F_{j-1} is the same as Equation 15.20. Since the compression of the blue channel is symmetrical to that of the red channel, it suffices to only discuss the coding of $\mathbf{R}_j$. Notice that if (d_x, d_y), the displacement between frame F_j and F_{j-1}, is equal to $(0,0)$, then the pixel $R^j_{n,m}$ in $\mathbf{R}_j$ can be matched to an original red pixel $R^{j-1}_{n,m}$ in frame F_{j-1}. Otherwise, a reference sample of $R^j_{n,m}$, denoted by $\hat{R}^{j-1}_{n+dy,m+dx}$, has to be interpolated in frame F_{j-1}. However, because the sampling frequency of red channel is only half of that of green channel, the interpolation accuracy of missing red samples is much lower than that of missing green samples. Therefore, though $\hat{G}^{j-1}_{n+dy,m+dx}$ may be a good prediction of $G^j_{n,m}$, $\hat{R}^{j-1}_{n+dy,m+dx}$ may not predict $R^j_{n,m}$ well. With this consideration, we apply interframe coding to $\mathbf{R}_j$ only when $(d_x, d_y) = (0,0)$. Denote by

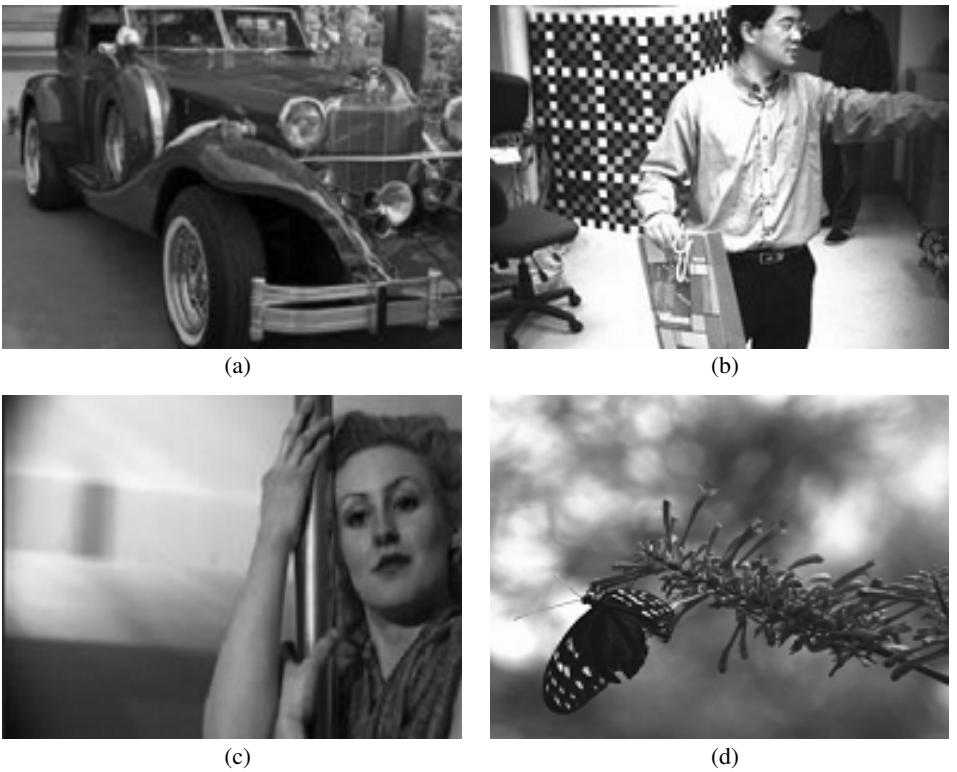
$$\tilde{\mathbf{R}} = \mathbf{R}_j - \mathbf{R}_{j-1} = \begin{matrix} \times & \tilde{R}_{0,1} & \times & \tilde{R}_{0,3} & \cdots \\ \times & \times & \times & \times & \cdots \\ \times & \tilde{R}_{2,1} & \times & \tilde{R}_{2,3} & \cdots \\ \times & \times & \times & \times & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{matrix} \tag{15.21}$$

the difference red image, where $\tilde{R}_{n,m} = R^j_{n,m} - R^{j-1}_{n,m}$. The encoding procedure for $\mathbf{R}_j$ and $\mathbf{B}_j$ is simply as: If $(d_x, d_y) = (0,0)$, then perform entropy coding of $\tilde{\mathbf{R}}$ and $\tilde{\mathbf{B}}$ as that of $\tilde{\mathbf{G}}$; otherwise, perform intraframe coding of $\mathbf{R}_j$ and $\mathbf{B}_j$ according to the description in Section 15.6.

15.7.3 Experimental Results

The proposed mosaic video lossless coding techniques are evaluated in comparison with JPEG-LS and JPEG-2000 lossless mode. Four mosaic video sequences were used in our experiments. The first video sequence is originally captured on film at a rate of 24-frames/second and then digitized by high-resolution scanner. The mosaic data are simulated by subsampling the true color image using the Bayer pattern. Figure 15.11a shows the scene of it, whose size is 512×768. The second to fourth video sequences are captured by a digital video camera at a rate of 24-frames/second. The spatial resolution for the second clip is 480×640 (Figure 15.11b), the third clip is 512×768 (Figure 15.11c) and the last clip is 1280×1536 (Figure 15.11d). All the videos are stored in 8-bits depth. Four lossless compression methods were used to code the clips: JPEG-LS, JPEG-2000 (in its lossless mode using 5-3 integer wavelet), the intraframe lossless coding method presented in the previous section (denoted as IFC-LS), and the hybrid interframe and intraframe coding scheme described in Section 15.7.2 (denoted as HC-LS). In the implementation of the hybrid method, the input mosaic image is divided into blocks (such as 128×128, 64×64, etc.) and the HC-LS algorithm is applied to each block. In our experiments, the block size is set as 64×64. Table 15.6 lists the coding rates of the four methods on the four clips. The reported compression results are the average of 50 consecutive frames for the first clip, 100 frames for the second clip, 24 frames for the third clip and 48 frames for the fourth clip.

From Table 15.6 we see that the proposed hybrid coding scheme for lossless compression of mosaic video achieves the lowest bit rates. The presented intraframe mosaic coding method also outperforms the JPEG-LS and JPEG-2000 standards.



(a) (b)

(c) (d)

FIGURE 15.11

The scenes in four tests clips with spatial resolution of (a) 512×768, (b) 480×640, (b) 512×768, and (d) 1280×1536 pixels.

TABLE 15.6
Lossless compression results (bpp) for the four video sequences by JPEG-LS, JPEG-2000, the proposed intraframe coding and the hybrid intra-interframe coding.

Video	JPEG-LS	JPEG-2000	IFC-LS	HC-LS
Clip 1	6.33	5.10	4.78	4.48
Clip 2	5.77	4.99	4.65	4.41
Clip 3	5.86	4.49	4.38	4.30
Clip 4	**5.38**	**4.30**	**4.09**	3.98

15.8 Conclusion

Various techniques for lossless coding of raw color mosaic images generated by CCD cameras of Bayer pattern were investigated. It turned out that the integer wavelet transform is ideally suited to the task, by offering efficient energy packing in both spatial and spectral domains. JPEG-2000 standard (in its lossless mode) can be conveniently applied to lossless compression of color mosaic images. As a simpler and faster alternative we have developed

a lossless mosaic image codec based on integer Mallat packet transform and fast context-based Rice coding [17]. This codec outperforms JPEG-2000 and JPEG-LS in both bit rate and speed. We also proposed a hybrid inter and intraframe coding technique that can exploit temporal correlation in addition to both spectral and spatial correlations. The hybrid method can significantly improve the lossless compression performance at a slightly higher computational cost than intraframe coding techniques. This is made possible by a judicious use of motion estimation and by greatly simplified motion vectors.

Lossless compression of mosaic data is relatively a new research area. The techniques discussed so far in this chapter are mainly targeting practical solutions for industrial applications. A variation of the solution presented in Reference [17] has been implemented using FPGA in the digital camera system of DALSA Corporation. It can encode 4096×2048 frame size mosaic data at 60 frames per second in real time.

For high-end compression performance, the ideas from state of the art single image lossless compression algorithms like MRP [18] and GLICBAWLS [19] certainly can be borrowed and adapted for single mosaic image. For lossless video coding, the temporal extensions of CALIC [14], MRP and GLICBAWLS have also been proposed in References [20], [21], and [22]. They are helpful and inspiring for lossless mosaic video coding solutions. In our experience, adapting the above ideas on mosaic data can improve coding performance reported in this chapter by approximately 10 to 20 percent. The major gain is coming from using high order auto regression model to design sophisticated adaptive predictors. However, by the nature of its application, lossless mosaic coding has extremely high demand on encoding speed. There remain challenges on designing high performance lossless coding methods for mosaic data and justifying the involved cost at the same time.

Acknowledgment

We would like to thank DALSA Corporation for supporting our research project on the topics presented in this chapter.

References

[1] B. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 1976.

[2] J. Hamilton and J. Adams, "Adaptive color plane interpolation in single sensor color electronic camera," U.S. Patent 5 629 734, May 1997.

[3] X. Wu and N. Zhang, "Primary-consistent soft-decision color demosaicking for digital cameras," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1263–1274, September 2004.

[4] B. Gunturk, Y. Altunbasak, and R. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 997–1013, September 2002.

[5] L. Zhang and X. Wu, "Color demosaicking via directional linear minimum mean square-

error estimation," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2167–2178, December 2005.

[6] N. Lian, L. Chang, V. Zagorodnov, and Y. Tan, "Reversing demosaicking and compression in color filter array image processing: Performance analysis and modeling," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3261–3278, November 2006.

[7] C. Koh and S. Mitra, "Compression of Bayer color filter array data," in *Proceedings of the International Conference on Image Processing*, Barcelona, Spain, September 2003, vol. II, pp. 255–258.

[8] K. Hisakazu, M. Shogo, T. Ishida, and T. Kuge, "Reversible conversion between interlaced and progressive scan formats and its efficient implementation," in *Proceedings of the European Signal Processing Conference*, Toulouse, France, September 2002.

[9] I. JTC1/SC29/WG1, "Information technology - Lossless and near-lossless compression of continuous-tone still images," Tech. Rep. ISO FDIS 14495-1 (JPEG-LS), also ITU Recommendation T.87, 1998.

[10] I. JTC1/SC29/WG1, "JPEG 2000 part I final draft international standard," Tech. Rep. N1890, 2000.

[11] O. Rashkovskiy and W. Macy, "Method of determining missing color values for pixels in a color filter array," U.S. Patent 6 181 376, January 2001.

[12] D. Cok, "Signal processing method and apparatus for sampled image signals," U.S. Patent 4 630 307, December 1987.

[13] R. Hibbard, "Apparatus and method for adaptively interpolating a full color image utilizing luminance gradients," U.S. Patent 5 382 976, January 1995.

[14] X. Wu and N. Memon, "Context-based, adaptive, lossless image codec," *IEEE Transactions on Communications*, vol. 45, no. 4, pp. 437–444, April 1997.

[15] M. Adams, "Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation and analysis," *IEEE Transactions on Image Processing*, vol. 9, no. 6, pp. 1010–1024, June 2000.

[16] X. Wu, "High-order context modeling and embedded conditional entropy coding of wavelet coefficients for image compression," in *Proceedings of the 31st Asilomar Conference on Signals, System and Computers*, Pacific Grove, CA, USA, November 1997, pp. 1378–1382.

[17] N. Zhang and X. Wu, "Lossless compression of color mosaic images," in *Proceedings of the International Conference on Image Processing*, Singapore, October 2004, vol. 1, pp. 517–520.

[18] I. Matsuda, H. Mori, and S. Itoh, "Lossless coding of still images using minimum-rate predictors," in *Proceedings of the International Conference on Image Processing*, Vancouver, BC, Canada, September 2000, vol. I, pp. 132–135.

[19] B. Meyer and P. Tischer, "Glicbawls – grey level image compression by adaptive weighted least squares," in *Proceedings of the Data Compression Conference*, Snowbird, UT, USA, March 2001, p. 503.

[20] E. Carotti and J. Martin, "Motion-compensated lossless video coding in the calic framework," in *Proceedings of the 5th IEEE International Symposium on Signal Processing and Information Technology*, Athens, Greece, December 2005, pp. 600–605.

[21] H. Maeda, A. Minezawa, I. Matsuda, and S. Itoh, "Lossless video coding using multi-frame MC and 3D bi-prediction optimized for each frame," in *Proceedings of the European Signal Processing Conference*, Florence, Italy, September 2006.

[22] S. Andriani, G. Calvagno, and G. Mian, "Lossless video compression using a spatio-temporal optimal predictor," in *Proceedings of the European Signal Processing Conference*, Antalya, Turkey, September 2005, pp. 1324:1–4.

16

Automatic Red-Eye Removal for Digital Photography

Francesca Gasparini and Raimondo Schettini

16.1 Introduction

The red-eye effect is a well-known problem in photography. It is often seen in amateur shots taken with a built-in flash, but the problem is also well known to professional photographers. This effect is caused by the reflection of the blood vessels in the retina, usually when a strong and sudden light strikes the eye. Red eye is a function of at least three things:

- *Ambient light level.* Low level causes the subject's pupil to open wider to admit more light, exposing the retina.
- *Age of the subject.* Usually, young subjects suffer more from the red-eye effect since their retinas are wider.
- *Angle of the flash beam.* The light is bounced off the subject back to the camera. The closer the outgoing beam is to the reflected beam, the greater the red-eye effect. A flash bracket can help to get the flash a bit farther from the proximity of the lens. However, today's compact cameras worsen the problem because of the proximity of the flash unit and the lens.

The widespread use of small devices with built-in flash, including cell phones and handheld computers, produces a large amount of digital photographs that are potentially affected with red eye. A common technique to reduce red eye is to adopt multiple flashes to contract the pupils before the final shot. However, the effect can only be reduced and not completely removed. In addition, the flash consumes a significant amount of power.

With the advent of digital technologies, which make digitalized images either directly with a digital camera or by converting traditional photos with scanners, fixing red-eye artifacts digitally became an important skill. Currently, many image processing software applications in the market offer red-eye removal solutions. Most of them are semiautomatic or manual solutions. The user has to either click on the red eye or draw a box containing it before the red-eye removal algorithm is applied [1], [2], [3], [4]. There also exist methods which detect red-eye artifacts automatically [5], [6], [7], [8], [9], [10], [11], [12]. Existing methods are typically included in digital cameras, printers (e.g., *Picture Project* developed by Nikon and *Zoom Browser* by Canon), and image processing software (e.g., Adobe Photoshop, Corel Draw, PhotoPaint, Stoik RedEye AutoFix, RedBot). A typical problem for most available tools is poor pupil segmentation which leads to unnatural red-eye correction. Even with user input, these algorithms sometimes correct red-eye pixels too aggressively or leave many red-eye pixels uncorrected.

In the case of completely automatic solutions, there are several aspects to be considered. A first aspect is the low effectiveness of several methods; to avoid erroneous corrections of non red-eye pixels, it is preferable to reduce the overall effectiveness instead of creating a huge number of new defects. Another critical aspect of automatic procedures is related to the great variability of the red-eye artifacts. This variability strongly depends on the quality of the image and on the scene illumination. As can be seen in Figure 16.1, where some representative images with red-eye artifact are shown, the color of the red eyes can vary significantly for the same subject in different images and for different subjects in the same photos. Even worse, for the same subject it is not uncommon to have two differently colored red eyes, or only one red eye. This can lead either to the correction of only one of the two eyes or to different corrections for the eyes of the same person. Obviously, any method should avoid image degradation which could be caused by correcting false red eyes identified incorrectly in the detection step or improperly correcting true red eyes. Even if the detection is accurate, it is quite often that the correction step severely degrades the image compared to the appearance of the original red-eye defect.

Dobbs and Goodwin proposed a mechanism for recoloring a selected region of a digital image with red-eye defects [1]. Benati et al. [2], [3] reduced the user intervention by reducing the general region of interest. Schildkraut et al. [5] further developed the previous works and presented a fully automatic method able to detect and correct red-eye pairs without any user intervention. Their method uses eye and face detection techniques and a large number of features to distinguish between true red-eye pairs and other small red regions in the image. This results in a very complicated algorithm, which requires a large amount of computation. However the software produced is highly optimized [6]. In 1998, Patti et al. [13], [4] presented a method of correcting red-eye artifact with minimal user intervention. Their method operates on the assumption that the user selects a box around each eye and was among the first which enhanced the red-eye artifacts, highlighting the redness in order to better locate the red pupil. In 2001, Wang and Zhang [7] realized an

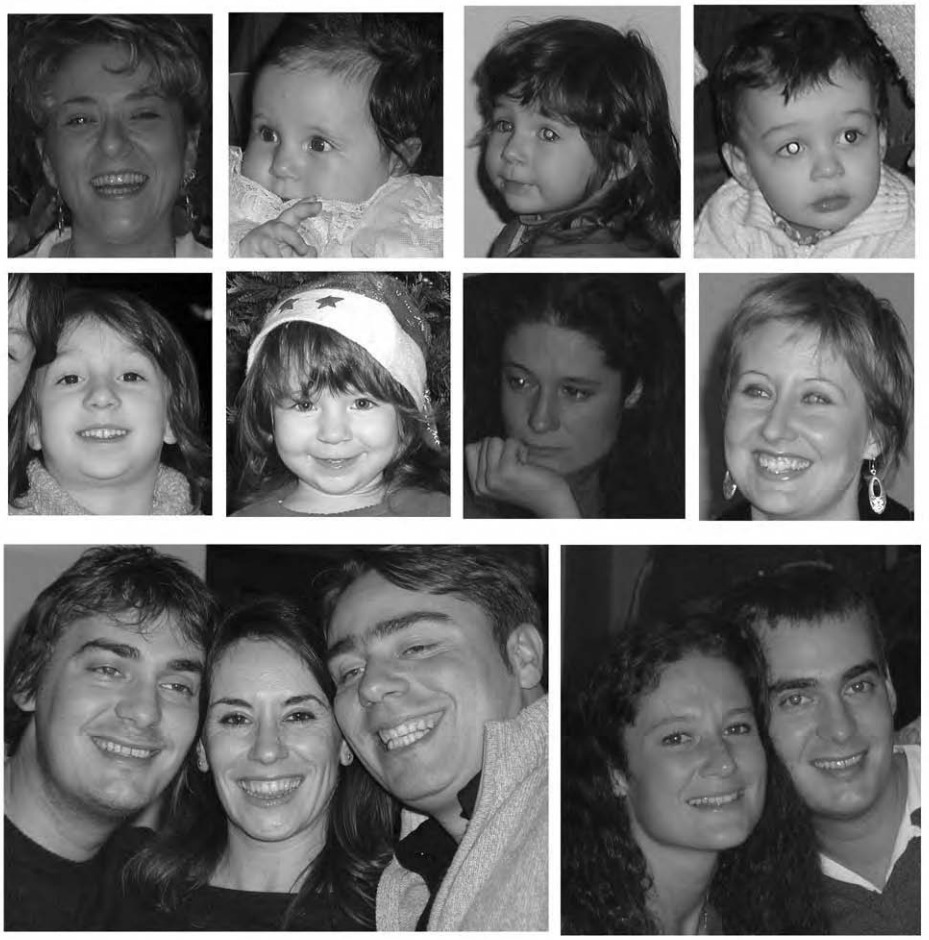


FIGURE 16.1 (See color insert.)

Examples of the variability of the red-eye artifacts.

automatic detection and correction system that reduces red-eye artifacts while maintaining the natural appearance of the eye. The red-eye detection module uses face and eye detectors based on neural networks [14] and principal component analysis. Another automatic red-eye detection and correction procedure was presented by Gaubatz and Ulichney in 2002 [8]. In their work, faces are detected with a cascade of multi-scale classifiers [15]. In late 2003, Ulichney et al. [16] developed a one-click system called *RedBot* available online [12]. Considerable work in red-eye detection and correction was done by Hardeberg and collaborators [10], [11], [17], [18], [19]. Hardeberg first proposed a semiautomatic method with manual selection of red-eye regions [17], [18]. In more recent works, Hardeberg et al. [19], [10] improved this semiautomatic approach and developed a fully-automatic method [11] that uses a preliminary color segmentation based on thresholding in different color spaces to locate skin regions. A gray-scale conversion of these regions is performed to enhance the redness of the red-eye defects. A gray-scale conversion of a red-enhanced image was also used in the method proposed by Held [20]. In this method, the search

space for the red-eye location is consecutively reduced by taking into account semantical information about the image, and the Hough transform is used to detect the eye center. The red-eye detection method proposed by Joffe [21] uses computer vision and machine learning techniques and can be combined with a red-eye correction method to obtain a fully-automatic red-eye removal solution.

A two-step (detection and correction) procedure was presented by Luo et al. [9]. The red-eye detection part is modelled as a feature-based object-detection problem which uses the Adaboost algorithm to simultaneously select relevant features and assign feature weights for the classifier. In the correction step, an adaptive recoloring algorithm is applied to perform desaturation and darkening in the detected red-eye region.

An approach combining image processing and classification techniques was developed by Zhang et al. [22]. In order to automatically detect red eyes, a heuristic algorithm is first adopted looking for a group of candidate regions. An eye classifier is then utilized to confirm whether each candidate region is a human eye or not. The authors also provided a semiautomatic red-eye detector that requires the user to click on the eye.

Gasparini and Schettini [23] developed a modular procedure for automatic detection and correction of red-eye artifacts in images of unknown origin. The most likely facial regions are identified by combining the results of a color-based face detector and multi-resolution neural network-based face detector. Red eyes are searched within these regions by seeking areas with high redness and applying various geometric constraints.

Wan et al. [24] developed a method based on active appearance models which constitute a statistical approach able to model a deformable object shape. Joining color information and a deformable model they locate red eyes as deformable objects. Volken et al. [25] used image processing and heuristics by looking first for the red zones of the whole image and then estimating the probability of each of the red-eye candidate zones through the evaluation of zones' roundness, the amount of white around these zones corresponding to the sclera, and the amount of skin. A probabilistic approach, based on stepwise refinement of a pixel-wise red-eye probability map, was presented by Willamowski and Csurka [26].

Marchesotti et al. [27] focused more on the problem of red-eye correction, generally less analyzed than red-eye detection. They proposed three correction methods, compared according to image degradation risk and expected perceptual quality improvement criteria. Depending on these criteria and the red-eye detection confidence, they proposed an adaptive system to select the correction strategy.

As can be seen from the above discussion, a number of techniques were developed for red-eye detection and removal. This chapter provides an overview of popular solutions. Namely, Section 16.2 describes the problem of red-pupil detection based on the assumption that the eye or face region has been automatically or manually selected. Different approaches available in the literature are presented and compared. Section 16.3 describes several modules that can reduce the region which is searched for red-eye artifacts. In particular, the section surveys relevant skin, face and eye detection techniques. The problem of correcting red-eyes is the focus of Section 16.4. The automatic red-eye detection and correction method proposed by the authors of this chapter is presented in Section 16.5. Finally, Section 16.6 describes red-eye detection and correction performance evaluation and summarizes the main ideas presented in this chapter.

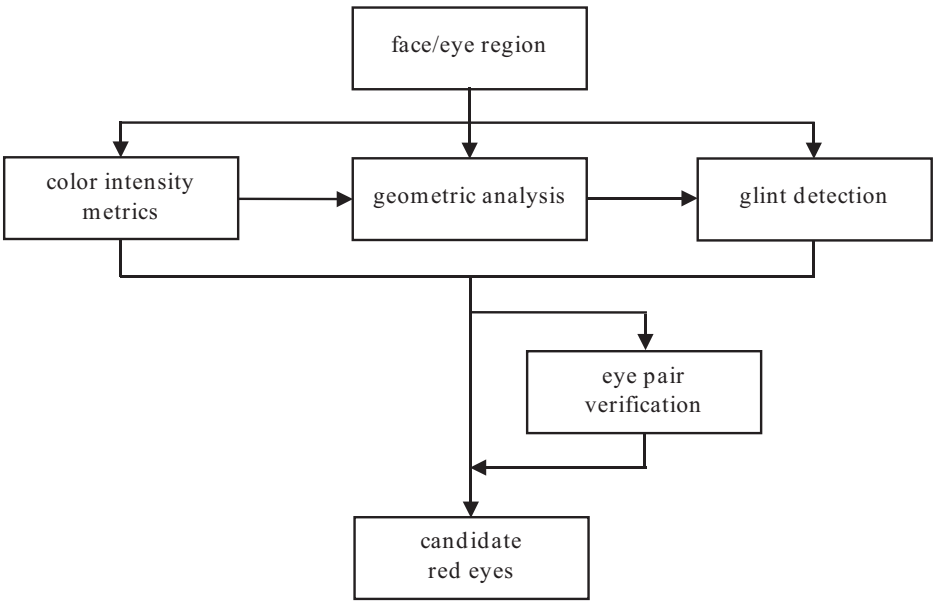


FIGURE 16.2

A generic modular approach for red-eye detection.

16.2 Red-Eye Detection

Red-eye detection strategies can be broadly divided into two classes. In the first class, it is assumed that the candidate eye regions are identified either manually or automatically. Figure 16.2 shows a general modular detection approach which starts by localizing the eye or face region. Note that the methods belonging to this category may not use all processing steps listed in this figure. In the second class, the red eyes are directly detected by scanning the whole image without a previous reduction of the search space. Initial candidate red-eye regions are re-evaluated in further verification steps based on low-level features (e.g., color, shape) and/or using high-level classifiers. Both these approaches are justified, considering the variety of red-eye artifacts. Figure 16.3 shows red eyes of different subjects acquired under different lighting conditions and with different digital cameras. The red-eye color distribution cannot be easily clustered regardless of the color space adopted. Furthermore, red-eye and non-red-eye color distributions are significantly overlapped. Figure 16.4 shows the color distributions of twenty-two representative red eyes and twenty-two representative normal eyes, expressed in the red-green-blue (RGB) color space, together with their overlapping zone.

16.2.1 Red-Eye Detection Within a Confined Region

In the first work on red-eye correction, the user had to hand-pick a pixel representative of the color of the artifact from a manually selected region [1]. Then, an elliptical chrominance discriminator was defined in the YIQ color space centered at this color. For each pixel

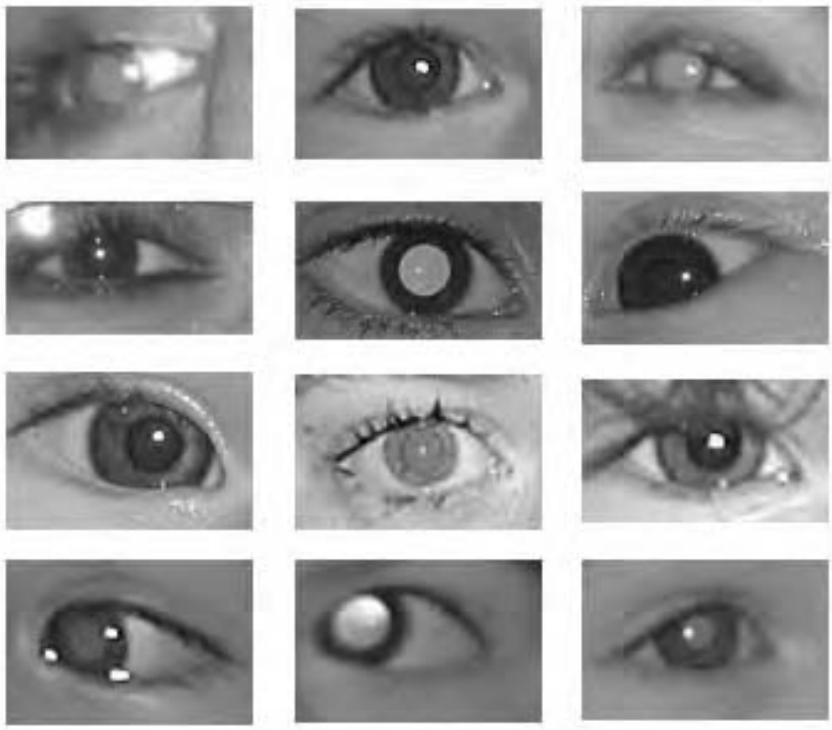


FIGURE 16.3 (See color insert.)

Examples of red eyes of different subjects, that shows the spread color distribution of the red-eye defects.

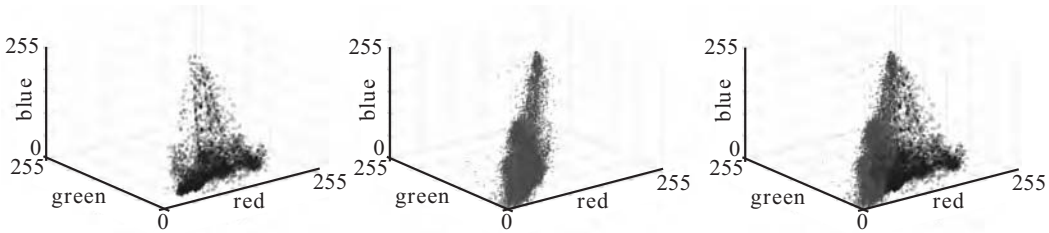


FIGURE 16.4

Color distributions in the RGB color space of: (left) 22 representative red eyes manually segmented, (middle) 22 representative normal eyes segmented manually, (right) the overlapping zone of the previous two.

location within the selected spatial region, the value of the pixel falling within the ellipse is modified into a new chrominance value.

Benati et al. [2], [3] reduced the user intervention by requiring only the general region of interest. To identify red-eye pixels, first a threshold in the HLS color space is applied, then the pixels are grouped into spatially contiguous regions, and a score is assigned to each region based on size, shape, color and brightness. The boundary location of each region is then refined with a region growing algorithm and a further score is calculated. The region with the highest score corresponds to the pupil to be corrected.

FIGURE 16.5 (See color insert.)

Further examples of the great variability of the red-eye artifacts. In the image on the left the subject has only one red eye, while in the image on the right the colors of the two eyes of the same subject are so different that probably most of the existing detection algorithms will miss one of them.

In general, most approaches convert the color portion of the image corresponding to a pupil into a new, typically grayscale, image using a nonstandard transformation, so that the red-eye artifacts are enhanced for better localization of the red pupil to be corrected. This grayscale image is usually defined as the *redness map*, and different transformations can be adopted to generate it. The candidate red pupils are usually located by binarizing the redness map using empirically defined thresholds. Morphological filters, geometric constraints and other approaches described in the following sections are then adopted to discriminate between true red pupils and other red spots. Further geometric and color assumption can also be used to precisely detect the pupil boundary. Some methods look only for a pair of red eyes, implicitly discarding the cases of a single red eye or the missed detection of one of the two red eyes. Figure 16.5 shows the case with a single red as well as the case where the detection algorithm can easily fail by omitting one of the two red eyes due to their significantly different redness.

Patti et al. [13] highlighted the red pixels using a non standard luminance-chrominance representation. After the user has selected a box around each eye, the algorithm performs a segmentation based on color information extracted from the selected region. The method employs an RGB to YCbCr color conversion with RGB values raised to the 1/3 power according to the following equations:

$$K' = \begin{cases} 4.5K & \text{if } K \leq 0.018 \\ 1.099K^{1/3} - 0.99 & \text{otherwise} \end{cases} \tag{16.1}$$

where K denotes, respectively, R, G, or B color component and

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}. \tag{16.2}$$

Assuming that the image is already gamma-corrected gives an effective gamma of about 6.6 to the data that increases the separation between red and other pixels. A binary map is created by thresholding the Cr chrominance component in this nonstandard YCbCr space. The threshold value is determined as:

$$T = Cr_{avg} + 0.2\,(Cr_{max} - Cr_{min}) \tag{16.3}$$

where the average (avg), the maximum and the minimum are all computed over the selected image region. After spatially filtering and refining the binary mask, the pupil location is detected with a block-matching estimation scheme under the assumption that the pupil is circular. The boundary of this circular mask is finally refined, taking into account the fact that red pupils often have no circular shape. Thus, pixels in the neighborhood of the initial mask are merged if they are close enough in terms of perceived redness which is defined using the median red (r_{med}) and median green (g_{med}) values of the candidate red-eye region. The authors chose the median instead of the mean in order to prevent the use of pixels of the glint, which is usually present inside the red-eye area, in a measure of redness. Pixels p_{ij} are merged to the mask if the change in this perceived redness is small enough. To measure the change two quantities were used. The first is the percentage deviation of a candidate pixel red value with respect to r_{med}, defined as follows:

$$\eta_{ij}^{r} = \frac{\left|r_{ij} - r_{med}\right|}{r_{med}} \cdot 100\% \tag{16.4}$$

The other is the percentage deviation in red-green difference expressed as:

$$\eta_{ij}^{r-g} = \frac{\left|\Delta_{ij}^{rg} - \Delta^{rgm}\right|}{\Delta^{rgm}} \cdot 100\% \tag{16.5}$$

where $\Delta_{ij}^{rg} = \left|r_{ij} - g_{ij}\right|$ and $\Delta^{rgm} = |r_{med} - g_{med}|$. If both η_{ij}^{r} and η_{ij}^{r-g} are below some predetermined thresholds, pixel p_{ij} is included in the final red-eye mask.

Gaubatz et al. [8] looked for red eyes in the upper half of a previously detected face and defined four metrics to emphasize redness, changes in redness, luminance and the glint caused by the reflection of the flash. These four metrics were employed to limit the search space, due to the variability of the red-eye artifact in images coming from different cameras with different performance characteristics. In particular, the redness map is defined as the ratio between the energy of the R component and the energy of the remaining G and B components:

$$Redness = R^2 / \left(G^2 + B^2 + c\right) \tag{16.6}$$

where c is a constant (with suggested value $c = 14$, Reference [28]) used to avoid singularities. The glint map of the image can be defined using a low-pass version of the Laplacian over the luminance plane Y:

$$glint(i,j) = \left(-\nabla^2 Y\,(i,j) * h\,(i,j)\right) \tag{16.7}$$

where $h(i,j)$ is a generic low-pass filter. Once all the metrics have been evaluated, the detection of the red pupils can be completed and the presence of two eye-sized clumps of pixels can be verified.

(a) (b) (c) (d)

FIGURE 16.6

Comparison of the three operators proposed by Smolka et al.: (a) original image, (b) operator defined in Equation 16.8, (c) operator defined in Equation 16.9, and (d) operator defined in Equation 16.10.

Hardeberg [17], [18] emphasizes the redness by calculating the colorimetric distance in the CIELAB color space between a prototypical reference red-eye color and each pixel of manually selected regions of interest containing the red eyes. The drawback of this approach is its high computational complexity and insufficient robustness due to the wide distribution of the colors corresponding to red-eye defects. Since the mask usually does not identify the location of the red eye with sufficient precision, several image processing techniques are combined to obtain the final mask. These techniques include *morphological opening* and *closing* to remove noise pixels and to fill holes, a *blob analysis* to group the pixels of the mask into objects and to select candidates based on features such as size and shape, *final postprocessing* such as replacing the object with a circular disc.

Smolka et al. [10] introduced three nonparametric methods (see Figure 16.6) for enhancing redness. The corresponding redness maps were obtained by applying the following three equations:

$$T_1 = (R - max\{G,B\}) \tag{16.8}$$

$$T_2 = \frac{(R - max\{G,B\})}{R} \tag{16.9}$$

$$T = T_1 \cdot T_2 = \frac{[(R - max\{G,B\})]^2}{R} \tag{16.10}$$

in each pixel location of the input color image. Among these three solutions, the best results were achieved by applying Equation 16.10.

Smolka et al. detected faces using color information and by performing morphological operations. Assuming the circular shape of the human eyes, the algorithm continues by detecting circles of high intensity in the grayscale redness image, filtering it with a series of annular filters of increasing radius. The maximal output of the edge-detecting filter series is assigned to each pixel of the redness map. In practice, the final red-eye map is formed as a union of disks of various radii, taking into account the fact that the shape of the red eye can significantly deviate from the circular shape. The final binary mask can be obtained by thresholding this map either with a parameter set experimentally or evaluated automatically using binarization methods.

Another definition of a red-enhanced space is given in Reference [20] as follows:

$$I_{red} = (R - min\{G,B\}) \tag{16.11}$$

The method first detects faces and then eyes within the detected faces. The red-eye mask is estimated by analyzing the redness distribution around the center of the detected eyes using progressive thresholds to find the optimal value to discriminate between foreground and background pixels. In the next step, morphological closing and opening are applied to clean the resulting mask, in particular to remove small holes, intrusions and outgrows.

The algorithm developed by Gasparini and Schettini [23] uses a similar definition of redness. The method looks for red eyes within the most probable face regions, by analyzing color, applying geometric constraints, and detecting candidate red eyes as the regions with high values of redness:

$$Redness = \max\left\{0, \frac{2R-(G+B)}{R}\right\}^2 \tag{16.12}$$

In order to limit the number of false hits the algorithm exploits additional geometric constraints, such as the percentage ratio between the area of the candidate red eye and the whole face, the red-pixel spatial distribution, and the roundness of the region considered.

16.2.2 Red-Eye Detection Within the Whole Image

Unlike previous methods which operate in manually or automatically selected regions of the image, other existing methods process the whole image. The method developed by Zhang et al. [22] searched the whole image for pixels that may be marked as red-color or nonskin color pixels. Operating in the RGB space, pixels satisfying

$$\begin{cases} R > 50 \\ R/(R+G+B) > 0.40 \\ G/(R+G+B) < 0.31 \\ B/(R+G+B) < 0.36 \end{cases} \tag{16.13}$$

are marked as red-color pixels, whereas nonskin pixels are identified as follows:

$$G/(R+G+B) > 0.40 \ \text{ or } \ B/(R+G+B) > 0.45 \tag{16.14}$$

To restrict red-color areas, the algorithm proceeds to remove invalid regions by size and brightness constraints and checks the surrounding nonskin pixels. Finally, an eye classifier is utilized to confirm each candidate region found in the previous steps. Only regions confirmed by the classifier as human eyes are passed to the auto-correction stage.

Willamowski and Csurka [26] proposed a probabilistic approach for red-eye detection and correction. The detection step produces a map which indicates the red-eye probability of each pixel. This is done by combining the color characteristics of the individual pixels with the characteristics of the pixels in their neighborhoods. First, the redness and the luminance of each individual pixel are computed as follows:

$$Redness = R-(G+B)/2 \tag{16.15}$$

$$Lum = 0.25R+0.6G+0.15B \tag{16.16}$$

$$RedLum = max(0, 2\cdot Redness - Lum) \tag{16.17}$$

and pixels that are more luminant than red are rejected by setting $RedLum = 0$. To obtain a probability map, the *RedLum* values obtained are normalized by dividing the maximum *RedLum* value found over the whole image. The probability map is then adjusted by favoring circular regions with high *RedLum* values. A threshold of 0.1 is applied to the probability map in order to eliminate pixels with low probabilities. Then, circular *min* filters of different radii are used to detect the pixels with the maximum aggregated response to these filters. Regions that are too large or too small, too elongated or not compact enough are rejected. Red-eye candidate classification allows for better discrimination between detected true red eyes and false positives on faces (usually leading to significant visual artifacts in lips and nose regions) and in the background. This classification is performed using two different classifiers, to distinguish separately red-eye patches from face errors and from background errors, thus penalizing face errors more than background errors. The output of the classification is used for further adjustment of the previous red-eye probability map. If classifiers are confident that the patch corresponds to a red eye rather than a face error or a background error, the probability attributed to the corresponding pixels is set to one. If only one classifier is confident, the probability attributed to the corresponding pixels is set to zero. Otherwise, the probability assigned to the region remains unchanged. To concentrate on the differences between true and false positive patches the classifiers are not trained on independently collected data, but on the outputs generated by the detection module.

Figure 16.7 allows for comparison of different redness maps. As can be seen, depending on the employed redness measure, the achieved redness maps can differ significantly. Since each method responds uniquely, it is usually difficult to select the best method, considering a variety of visual scenarios in the real-life.

A different approach to red-eye detection, completely based on machine learning techniques, was developed by Joffe [21]. First, a classifier is trained using boosting [29] to detect red eyes, feeding it with both positive and negative examples. The features considered are mainly related to color properties of the red eyes. Thus, the RGB image is converted into the YCbCr color space. Since skin often appears visually identical to red-eye pixels, the method considers a fourth color channel Cr*, properly designed using linear discriminant analysis as the linear combination of RGB values that maximizes the difference between red-eye pixels and skin pixels. A face detector based on wavelets [30] limits the false positives of the red-eye detection, constraining the eyes to belong to a face. Also the detection of the red-pupil boundaries is performed by training an edge detector with both positive and negative examples of red eyes.

Another approach that is quite different from most red-eye detection methods was developed by Wan et al. [24]. The approach is based on active appearance model technology [31] which allows to statistically model shape and gray-level appearance of the object of interest and locate deformable objects. Typical red-eye images can be used to construct a single red-eye active appearance model; however, the approach requires manually labelling each single red eye with 28 points. Namely, seven points for eyebrow, twelve points for the eye boundaries, eight points for the pupil and one point for the glint. Analyzing the whole image, several rectangles that potentially contain the red eyes are selected. The matches between the red-eye active appearance model and the actual image are performed in each of these candidate rectangles, based on iterative perturbations and testing of the model.

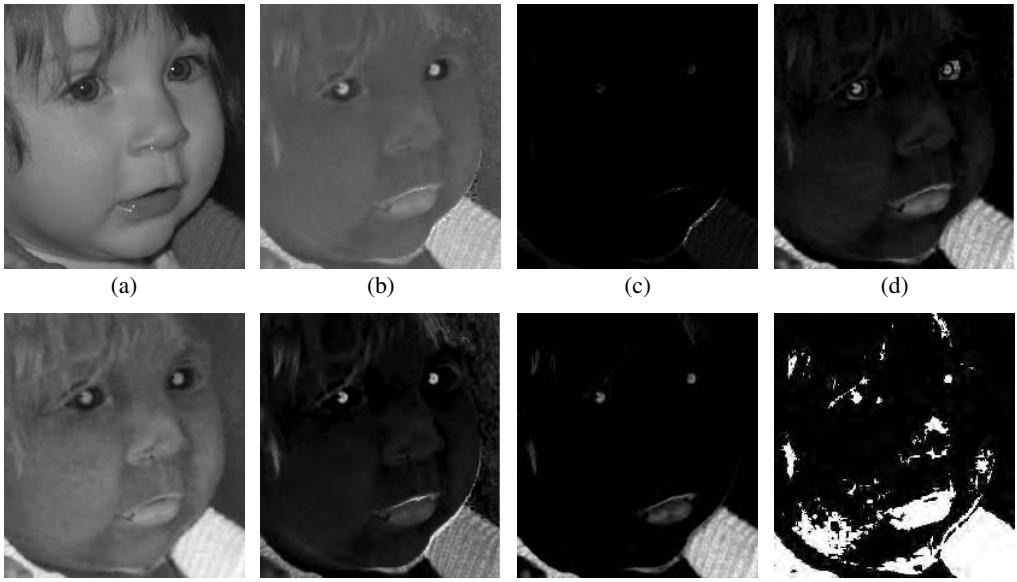


FIGURE 16.7

Different redness maps: (a) original image, (b) Cb channel image obtained by Equation 16.2, (c) redness obtained by Equation 16.6, (d) redness obtained by Equation 16.10, (e) red-enhanced map obtained by Equation 16.11, (f) redness obtained by Equation 16.12, (g) *RedLum* obtained by Equation 16.17, and (h) mask of red pixels defined by conditions in Equation 16.13.

16.3 Methods for the Reduction of the Red-Eye Search Space

There are several methods that can reduce the search area. These methods are often general purpose methods because they do not necessarily need any particular knowledge about red eye. These modules can be combined in several ways. One possible approach (Figure 16.8) could be to detect faces first, then to detect eyes in the detected faces, and finally to determine whether the eyes suffer from red-eye effects. Missing a face in these approaches results in undetected eyes. Thus, the performance of such red-eye detection strongly depends on the face detection performance. In addition, most face detection algorithms can not detect rotated, not in-plane, or partially occluded faces. In addition, face detection is usually computational and memory demanding. Other approaches to detecting the red-eye candidates process the image by localizing all the red regions and then re-evaluating these candidates through successive verifications based on color, contrast, geometry, presence of skin, sclera and glint.

16.3.1 Skin and Sclera Detection

Skin information can be used for detecting faces and/or checking the correctness of a candidate red eye. Many different methods for discriminating between skin pixels and nonskin pixels are available in the literature. These can be grouped in three types of skin

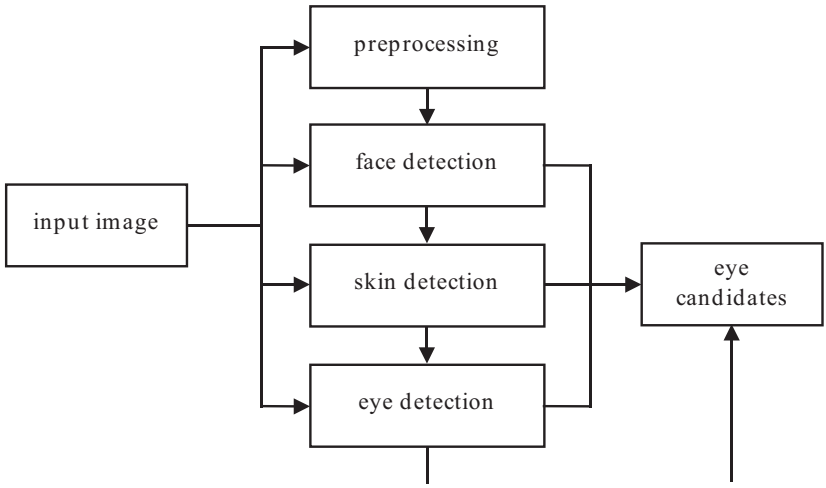


FIGURE 16.8

Flowchart of possible reduction space modules.

modelling: parametric, nonparametric, and explicit skin cluster definition methods. The parametric Gaussian models [32] assume that skin color distribution can be modelled by an elliptical Gaussian joint probability density function. Nonparametric skin modelling methods estimate skin color distribution from the training data without deriving an explicit model of skin color; an example is the histogram-based nonparametric skin model [33]. The simplest and most often applied method is to build what is called an *explicit skin cluster* classifier which defines the boundaries of the skin cluster in certain color spaces [34], [35], [36], [37], [38], [39], [40]. The underlying hypothesis of methods based on explicit skin clustering is that skin pixels exhibit similar color coordinates in a properly chosen color space. The main difficulty in achieving high skin recognition rates and producing the smallest possible number of false positive pixels is related to the definition of accurate cluster boundaries through simple and often heuristically chosen decision rules.

Smolka et al. [10] detected the skin tone regions combining the outputs of threshold operations in rgb and HSV color spaces as follows:

$$r \in [35,55]\,;\; g \in [25,38]\,;\; H \in [0,50] \cup [340,360]\,;\; S > 0.2;\; V > 0.35 \qquad (16.18)$$

The face-like regions are then obtained removing small regions and filling gaps and holes in the skin-tone areas, using morphological closing and opening operators.

One of the advantages of using color information as the key feature to detect and localize possible faces in a scene is that color is generally invariant to rotations, translations, and scale changes. However, difficulties in effective face detection and localization arise when the image is acquired under varying and uncontrolled lighting conditions. To solve the problem, Gasparini and Schettini [23] suggested pre-processing the image by using a color balance algorithm to discount illuminant colors.

Several approaches use skin tone to limit a posteriori the false positive red-eye candidates in a map of red patches. Luo et al. [41] performed several local verification tests after a preliminary screening where global candidate red-eye areas have been detected mainly on

the base of redness and contrast. In particular, each candidate red-eye area that does not correspond to an isolated nonskin tone 'island' in a skin-tone region is filtered from the final red-eye map. They also considered the proportion of nonskin tone and skin-tone pixels in a target area as the discriminant feature. Any skin-tone classification can be used in their approach. Wu [42] first detected red-eye pixels through successive thresholds of a redness map, followed by grouping contiguous pixels and rejecting those that are not circular. These patches can be further reduced, verifying first their proximity to a skin region and then to a white region corresponding to the sclera. The threshold skin-color values are expressed in the HSV color space as follows:

$$S \in [5,80]\,;\; V \in [20,80]\,;\; H \in [-80,50] \tag{16.19}$$

Volken et al. [25] first detected bright red pixels using proper thresholding operations in the CIELAB color space and then searched for white (sclera) and skin-tone pixels. The skin pixels were obtained via simple thresholding. By counting the percentage of sclera and skin in two different crops around each candidate red region, red-eye and non-red-eye areas can be classified.

16.3.2 Face Detection

One of the advantages of using face detection as a preliminary step for the reduction of the search space is that together with the information on eyes' position it is possible to estimate the eye size. Then, other geometric constraints, such as the distance between the two eyes, approximate size of the sclera and glint can be applied. Any system that permits the detection of faces in a digital image can be applied. For instance, a neural network approach can be used, as described by Rowley et al. [14]; or a wavelet approach as proposed by Schneidermann et al. [30]. Other commonly used face detectors follow the rationale behind the approach proposed by Viola and Jones [15]. Their algorithm uses a multiscale, multistage classifier which operates on image intensity information. Of importance at this stage is that the detection of faces happens fully automatically, the detection rate is reasonably high, and that the false positive rate (nonface regions marked as a face) is reasonably low. The rotation invariance of each detector varies widely and should be insured by external means such as pre-rotation of the image during the training of classifiers followed by normal face detection.

Gaubatz and Ulichney [8] used a face detector based on the Viola-Jones classification algorithm. Then, they searched for the red eyes in the upper half of the obtained face bounding box. They use color, intensity, and size information obtained from the face detector to detect red eyes. In fact, because the ratio between eye and face sizes is similar in most faces, the face detector gives an indication of expected eye size in a detected face. Joffe [21] applied an eye detector to locate red pupils, followed by a wavelet-based face-detector approach similar to the one developed by Schneidermann and Kanade [30] in order to limit the number of false detections based on the fact that a red eye must be part of a face. The order of the detection steps was given by the higher processing speed of the eye detector compared to that of the face detector.

16.3.3 Eye Detection

There are several approaches that can be applied for detecting eyes. Similar to the detection of faces, it is important to have an approach that works fully automatically, has a high recognition rate, and a low false-positive rate. Zhang et al. [22] adopted a human eye classifier to confirm each candidate red-eye region found in a previous step based on a sequence of heuristic algorithms. Their classifier was trained based on the Adaboost method [15] using a large training set of 20×20 gray images. Since eyes were detected based on grayscale images, it prevented the difficulty in collecting numerous red-eye images. As there are usually a few dozen candidate red-eye regions in an image, the difficulty of red-eye verification here is less compared to performing red-eye detection in the whole image.

As a first step of his automatic red-eye correction system, Joffe [21] adopted an automatic red-eye detection module. Having sufficient positive examples of red eyes and negative examples of non-red eyes, the detector is automatically trained by these data, applying machine learning methods. This classifier looks at an image patch of a fixed size (in this case 11×11) and determines whether or not it represents a red eye. Each image patch is described by a set of features having a finite set of possible values. This representation lends itself to a boosting training method. To detect red eyes in an image, the classifier scans the image considering the 11×11 image patches at all possible positions. Since red eyes may have different size, the detector is also applied to scaled versions of the image.

Held [20] developed a multistage eye detector extending the approach described by Benn et al. [43]. In these works, eye centers are detected based on the Hough transform. It was shown that the large requirements on memory and processing speed of this transform can be greatly reduced by using the so-called gradient-decomposed Hough transform. In order to enhance facial features, preprocessing steps such as histogram normalization, local-contrast enhancement, and red-enhancement can be applied. Then, coarse eye detection and a refining detection procedure, both based on the Hough transform, can be used to be finally followed by some heuristics to avoid unreasonable eye candidates.

Wan et al. [24] adopted an active appearance model to locate the eye regions. This model is trained using the relationship between model parameter displacements and the residual errors between the training image and a synthethesized example. In the detection stage, the method measures the current residual errors and predicts changes of the parameters that lead to a better fit. A good overall match is obtained in a few iterations, even from poor starting estimates.

16.4 Red-Eye Correction

Correction is seemingly much easier than detection, but actually it is difficult to generate a perfect picture. An important issue with red-eye correction is that it might result in image degradation. This is why most photo retouching software solutions propose semiautomatic approaches. If the system is completely automatic, the risk of further degrading the image has to be taken into account. Several aspects must be considered; in particular, the cor-

rect color of the pupil is not the color of the iris but it should be essentially neutral. The correction should also preserve the glint, which is the specular reflection of the flash in the eye, that gives the eyes a natural aspect and reflects the subject's personality. Finally, the correction should also be smooth enough to avoid abrupt color transitions between the corrected and the uncorrected areas.

A simple neutral correction was performed by Patti et al. [4], [13], where all the pixels belonging to the final pupil mask, evaluated as described in Section 16.3, assume gray values of 80% of their original luminance. Using this experimentally evaluated scaling factor preserves the glint in the pupil. Before applying such color correction, a morphological pruning is performed on the mask in order to avoid the correction of nonpupil regions such as eyelids. Other approaches adopt very simple corrections. For example, Wu [42] simply substituted the red color of the defective eyes by black. Corrections of this type could be very dangerous leading to a processed image which is even worse than the defective original. Having cleared the drawback of too simple removal, a particular effort to minimize invasive correction is seen in the work by Gabautz et al. [8]. Once a candidate red eye is located, the color of the excessively red pixels within an eye-sized neighborhood is removed. A pixel is considered excessively red if its redness, defined according to Equation 16.6, exceeds a certain threshold [28]. By handling artifacts in this manner, reddish pixels outside the actual pupil can be desaturated, thus introducing other undesirable effects. Moreover, a patch of eye pixels devoid of color can look more unnatural than a patch of slightly reddish pixels. Finally, a hard boundary between corrected and uncorrected pixels can have a displeasing look. To overcome these problems, a correction factor $p(i,j)$ for a pixel located at the pixel location (i,j) in the corrective mask is introduced to control the desaturation process. This correction factor removes non-red eye pixels from the mask which are associated with significant changes in luminance due to the presence of the eyelids. This factor also smooths the perceived edges due to hard decision boundaries, tapering these boundaries. The final correction, performed in the YCbCr color space, can be obtained as:

$$\begin{aligned} Y(i,j)_{corrected} &= Y(i,j)_{original} \\ C_r(i,j)_{corrected} &= (1-p(i,j)) * C_r(i,j)_{original} \\ C_b(i,j)_{corrected} &= (1-p(i,j)) * C_b(i,j)_{original} \end{aligned} \tag{16.20}$$

Hardeberg [17] proposed to smooth or fuzzify the red pupil mask to achieve a softer correction that should appear more natural. In doing so, the target color for the corrected red eye was defined in the CIELAB color space. The color components a^* and b^* determining the hue and saturation are set to zero in order to achieve gray-scale effects. The lightness component L^* was then determined by stretching out the lightness of the original image so that its minimum value becomes black (or almost black) and its maximum value remains constant:

$$\begin{aligned} L^*_{corrected} &= \frac{\max L^*}{(\max L^* - \min L^*)} (L^* - \min L^*) \\ a^*_{corrected} &= 0 \\ b^*_{corrected} &= 0 \end{aligned} \tag{16.21}$$

The new color for each pixel $i_{corr}(i,j)$ is determined as a combination of the original color $i_{or}(i,j)$ and the target color $t(i,j)$ by weighting the fuzzy mask values $m(i,j)$ as follows:

$$i_{corr}(i,j) = t(i,j) * m(i,j) + i_{or}((i,j) * (1-m(i,j))) \tag{16.22}$$

(a) (b) (c)

FIGURE 16.9 (See color insert.)

Red-eye correction process: (a) the original red-eye artifact, (b) the final correction, and (c) the corresponding smoothed pupil mask.

Since the calculation of Equation 16.22 is carried out in CIELAB space, the pixels need to be converted back to an RGB representation to complete the correction process. Operating in the CIELAB color space ensures that the glint remains at the same level of lightness in the corrected image while removing the unwanted reddish hues. Figure 16.9 shows an example of red-eye correction and the corresponding fuzzy mask.

In the color-correction step of their automatic procedure, Smolka et al. [10] replaced the red-eye pixel (i, j) with an achromatic pixel with intensity $I(i, j) = mean(G, B)$.

Another simple color correction method was proposed by Held [20] who used a correction mask obtained by smoothing the red pupil binary map with a Gaussian filter. This correction mask $m(i, j)$ can also be considered as the probability of a pixel (i, j) belonging to a red-defect region. Pixels approaching to the eye boundaries receive a gradually decreasing probability, allowing for smooth transitions between correct and uncorrected regions. Using such probabilities, the red-eye defects can be corrected as follows:

$$R_{new}(i, j) = R(i, j) - m(i, j) * (R(i, j) - \min(G(i, j), B(i, j))) \tag{16.23}$$

Thus, if the probability of a pixel belonging to a red-eye defect is zero, then the correction factor is zero, as well. Otherwise, the red channel will be pulled toward the minimum of both the blue and green channels. To avoid an unpleasant color shift, the correction is adjusted in case of too large difference between the blue and green channels, as indicated by the following equations:

$$\text{if } G > R_{new} \quad \text{then } G_{new} = (R_{new} + B)/2 \tag{16.24}$$

$$\text{if } B > R_{new} \quad \text{then } B_{new} = (R_{new} + G)/2 \tag{16.25}$$

If $m(i, j) = 1$, then Equations 16.24 and 16.25 lead the correction through the neutral axes, as in the case of $R = G = B = \min(R, G, B)$. This correction preserves the glint highlights and replaces the red defects with a neutral or nearly neutral color with minimum lightness, leading to a rather darkish pupil of an undefined color.

Zhang et al. [22] developed an adaptive correction algorithm which decreases the value of the red channel and increases the value of green and blue pixel components in order to remove the red-eye effects and preserve the glint. The correction consists of two operations at the pixel level; brightness adjustment and contrast adjustment, defined as follows:

$$\text{AdjustBrightness}(C, K_1) = (255 \cdot 0.4K_1 + C) \tag{16.26}$$

$$\text{AdjustContrast}(C, K_2) = (C + (C - meanL)K_2) \tag{16.27}$$

where C is the pixel color in R, G or B channel, K_1 is an adaptive parameter in the range $[-1,1]$ used to control the brightness adjustment, K_2 is a constant, and $meanL$ is the average luminance of the red-eye region. The above functions are used in each pixel location of the red-eye region:

$$\begin{aligned} RR &= \text{AdjustBrightness}(R, -factor_B) \\ GG &= \text{AdjustBrightness}(G, factor_B/3) \\ BB &= \text{AdjustBrightness}(B, factor_B/3) \end{aligned} \tag{16.28}$$

$$\begin{aligned} R_{new} &= \text{AdjustContrast}(RR, -10) \\ G_{new} &= \text{AdjustContrast}(BB, -10) \\ B_{new} &= \text{AdjustContrast}(BB, -10) \end{aligned} \tag{16.29}$$

The parameter $factor_B$ is evaluated adaptively on the whole red-eye region as follows:

$$factor_B = 3.2(AvgR - \max(AvgG, AvgB))/255 \tag{16.30}$$

where $AvgR$, $AvgG$, $AvgB$ are the average of red, green, blue components in the red-eye region, respectively, excluding the pixels that do not really appear red, as guaranteed by $R/(R+G+B) > 0.35$. According to these equations, the value of the red component is decreased whereas the values of the green and blue components are increased. Meanwhile, the contrast in the red-eye region is slightly decreased.

Major attention in red-eye correction was given in the last years by Ulichney and Gabautz [44], Willamowsky et al. [26] and Marchesotti et al. [27]. Most of the previous approaches adopt a binary decision and apply either no correction or the maximum possible correction. In difficult cases, hard approaches make significant mistakes, often missing red eyes or introducing disturbing artifacts in non-red-eye regions. The resulting correction seems often unnatural, producing a reddish ring around the corrected zone of the eye. In order to obtain more natural correction, previous approaches usually blur detected red-eye regions. Ulichney and Gabautz considered the correction of red-eye defects with a perceptual-based approach. They investigated how much the average luminance can be lowered in the desaturating correction step. In fact, the natural solution of desaturating the red-eye defect usually leaves the region gray, and much lighter than what would appear natural. In order to solve this problem, the authors performed a perceptual study in order to find the most visually pleasing target luminance for corrected images. Controlled subjects were chosen to evaluate a series of sample red-eye corrected images. The variable in this experiment was the average luminance of the desaturated pixels. The results of this experiment were analyzed and correlated with the test data to calculate the target luminance $f(Y)$:

$$f(Y) = 0.167Y_{av} + 11.523 \tag{16.31}$$

The whole procedure can be summarized as follows:

$$\begin{aligned} Cb_{new} &= Cb * (1-M) \\ Cr_{new} &= Cr * (1-M) \\ Y_{new} &= (1-M) * Y + M * f(Y)/Y_{av} \end{aligned} \tag{16.32}$$

where M is a tapered mask that extends over 10% of the diameter of the eye region to avoid artificial hard edges in the eye.

Willamowsky et al. [26] adopted a different approach to obtain visually pleasing correction. In particular, they were interested in reducing the corrections relative to false detection in face regions, accepting corrections in the background. In a detection stage, each pixel was assigned to a red-eye probability P (see Section 16.3) and then corrected accordingly by modifying only the red component of the image as follows:

$$R_{new} = (1-P)\,R_{old} + P\,(G+B)\,/2 \tag{16.33}$$

The probability P can be adjusted to improve over the hard case results according to the importance attributed to the correction of red eyes relative to face and background errors.

Marchesotti et al. [27] proceeded further in this direction. They proposed and compared three correction methods according to their image degradation risks. Depending on degradation risks, perceptual quality, and red-eye detection confidence, the correction strategy previously estimated as in Reference [26] is selected by an adaptive system. The three methods considered are soft-pupil desaturation (SPD), template-based pupil correction (TPC), and template-based pupil correction with glint detection and insertion (TPCG).

The SPD method combines the color channels of the original image with the grayscale version $g(i,j)$ of the image using the red-eye probability map $P(i,j)$:

$$I_c\,(i,j,k) = g\,(i,j)\,P\,(i,j) + I\,(i,j,k)\,(1-P\,(i,j)) \tag{16.34}$$

with $k = R,G,B$ denoting the color channel, and I_c and I denoting the corrected and the original image, respectively. The TPC method is a chromatic correction that smooths the corrected color components on the pupil boundaries and palliates the JPEG artifacts. The correction intensity over the pupil is modelled using a Gaussian function. The main advantage of TPC over SPD is that it allows for smoother transitions between the pupil and the iris, especially in the case of red-eye binary maps. The most interesting and innovative contribution in this work comes from the TPCG method which performs glint absence detection and insertion. Glint occurs at the cornea, which has a fixed curvature. When the luminance is too flat over the whole pupil, then the specular reflection is missing, thus giving a dead-eye effect to the eye. To avoid these effects, a glint can be inserted whenever its absence is detected. Therefore, a glint-probability map is obtained using locally high luminance values as follows:

$$Pg\,(i,j,k) = \frac{1}{Z}\left(q\,(i,j) - \frac{\sum_{(l,m)\in R(i,j)} q\,(l,m)}{|R\,(i,j)|}\right) \tag{16.35}$$

where Z is a normalization factor, $q(i,j)$ is the pixel luminance values, $R(i,j)$ is a circular region centered at the location (i,j), and $|R(i,j)|$ is the number of pixels in the region. The map is further intersected with a region larger than the assumed pupil region, taking into account that the glint is often outside the pupil but inside the cornea. If no candidate glint region is present within this area, a glint should be inserted artificially. The authors model the glint to be inserted by a Gaussian similar to one proposed for the pupil correction, with the same position, orientation, and size proportional to the pupil size. To select the type of correction (SPD, TPC, TPCG) to be applied, an image degradation risk is evaluated for each correction method. It was found that TPC has a higher degradation risk than SPD because it boosts the correction in the center of the assumed red-eye region. Therefore, TPC can

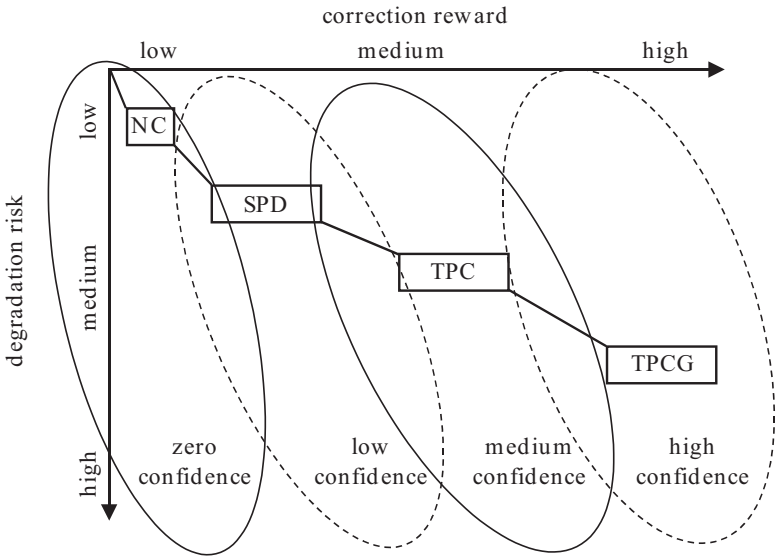


FIGURE 16.10

Comparison of different correction techniques; no correction (NC) versus SPD, TPC and TPCG in a common reward-risk space (as in Reference [27]).

increase the modification of a non-red-eye patch. The SPD works differently because it is based on the predefined red-eye probability P. It was further observed that the degradation risk in TPCG increases due to false negatives in the glint detection process that can lead to a double glint, and the insertion of a glint in a noneye patch. These considerations may also take into account that these correction techniques will have different impacts on different errors. They distinguish between errors in the background (less damaging) or in the face. Face errors are also of two types: non-red-eye regions detected as red-eye candidates (e.g., mouth, nose) or incorrectly segmented red-eye regions (the region overflows the eye area). In contrast to the degradation risk, a correction reward which is related to the user preferences on the corrected image compared to the original image can also be evaluated. Generally, SPD and TPC corrections are preferred over no correction of true red-eyes. TPC is either visually similar to or better than SDP. Its advantages consist of the smooth transition between pupil and iris and lower ring effect in case of undercorrection. Experiments show that inserting an artificial glint in a dead-eye (TCPG idea) brings a potentially high reward since it greatly improves perceptual quality of the red-eye region.

Generally, the risk increases with the reward, as depicted in Figure 16.10. Better corrections correspond to a higher degradation risk (TPCG), lower risk to a less pleasing correction (SPD). A compromise solution between risk and reward can be obtained with TPC. The analysis can be used to guide the choice of the method by the confidence that a given region contains a red eye, as well as to propose the correction strategy. Namely, when the confidence is high, a more rewarding strategy (e.g., TPCG) should be chosen. On the other hand, a conservative method such as pupil desaturation (SPD) should be preferred for low confidences where the degradation risk should be reduced. For a medium confidence, TPC should be selected as a compromise between reward and degradation risk.

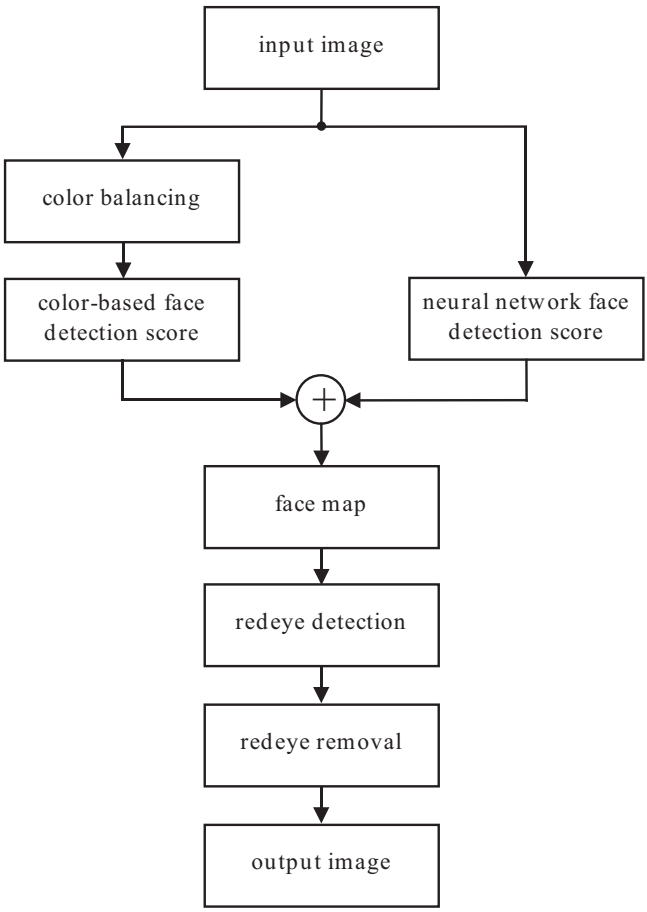


FIGURE 16.11

Flowchart of an automatic procedure for redeye removal.

Another interesting solution to the re-coloration problem was proposed by Miao et al. [45]. They corrected red-eye defects automatically by combining flash and nonflash digital images. Obviously this method is suitable only within digital cameras that support continuous shooting. This kind of camera would eliminate red eyes immediately after image capture. The basic idea is to replace the red eye in the flash image with the true eye color obtained from the nonflash image.

16.5 A Complete Procedure for Automatic Red-Eye Removal

This section details an automatic detection and correction solution which is capable of removing the red-eye effects in images of unknown origin (unknown imaging systems, unknown lighting conditions) such as images downloaded from the web or received by friends through e-mail and cell phones. The proposed method is modular (Figure 16.11)

so that each processing step can be removed, substituted with a more efficient one, or new modules can be inserted. The procedure starts correcting the color photo using a smart algorithm [46]. This phase facilitates the subsequent steps of processing and improves the overall appearance of the output image. In subsequent step, the method looks for red eyes within the most likely face regions. The localization of these candidate regions is obtained by using a scoring process to combine the results of a color-based face detector and a face detector based on a multi-resolution neural network, working only on the intensity channel. In the final phase, red eyes are automatically corrected, aiming at the natural appearance of the resulting image.

16.5.1 Image Color Correction

Our color correction method [46] can reliably classify and remove color cast (i.e., a superimposed dominant color) in a digital image without any a priori knowledge of its semantic contents. Using simple image statistics, an employed cast detector classifies the input images as presenting no cast, evident cast, ambiguous cast, a predominant color that must be preserved (such as in underwater images or single color close-ups), or as unclassifiable. A cast remover, a modified version of the white balance algorithm, is then applied in cases of evident or ambiguous cast. Since the color correction is calibrated on the type of the cast, even ambiguous images can be processed without color distortion.

16.5.2 Face Detection

Face detection in a single image is a challenging task because the overall appearance of faces ranges widely in scale, location, orientation and pose, as well as in facial expressions and lighting conditions [14], [47]. Thus, our objective is not to determine whether images contain faces, but instead to generate a score map which reflects the confidence of having a facial region that might contain red eyes. This score map is obtained by combining the scores from a color-based face detector and a neural-network face detector, working only on the intensity channel.

First a skin detector, for instance, one of those described in Section 16.4, is applied to the color balanced images. The final skin-like region mask used to segment the original image is obtained after morphological operators have been used to fill holes and gaps usually corresponding to eyes, noses, or lips, and remove small areas that are unlikely to be faces. Figure 16.12a shows an input image, with its corresponding skin mask shown in Figure 16.12b. The skin-based candidate faces are obtained through a score map $Score_{CB}$, shown in Figure 16.12c, which is generated taking into account: i) $Score_D$ which reflects discontinuities in the intensity channel that may correspond to eyes, nose and lips, ii) $Score_S$ which reflects the shape of the region that should fit an oval; and iii) $Score_A$ which reflects the ratio between the area of the region and the area of its filled version. These scores are all normalized to one, providing the final score as follows:

$$Score_{CB} = Score_D + Score_S + Score_A \tag{16.36}$$

Exploiting only intensity information, to output a score map reflecting the confidence of the presence of faces in the input image, we have also trained an auto associative neural

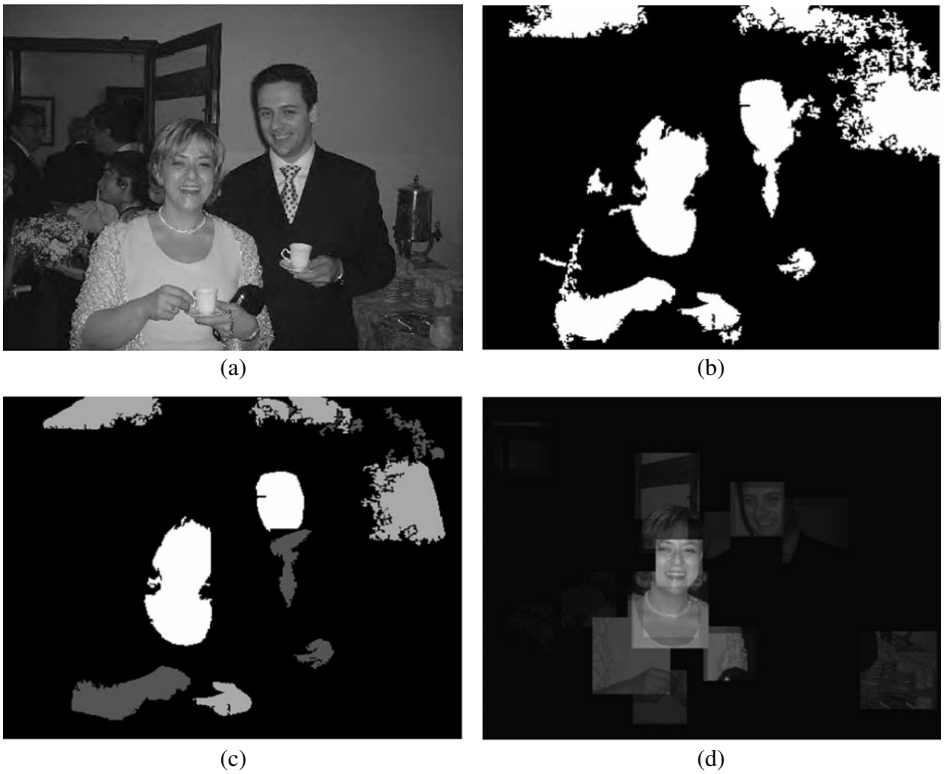


FIGURE 16.12

Intermediate score maps: (a) input image, (b) the corresponding skin mask, (c) score of the color-based face detector, and (d) score of the neural network face detector.

network [47]. This network has three layers, where each pattern of the training set is presented to both the input and the output layers, and the whole network has been trained by a backpropagation sum square error criterion on a training set of more than 150 images considering both face images (frontal faces) and images with no faces. The network processes only the intensity image, so that the results are color independent. To locate faces of different sizes, the input image is repeatedly scaled down by a factor of 15%, generating a pyramid of subsampled images. A window of 20×21 pixels is applied at every point of each scaled image. This 20×21 portion of the image, properly equalized, forms the input of the network. The output is obtained with a feedforward function, and the root mean square error ε between output and input is calculated. The smaller this value is, the higher the probability of having detected a face becomes. The true-positive (TP) versus the false-positive (FP) detection rate can be plotted varying the error ε. The score map of the input image Score_{NN} evaluated as 1-FP(ε) is obtained by collecting the confidence of being a facial region of each single 20×21 window in the pyramid with root mean square error ε. An example of this score map is depicted in Figure 16.12d.

The final score map is obtained by combining these scores:

$$Score = (Score_{CB} + Score_{NN})/2 \tag{16.37}$$

and normalizing the result so that it ranges between zero and one. The most likely facial

(a) (b)

FIGURE 16.13

Achieved face detection results: (a) final score map of the input image of Figure 16.12a, and (b) the most likely facial regions for score ≥ 0.6.

regions correspond to the regions with the highest values of the score. In this work we consider all the regions with *Score* greater than 0.6 as faces. Figure 16.13a shows the final score map corresponding to the input image shown in Figure 16.12a. The most likely facial regions are depicted in Figure 16.13b. Even if both face detectors show high values of false positives to guarantee a good face-detection rate (especially when applied to search faces different in scale, location, orientation, pose, and expression), the union of the two methods through the definition of a score process can improve the performance of the final system significantly.

16.5.3 Red-Eye Detection

Within the most likely face regions the algorithm looks for red eyes, applying a color analysis step followed by geometric analysis.

- *Color analysis*: The algorithm looks for the regions with high value of redness

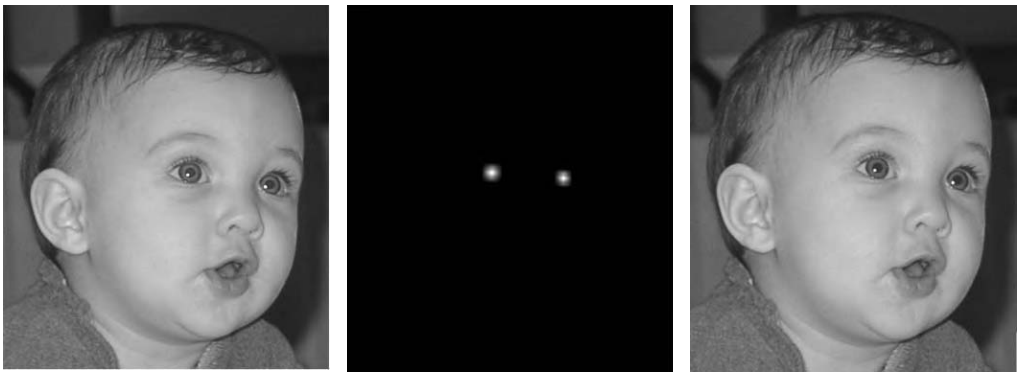
$$redness = (2R - (G + B))/R. \tag{16.38}$$

  Basically, the color image is converted into a monochrome (redness) image, in which a red eye is highlighted as a bright spot.

- *Geometric analysis*: In order to limit the number of false hits, the algorithm exploits a number of geometric constraints such as the ratio between the red-eye area and the face bounding box area (i.e., $P > 1.5\%$), the ratio between minimum and maximum dimension of the eye (i.e., $F > 0.4$), and the roundness ratio between the red-eye area and the area of the ellipse that has the same second-moment as the region (i.e., $O > 0.7$).

16.5.4 Red-Eye Removal

The last step of the algorithm is the color correction of the detected red-eye artifacts. If a pixel has been detected as belonging to a red eye, it is replaced with a substantially



(a) (b) (c)

FIGURE 16.14 (See color insert.)

Correction process: (a) original image, (b) the smoothing mask of the red-eye areas, and (c) the corrected image.

monochrome pixel, applying the following equation to RGB channels of a candidate region:

$$R_{new} = (R_{old} * (1 - Mask_{smooth}) + Mask_{smooth} * R_{mch}) \tag{16.39}$$

The coordinates of the monochrome pixel are R_{mch}, G_{mch} and B_{mch}, evaluated considering the intensity equal to the mean of G and B components. The color correction is weighted using the smoothing mask ($Mask_{smooth}$) of the incriminated area to avoid unnatural transitions between corrected and uncorrected parts of the eyes. This color pixel correction preserves the bright specular reflection of the eyes. Figure 16.14 shows the original image together with the smoothed area corresponding to the red eyes, and the resulting image with corrected red-eye artifacts via Equation 16.39.

16.6 Evaluation and Conclusion

As discussed in this chapter, most solutions for fixing red eyes usually perform detection and correction steps. The detection phase is considered as a more difficult analytical and computational problem. Once the precise pixels in the input image that contain red-eye artifacts are properly identified, the second phase is applied and the offending pixels are corrected. Since it is rather difficult to generate a visually pleasing corrected picture, the overall performance of a red-eye removal method should be evaluated considering two distinct evaluations, one for red-eye detection and the other for red-eye correction. Results of the methods available in the literature are difficult to compare since there is no publicly available red-eye test image database. Without knowledge of the variability in size, intensity and quality of test images, different methods cannot be directly compared.

Usually, any detection method is evaluated in terms of true-positives, false-negatives, and false-positives. In case of red-eye detection, the criterion based on true-positives measures the number of red eyes correctly identified. False-negatives denote the number of missed

red eyes whereas the criterion based on false-positives indicates the number of false alarms, given by non-red-eye regions incorrectly detected as red eyes. These are absolute measures that need to be somehow related to the image set to be processed. Some works (e.g., Reference [26]) report the number of test images and these absolute values. More common is the evaluation of the true positive rate, known also as detection rate that corresponds to the number of correctly classified red eyes with respect to the total number of true red eyes. This value alone is not significant and needs to be considered together with a normalized version of the false alarms. Unfortunately, normalizing the false positive is less intuitive. Generally, when a percentage of false positives is reported, it is referred to the total number of detections of the system. Another possibility is to represent the number of false positives per image.

Several authors (e.g., Luo et al. [9]) realized that if false alarms from red-eye detection are desaturated, then the final result is not perceivable by the end-user. This is generally related to the false alarms not located in the region of interest of known colors. This helps to further reduce the effective false alarm artifacts of the system.

Other works (e.g, Reference [10]) classified the corrected images into four groups in order to evaluate the overall red-eye detection / correction performance. The first group consists of images with all detected and corrected red eyes, without any false corrections. The second group consists of images in which all red eyes were detected and corrected together with some non-red-eye regions. The images with some undetected red eyes, but without false corrections, fall into the third class. The last group contains images with some undetected red eyes and some false corrections. Taking into account that the false corrections are sometimes difficult to remark and the whole correction process can be treated as successful, each group of images can be further divided into two subgroups in order to provide the information whether the overall appearance of the image after the correction process was essentially improved or the method failed to increase the image quality.

A great effort to distinguish between significant and less significant false detections can be seen in the work by Willamowski et al. [26]. They performed a user experiment, where participants compared and ranked different corrected images, with artifacts introduced at different locations. This experiment showed that false positives on faces are much more damaging than false positives on the background. Background errors only have some secondary importance when the object on which the error is located is familiar to the participants or if its natural color is obvious to them. They trained a first classifier to distinguish red-eye patches from face errors and a second classifier to distinguish red-eye patches from background errors. This can penalize face errors more than background errors. Therefore, it can be a good idea to evaluate the performance using criteria based on i) true-positives and false-negatives expressed for red eyes, ii) true-negative and false-positive face errors, and true-negative and false-positive background errors.

To also take into account the effect of the correction, usually the percentage of the test images that provides visually better results than the original images is reported without distinguishing if the damage is due to a false-positive detection or to a degradation due to the correction. Another possible damage could be due to the correction of only one of the two red eyes, corresponding to a false-negative, or due to a different correction of a pair of eyes of the same person.

Determining the most visually pleasing target luminance for corrected images, as explored by Ulichney and Gaubatz [44], can be the way to achieve the desired image quality. Of significant importance for what concerns the correction performance is the analysis performed by Marchesotti et al. [27]. In their work, different correction techniques are compared and evaluated in a common reward-risk space. They also showed where to search for a correction method in that space, given the degree of confidence of being a red eye and taking into account the different incidences of false alarms in face regions or background regions.

References

[1] C.M. Dobbs and R. Goodwin, "Localized image recoloring using ellipsoid boundary function," U.S. Patent 5 130 789, July 1992.

[2] P. Benati, R. Gray, and P. Cosgrove, "Automated detection and correction of eye color defects due to flash illumination," U.S. Patent 5 432 863, July 1995.

[3] P. Benati, R. Gray, and P. Cosgrove, "Automated detection and correction of eye color defects due to flash illumination," U.S. Patent 5 748 764, May 1998.

[4] A. Patti, K. Kostantinides, D. Tretter, and Q. Lin, "Apparatus and a method for reducing red-eye in a digital image," U.S. Patent 6 016 354, January 2000.

[5] J. Schildkraut and R. Gray, "Computer program product for redeye detection," U.S. Patent 6 252 976, July 2001.

[6] J. Schildkraut, R. Gray, and J. Luo, "Computer program product for redeye detection," U.S. Patent 6 292 574, September 2001.

[7] J. Wang and H. Zhang, "Apparatus and a method for automatically detecting and reducing red-eye in a digital image," U.S. Patent 6 278 491, August 2001.

[8] M. Gaubatz and R. Ulichney, "Automatic red-eye detection and correction," in *Proceedings of the IEEE International Conference on Image Processing*, Rochester, NY, USA, September 2002, vol. 1, pp. 804–807.

[9] H. Luo, J. Yen, and D. Tretter, "An efficient automatic redeye detection and correction algorithm," in *Proceedings of the 17th International Conference on Pattern Recognition*, Cambridge, UK, August 2004, vol. 2, pp. 883–886.

[10] B.Smolka, K. Czubin, J.Y. Hardeberg, K.N. Plataniotis, M. Szczepanski, and K. Wojciechowski, "Towards automatic redeye effect removal," *Pattern Recognition Letters*, vol. 24, no. 11, pp. 1767–1785, July 2003.

[11] J.Y. Hardeberg, "Red-eye removal using color image processing," U.S. Patent 6 728 401, April 2004.

[12] "Redbot automatic red eye correction," *http://redbot.net/*, Redbot Hewlett-Packard Labs.

[13] A. Patti, K. Kostantinides, D. Tretter, and Q. Lin, "Automatic digital redeye reduction," in *Proceedings of the IEEE International Conference on Image Processing*, Chicago, IL, USA, October 1998, vol. III, pp. 55–59.

[14] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–28, January 1998.

[15] P. Viola and M.J. Jones, "Robust real-time object detection," Tech. Rep. CRL 2001/01, Compaq Cambridge Research Laboratory, Cambridge, MA, February 2001.

[16] R. Ulichney, M. Gaubatz, and J. V. Thong, "Redbot - a tool for improving red-eye correction," in *Proceedings of the IS&T/SID Eleventh Color Imaging Conference: Color Science and Engineering Systems, Technologies, Applications*, Scottsdale, AZ, USA, November 2003.

[17] J.Y. Hardeberg, "Red eye removal using color image processing," in *Proceedings of the Image Processing, Image Quality, Image Capture System Conference*, Montreal, Canada, April 2001, pp. 283–287.

[18] J.Y. Hardeberg, "Digital red eye removal," *Journal of Imaging Science and Technology*, vol. 46, no. 4, pp. 375–381, July-August 2002.

[19] K. Czubin, B. Smolka, M. Szczepanski, J.Y. Hardeberg, and K.N. Plataniotis, "On the red-eye effect removal algorithm," in *Proceedings of the First European Conference on Color in Graphics, Imaging and Vision*, Poitiers, France, April 2002, pp. 292–297.

[20] A. Held, "Model-based correction of red eye defects," in *Proceedings of the IS&T/SID Tenth Color Imaging Conference: Color Science and Engineering Systems, Technologies, Applications*, Scottsdale, AZ, USA, November 2002, pp. 223–228.

[21] S. Ioffe, "Red eye detection with machine learning," in *Proceedings of the IEEE International Conference on Image Processing*, Barcelona, Spain, September 2003, vol. 2, pp. 871–874.

[22] L. Zhang, Y. Sun, M. Li, and H. Zhang, "Automated red-eye detection and correction in digital photographs," in *Proceedings of the IEEE International Conference on Image Processing*, Singapore, October 2004, vol. 4, pp. 2363–2366.

[23] F. Gasparini and R. Schettini, "Automatic redeye removal for smart enhancement of photos of unknown origin," *Lecture Notes in Computer Sciences*, vol. 3736, pp. 226–233, December 2005.

[24] J. Wan, X. Ren, and G. Hu, "Automatic red-eyes detection based on AAM," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, The Hague, the Netherlands, October 2004, vol. 7, pp. 6337–6341.

[25] F. Volken, J. Terrier, and P. Vandewalle, "Automatic red-eye removal based on sclera and skin tone detection," in *Proceedings of the IS&T Third European Conference on Color in Graphics, Imaging and Vision*, Leeds, UK, June 2006, pp. 359–364.

[26] J. Willamowski and G. Csurka, "Probabilistic automatic red eye detection and correction," in *Proceedings of the 18th International Conference on Pattern Recognition*, Hong Kong, August 2006, vol. 3, pp. 762–765.

[27] L. Marchesotti, G. Csurka, and M. Bresssan, "Safe red-eye correction plug-in using adaptive methods," in *Proceedings of the International Conference on Image Analysis and Processing*, Modena, Italy, September 2007.

[28] M. Gaubatz and R. Ulichney, "System and method for automatically detecting and correcting red eye," U.S. Patent 0 202 105, October 2003.

[29] R.E. Schapire, "The boosting approach to machine learning: An overview," in *Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA, March 2001.

[30] H. Schneiderman and T. Kanade, "A statistical method for 3D object detection applied to faces and cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, SC, USA, June 2000, vol. 1, pp. 746–752.

[31] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham, "Active shape models — their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, January 1995.

[32] M. Yang and N. Ahuja, "Gaussian mixture model for human skin color and its application in image and video databases," in *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases*, San Jose, CA, USA, January 1995, vol. 3656, pp. 458–466.

[33] M. Jones and J. Rehg, "Statistical color models with application to skin detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Fort Collins, CO, USA, June 1999, pp. 274–280.

[34] J. Kovac, P. Peer, and F. Solina, "2D versus 3D colour space face detection," in *Proceedings of the 4th EURASIP Conference on Video/Image Processing and Multimedia Communications*, Zagreb, Croatia, July 2003, pp. 449–454.

[35] D. Chai and K.N. Ngan, "Face segmentation using skin color map in videophone applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 551–564, June 1999.

[36] R. Hsu, M. Abdel-Mottaleb, and A.K. Jain, "Face detection in colour images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, May 2002.

[37] I.S. Hsieh, K.C. Fan, and C. Lin, "A statistic approach to the detection of human faces in color nature scene," *Pattern Recognition*, vol. 35, no. 7, pp. 1583–1596, July 2002.

[38] S. Tsekeridou and I. Pitas, "Facial feature extraction in frontal views using biometric analogies," in *Proceedings of the European Signal Processing Conference*, Rhodes, Greece, September 1998, vol. I, pp. 315–318.

[39] C. Garcia and G. Tziritas, "Face detection using quantized skin color regions merging and wavelet packet analysis," *IEEE Transactions on Multimedia*, vol. 1, no. 3, pp. 264–277, September 1999.

[40] G. Gomez and E.F. Morales, "Automatic feature construction and a simple rule induction algorithm for skin detection," in *Proceedings of the ICML Workshop on Machine Learning in Computer Vision*, Sydney, Australia, July 2002, pp. 31–38.

[41] H. Luo, J. Yen, and D.R. Tretter, "Detecting and correcting red-eye in a digital image," U.S. Patent 0 21 3476, October 2004.

[42] D. Wu, "Automatic red eye removal," U.S. Patent 0 232 481, October 2005.

[43] D. Benn, M. S. Nixon, and J.N. Carter, "Robust eye centre extraction using the Hough transform," in *Proceedings of First International Conference on Audio- and Video-based Biometric Person Authentication*, Crans Montana, Switzerland, 1997, pp. 3–9.

[44] R. Ulichney and M. Gaubatz, "Perceptual-based correction of photo red-eye," in *Proceedings of the 7th IASTED International Conference on Signal and Image Processing*, Honolulu, HI, USA, August 2005.

[45] X. Miao and T. Sim, "Automatic red-eye detection and removal," in *Proceedings of the IEEE Conference on Multimedia and Expo*, Taipei, Taiwan, June 2004, pp. 1195–1198.

[46] F. Gasparini and R. Schettini, "Color balancing of digital photos using simple image statistics," *Pattern Recognition*, vol. 37, no. 6, pp. 1201–1217, June 2004.

[47] M.H. Yang, D. Kriegmanand, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, January 2002.

17

Image Resizing Solutions for Single-Sensor Digital Cameras

Rastislav Lukac

17.1 Introduction

In digital photography, the desired size of the captured objects can be obtained by adjusting the camera optics and/or using the digital image resizing techniques. In the former case, the distance between the lens and the sensor's surface is adjusted in order to project the light corresponding to the captured visual scene onto a sensor under certain magnification. Increasing the focal length magnifies the objects in the scene while reducing the focal length produces photos with smaller objects. In the latter case, captured images are resized using specialized image processing algorithms.

This chapter focuses on *digital image resizing* solutions for single-sensor cameras. Spatial resolution of digital camera images is often modified by the user or by the application due to its own constraints. For instance, images are *downsampled* to reduce bandwidth for their transmission, to fit more pictures on the storage media, or to obtain the appropriate resolution for printing or publishing them on the Web. On the other hand, images are *upsampled* to overcome the limitations in optical capabilities of inexpensive cameras with fixed-zoom lenses. Specific spatial resolution of the visual input may also be required to achieve the desired performance in some processing steps such as scene analysis and object recognition. Thus, downsampling and upsampling refer to the process of resizing the digital image or changing the number of pixels representing the visual scene. In the literature, image resizing operations, particularly upsampling, are often referred to as *spatial interpolation*.

The chapter surveys in a systematic and comprehensive manner well-known upsampling and downsampling solutions which can utilize various signal processing concepts to produce camera images with the required spatial resolution. Examples presented in the chapter indicate the existence of single-sensor image resizing solutions which are computationally attractive and yield reasonable performance.

The chapter begins with Section 17.2 which briefly discusses the single-sensor imaging fundamentals, emphasizing three image resizing frameworks identified with respect to the demosaicking step in the camera image processing pipeline. Although based on the same principle, these frameworks have different computational complexities and produce resized images of different visual quality.

The main part of the chapter is devoted to image resizing solutions designed within the above frameworks. Such solutions can use various one-dimensional, two-dimensional, nonadaptive, edge-adaptive, componentwise, spectral modelling-based, vector, and compressed domain-based approaches which are introduced and commented upon. In this part of the chapter, image resizing solutions are taxonomized according to the type of resampling operation and the employed signal processing concept. Section 17.3 presents solutions suitable for resizing demosaicked, full-color images. Section 17.4 introduces image resizing solutions which operate directly on acquired sensor readings. Finally, Section 17.5 presents solutions which attempt to perform joint demosaicking and image resizing.

The chapter concludes with Section 17.6 by summarizing the main single-sensor image resizing ideas.

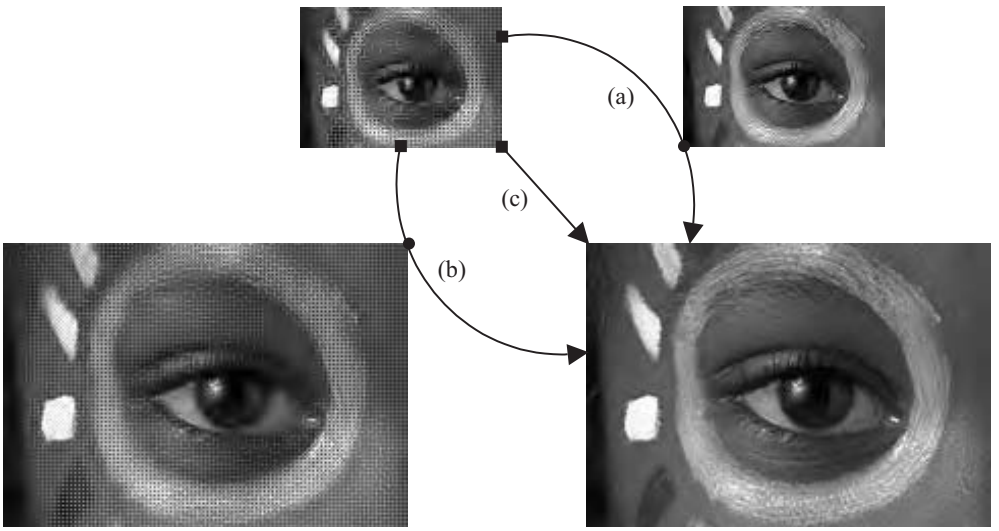


FIGURE 17.1 (See color insert.)

Image resizing approaches for single-sensor digital cameras: (a) demosaicked image resizing, (b) CFA image resizing, (c) joint demosaicking / image resizing. Displayed pictures represent: (top left) CFA image, (top right) demosaicked image, (bottom left) resized CFA image, and (right bottom) resized demosaicked image.

17.2 Single-Sensor Image Resizing Frameworks

As discussed in Chapter 1, *demosaicking* — the process of restoring the full-color information from mosaic grayscale sensor data captured using a single image sensor covered by a *color filter array* (CFA) — constitutes an integral processing step in the single-sensor imaging pipeline. Demosaicking $f_{\varphi}(\cdot)$ transforms a grayscale CFA image z with the resolution of $K_1 \times K_2$ pixels $z_{(r,s)}$ to a color image $\mathbf{x} = f_{\varphi}(z)$ of the same dimensions [1]. Each color pixel $\mathbf{x}_{(r,s)} = [x_{(r,s)1}, x_{(r,s)2}, x_{(r,s)3}]^T$ in the demosaicked red-green-blue (RGB) image $\mathbf{x}$ is comprised of its R ($k = 1$), G ($k = 2$), and B ($k = 3$) component $x_{(r,s)k}$, where $r = 1, 2, ..., K_1$ and $s = 1, 2, ..., K_2$ denote the image row and column, respectively. Thus, demosaicking alters the spectral representation of the input while preserving its spatial dimensions.

On the other hand, *image resizing* is an optional step which can be performed after demosaicking (Figure 17.1a), before demosaicking (Figure 17.1b), or even simultaneously with demosaicking (Figure 17.1c). Image resizing $f_{\phi}^{\tau}(\cdot)$ with a factor of τ preserves the spectral representation and alters the spatial dimensions of the input image from $K_1 \times K_2$ pixels to $\tau K_1 \times \tau K_2$ pixels [1]. Values $\tau < 1$, $\tau = 1$, and $\tau > 1$ indicate downsampling, identity (i.e. no resizing), and upsampling operations, respectively. Thus, image resizing transforms a color image $\mathbf{x}$ to a resampled color image $\mathbf{x}' = f_{\phi}^{\tau}(\mathbf{x})$ or a grayscale image z to a resampled grayscale image $z' = f_{\phi}^{\tau}(z)$. However, combining image resizing with demosaicking in a joint process produces the resampled full-color image $\mathbf{x}' = f_{\varphi\phi}^{\tau}(z)$ from grayscale CFA sensor readings in a single processing step.

Although all three pipelines shown in Figure 17.1 use a $K_1 \times K_2$ CFA image z as the input to produce a resized $\tau K_1 \times \tau K_2$ demosaicked image $\mathbf{x}'$ as the output, the different representation of the signals z and $\mathbf{x}$ and the different order of demosaicking and image resizing operations suggest following design, implementation and performance characteristics:

- The framework shown in Figure 17.1a performs *demosaicking before image resizing*, implying $\mathbf{x}' = f_{\phi}^{\tau}(f_{\varphi}(z))$. Since captured images are usually inspected on bigger displays than the camera's one prior to applying the resizing solution, this approach is commonly used today although the resampling process tends to amplify artifacts introduced during demosaicking.

- Figure 17.1b shows the framework which *resizes the captured image before demosaicking*, denoting that $\mathbf{x}' = f_{\varphi}(f_{\phi}^{\tau}(z))$. Such an order of the processing steps is suitable for cost-effective devices or for fast viewing of photos on the camera display. The limitation of the approach is that the resizing process usually introduces errors which can affect the performance of subsequent demosaicking.

- Finally, the framework depicted in Figure 17.1c produces the output image $\mathbf{x}' = f_{\varphi\phi}^{\tau}(z)$ via *simultaneous demosaicking and image resizing*. To the date, only very few solutions are known to demosaick and resample images in a more or less joint manner although both computational efficiency and performance resulting from performing these two processing step simultaneously make this framework possibly the best solution among the three pipelines.

17.3 Demosaicked Image Resizing

17.3.1 One-Dimensional Transforms

One-dimensional (1D) image resizing techniques [2] process the input image in two passes; for instance (Figure 17.2), first in the horizontal and then in the vertical direction. Operating on rows or columns of the visual input, the techniques map a finite 1D set of input pixel values into a finite 1D set of output pixel values. The mapping is determined by a resizing factor of the output line in relation to the input line and by a position factor in relation to the output line. 1D image resizing framework allows for image upsampling and downsampling in real time, and offers an effective and efficient match with many implementation technologies.

The image row or column in each color channel of the demosaicked image $\mathbf{x}$ can be seen as 1D set $I = \{I_1, I_2, ..., I_N\}$. Resizing the image with a factor τ, which is an arbitrary positive real number in the 1D resampling framework, produces the output set $O = \{O_1, O_2, ..., O_M\}$. Since the number of samples in the resized output set has to be an integer value, $M = \lfloor \tau N \rfloor$ where $\lfloor \cdot \rfloor$ denotes the floor rounding. Thus, $I = \{x_{(r,1)k}, x_{(r,2)k}, ..., x_{(r,K_2)k}\}$ and $O = \{x'_{(r,1)k}, x'_{(r,2)k}, ..., x'_{(r,\lfloor \tau K_2 \rfloor)k}\}$ for operating in the rth row and kth channel of $\mathbf{x}$, or $I = \{x_{(1,s)k}, x_{(2,s)k}, ..., x_{(K_1,s)k}\}$ and $O = \{x'_{(1,s)k}, x'_{(2,s)k}, ..., x'_{(\lfloor \tau K_1 \rfloor,s)k}\}$ for interpolating the sth

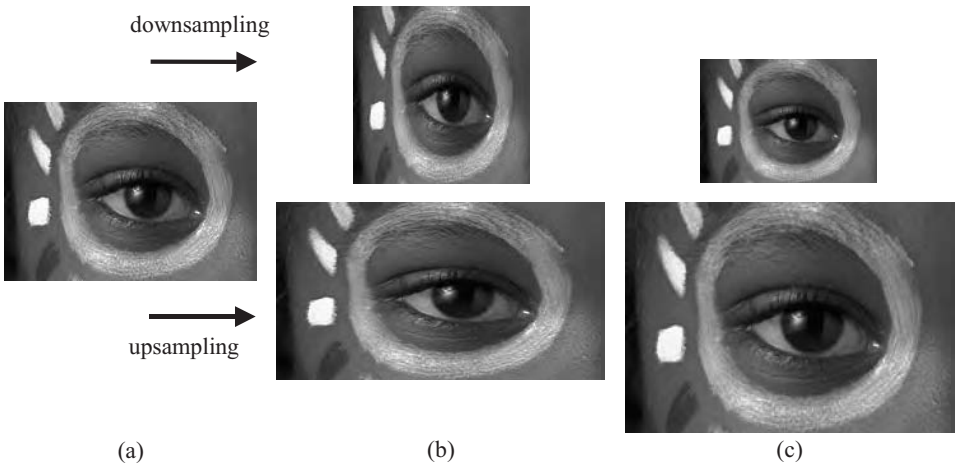


FIGURE 17.2

Demosaicked image resizing using 1D transforms: (a) input image, (b) image resized in the horizontal direction, (c) image resized in both horizontal and vertical directions. Factors of 0.7 and 1.3 were used, respectively, to downsample and upsample the input color image.

column and kth channel of $\mathbf{x}$. By knowing the dimension of the output set, the sample O_l, for $l = 1, 2, ..., M$, is obtained as follows:

$$O_l = \tau \sum_{j=\lfloor l\tau^{-1} \rfloor}^{\lfloor (l+1)\tau^{-1} \rfloor} \lambda_j \tag{17.1}$$

where j is the integer and λ_j denotes the input samples' contributions scaled using the position factor.

In the case of downsampling, the value of λ_j is given by

$$\lambda_j = \begin{cases} I_j(j - l\tau^{-1} + 1) & \text{if } l\tau^{-1} > j \\ I_j & \text{if } l\tau^{-1} \leq j \leq (l+1)\tau^{-1}, (l+1)\tau^{-1} - j \geq 1 \\ I_j((l+1)\tau^{-1} - j) & \text{if } l\tau^{-1} \leq j, (l+1)\tau^{-1} - j < 1 \end{cases} \tag{17.2}$$

whereas performing upsampling implies that λ_j is obtained as follows:

$$\lambda_j = \begin{cases} I_j(j - l\tau^{-1} + 1) & \text{if } l\tau^{-1} > j, \tau^{-1} \geq j - l\tau^{-1} + 1 \\ I_j\tau^{-1} & \text{if } l\tau^{-1} < j, \tau^{-1} \leq j - l\tau^{-1} + 1 \\ I_j((l+1)\tau^{-1} - j) & \text{if } l\tau^{-1} \leq j, (l+1)\tau^{-1} - j < 1 \end{cases} \tag{17.3}$$

where $I_j = I_N$ for $j \geq N$ to ensure that the pixels occupying the boundary locations in the image will have the desired intensity.

17.3.2 Two-Dimensional Standard Transforms

Unlike 1D transforms, two-dimensional (2D) transforms for image resizing are commonly used in various image viewing and editing software. Moreover, computationally efficient 2D resampling solutions are of frequent choice in embedded applications and implemented for the direct use in a digital camera.

17.3.2.1 Pixel Omission

Fast downsampling algorithms are required for quick inspection of captured camera images on camera displays. The pixel omission solution transfers the pixels selected in the input image into new spatial locations in the output image based on a downsampling factor. Operating on the color image $\mathbf{x}$, the *pixel omission* downsampling procedure can be described as follows:

$$\mathbf{x}'_{(m,n)} = \mathbf{x}_{((m-1)/\tau+1,(n-1)/\tau+1)} \tag{17.4}$$

where $m = 1,2,...,\tau K_1$ and $n = 1,2,...,\tau K_2$ denote the spatial location in the resized image $\mathbf{x}'$, and $0 < \tau \leq 1$ is the downsampling factor.

Depending on the value of τ, the quality of the downsampled image can vary significantly. Larger τ values (i.e., $0.2 \leq \tau \leq 1$) usually allow reasonable quality when displaying a Megapixel image on the camera display; however, the quality may not be sufficient for inspecting the images on monitors or when printed due to the grainy effects usually present in the areas with edges and fine details.

17.3.2.2 Pixel Replication

Pixel replication constitutes the simplest and fastest image upsampling solution. It transfers an input pixel occupying the (r,s) location under consideration into a $\tau \times \tau$ block of pixels with the value identical to that of the input pixel as follows:

$$\mathbf{x}'_{(\tau(r-1)+a,\tau(s-1)+b)} = \mathbf{x}_{(r,s)} \tag{17.5}$$

where (r,s) denotes the pixel location in the input image whereas $(\tau(r-1)+a, \tau(s-1)+b)$ denotes the location in the output, upsampled image. The output location is determined using the input location (r,s), upsampling factor $\tau \geq 1$, and parameters $a \in \{1,2,...,\tau\}$ and $b \in \{1,2,...,\tau\}$ which denote the pixel position within the block.

Due to the nature of the operation in Equation 17.5, the pixel replication solution preserves the details present in the input image. On the other hand, replicating the input pixels produces upsampled images with block effects. Note that the pixel replication solution performs the inverse steps to the pixel omission solution. Therefore, upsampling the image $\mathbf{x}$ using the pixel replication process with $\tau \geq 1$ followed by the pixel omission process with $1/\tau$ is equivalent to the identity operation $f(\mathbf{x}) = \mathbf{x}$.

17.3.2.3 Nearest Neighbor Interpolation

Similar to the pixel replication solution, the *nearest neighbor interpolation* algorithm [3] uses only one input pixel to determine the output pixel and does not consider the values of neighboring pixels. The nearest neighbor solution can be used to upsample or downsample the image.

The interpolated pixel is equal to the input pixel $\mathbf{x}_{(r,s)}$ which minimizes the spatial distance between the location (r,s) and the location $(m/\tau, n/\tau)$ which is a normalized variant of the interpolation location (m,n). Thus, the nearest neighbor interpolation process can be defined as follows:

$$\mathbf{x}'_{(m,n)} = \arg\min_{\mathbf{x}_{(r,s)}} d\left([m/\tau, n/\tau], [r,s]\right) \tag{17.6}$$

where $m = 1,2,...,\tau K_1$ and $n = 1,2,...,\tau K_2$ denote the location in the resized image whereas $r = 1,2,...,K_1$ and $s = 1,2,...,K_2$ denote the location in the input image. The term $d(\cdot,\cdot)$ denotes the distance, such as the absolute and Euclidean distance, between its two vector arguments.

The operations in Equations 17.5 and 17.6 are fast and easy to implement. In addition, they do not change the color information of the image and do not blur the image content (Figure 17.3a and Figure 17.4a). Therefore, image viewing and editing software often use the pixel replication or nearest neighbor interpolation solutions to enlarge a digital image for the purpose of closer examination. However, since these solutions increase the visibility of various visual impairments such as noise, jaggedness, and demosaicking artifacts, they are not suitable for upsampling the images in high-end applications.

17.3.2.4 Median Interpolation

This interpolation solution also selects the pixels from the input image to populate the output, resized image. However, unlike the nearest neighbor approach, the minimization criterion in *median interpolation* [4] is defined on the pixel values instead of the spatial locations:

$$x'_{(m,n)k} = \arg\min_{x_{(i,j)k}} \sum_{(p,q)\in\zeta} \left| x_{(i,j)k} - x_{(p,q)k} \right| \tag{17.7}$$

where k denotes the color channel and (i,j) denotes the pixel location from the area of support ζ centered around $(m/\tau, n/\tau)$.

Median interpolation produces sharp-looking images (Figure 17.3b and Figure 17.4b). Since the output of the median interpolation process is the most representative pixel of the neighborhood denoted by ζ, median interpolation can resize natural images with less processing error than the 2D transforms presented above. However, due to both the selective nature and the componentwise processing mode, the performance of median interpolation is rather insufficient, as the resized color images often suffer from the noticeable block effects, grainy appearance, and/or color artifacts present in the areas with significant structural content [5].

17.3.2.5 Bilinear / Bicubic Interpolation

To avoid the block effects in upsampled images or grainy appearance of downsampled images, the representative of the samples surrounding the interpolation location should be determined using the averaging operation. Bilinear and bicubic interpolations are probably the most popular resizing solutions based on the sample averaging concept [6], [7]. The pixels in the output image resized using these solutions are obtained as follows:

$$x'_{(m,n)k} = \sum_{(p,q)\in\zeta} w_{(p,q)} x_{(p,q)k} \tag{17.8}$$

where $w_{(p,q)}$ is a normalized weight associated with the pixel occupying the (p,q) location in the local neighborhood of input pixels determined by ζ. The term normalized means that $\sum_{(p,q)\in\zeta} w_{(p,q)} = 1$.

The weights $w_{(p,q)}$, for $(p,q) \in \zeta$, have fixed values determined as follows:

$$w_{(p,q)} = h(\Delta_p)h(\Delta_q) \tag{17.9}$$

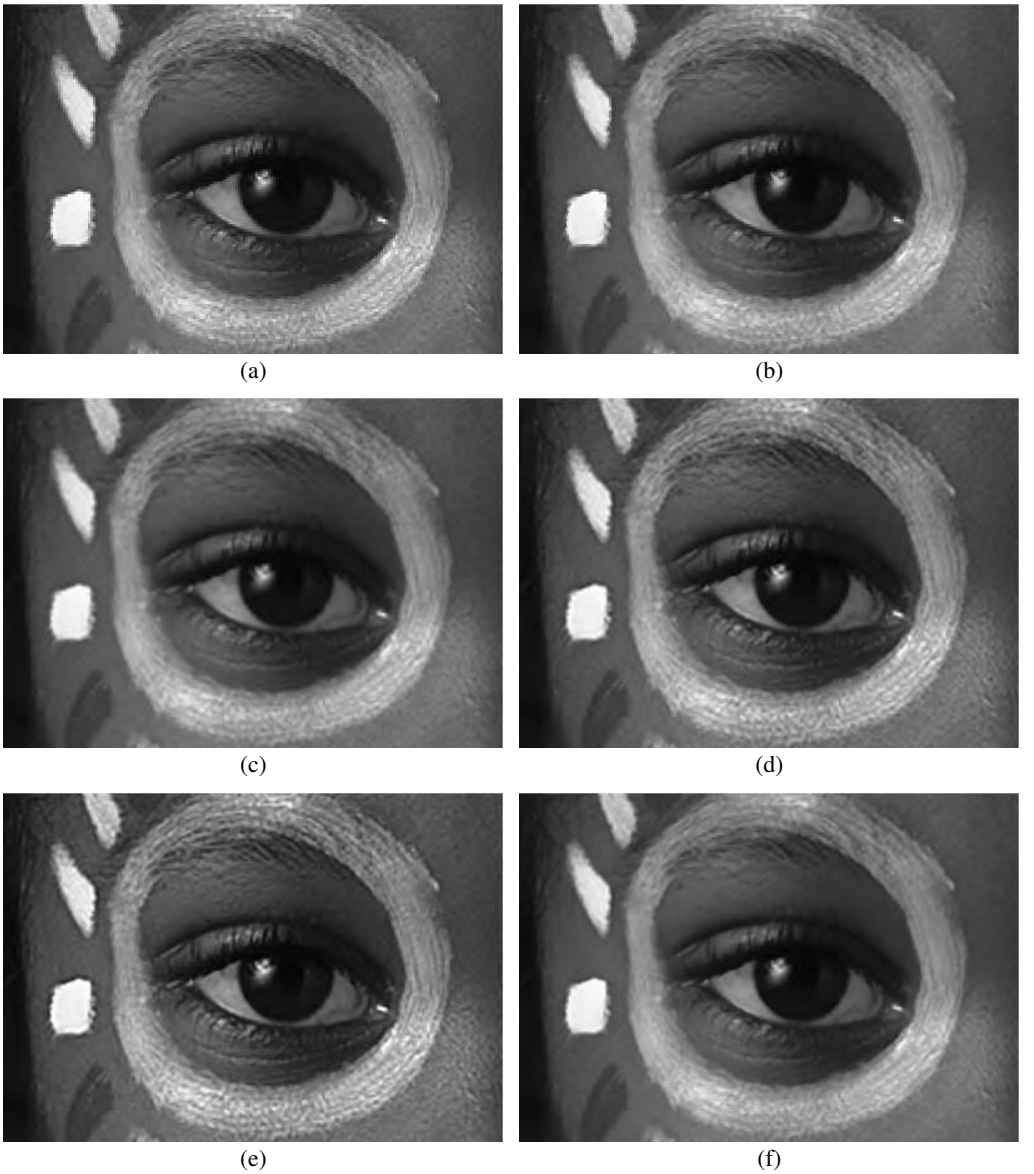


FIGURE 17.3 (See color insert.)

Camera image upsampling with $\tau = 2$. Cropped 280×400 areas from demosaicked images upsampled using (a) nearest neighbor, (b) median, (c) bilinear, (d) bicubic, (e) joint interpolation / edge enhancement, and (f) edge-adaptive solutions.

where $\Delta_p = p - m/\tau$ and $\Delta_q = q - n/\tau$ represent a pixel distance between the location (p,q) in the input image and the location under consideration in the output image expressed in the normalized coordinates, that is, $(m/\tau, n/\tau)$. To reduce the interpolation error and preserve the edge information, pixels occupying locations spatially closer to the interpolation location are given a higher weighting in the calculation.

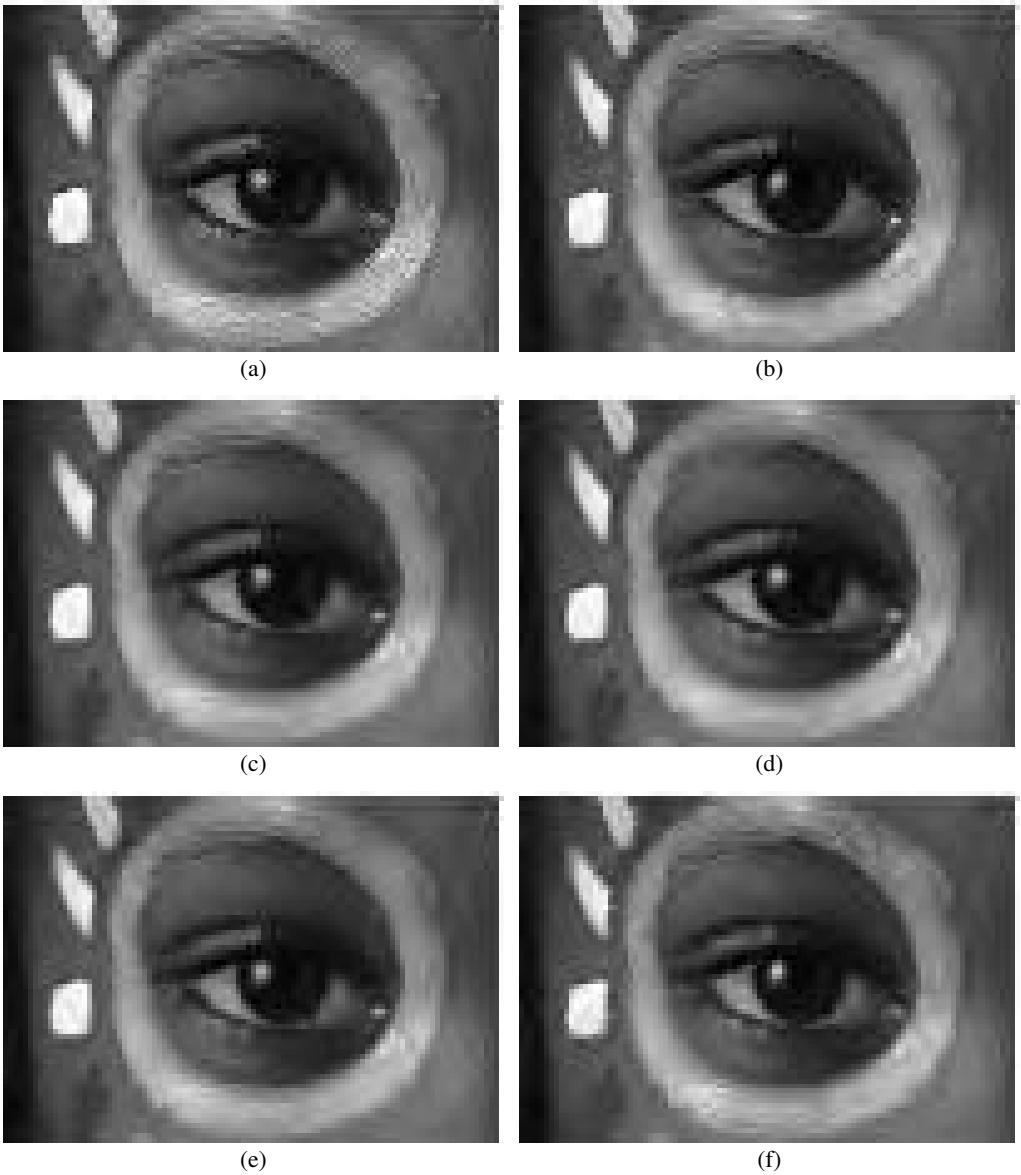


FIGURE 17.4 (See color insert.)

Camera image downsampling with $\tau = 1/2$. Cropped 70×100 areas from demosaicked images downsampled using (a) nearest neighbor, (b) median, (c) bilinear, (d) bicubic, (e) edge-adaptive, and (f) vector median solutions.

Bilinear interpolation considers the closest 2×2 neighborhood ζ of input pixel values surrounding the interpolation location $(m/\tau, n/\tau)$. It uses the weights from Equation 17.9 obtained through the function $h(\cdot)$ defined as follows:

$$h(\Delta) = \begin{cases} 1 - |\Delta| & \text{if } 0 \leq |\Delta| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{17.10}$$

Since the output, interpolated value is equal to a weighted average of four input pixels,

bilinear interpolation results in much smoother looking images (Figure 17.3c and Figure 17.4c) than those obtained by methods discussed in Sections 17.3.2.1 to 17.3.2.4.

Bicubic interpolation goes one step beyond bilinear by considering sixteen pixels inside the closest 4×4 neighborhood of known pixels. The weights in bicubic interpolation are determined using Equation 17.9 with $h(\cdot)$ expressed as:

$$h(\Delta) = \begin{cases} 1 - 2|\Delta|^2 + |\Delta|^3 & \text{if } |\Delta| < 1 \\ 4 - 8|\Delta| + 5|\Delta|^2 - |\Delta|^3 & \text{if } 1 \leq |\Delta| < 2 \\ 0 & \text{otherwise} \end{cases} \tag{17.11}$$

Bicubic interpolation produces noticeably sharper images (Figure 17.3d and Figure 17.4d) than bilinear interpolation. It is considered by many as the ideal combination of processing time and output quality. For this reason it is widely used in image editing software, printer drivers, and in-camera image resizing.

Higher-order interpolation algorithms such as those based on the spline and sinc functions use even more surrounding pixels than bicubic interpolation, and thus they are more computationally intensive. These algorithms are useful when the image requires multiple resizing and rotation in separate steps, whereas for single-step interpolation, they do not provide significant visual improvements.

17.3.3 Fast Kernel-Based Solutions for Image Upsampling

Unlike various solutions which produce an enlarged image in several passes or processing steps in order to overcome the lack of available samples during interpolation, the framework introduced in Reference [8] upsamples digital images in a single processing step. The framework is extremely computationally efficient and can perform spatial interpolation or even joint spatial interpolation / image enhancement in the real time by mapping each input pixel to a $\tau \times \tau$ block of pixels in the output image. Each of the output values is obtained as a combination of the input pixels inside the local neighborhood determined by the supporting window centered in the actual location to be mapped.

Operating on the demosaicked image $\mathbf{x}$, blocks of output color components are produced, in a componentwise processing manner, as follows:

$$x'_{(\tau(r-1)+a,\tau(s-1)+b)k} = \sum_{(p,q)\in\zeta} w^{a,b}_{(p,q)} x_{(p,q)k} \tag{17.12}$$

where r and s denote the pixel location in the input image whereas parameters $a \in \{1,2,...,\tau\}$ and $b \in \{1,2,...,\tau\}$ denote the location indices in the blocks of $\tau \times \tau$ output, interpolated values. The term ζ denotes the pixel locations in the input image which are centered in (r,s). Visually pleasing results are produced using supporting windows with ζ larger than the block size. The actual size of ζ depends on the enlargement factor τ and the overall edge smoothing and enhancement characteristics of the spatial interpolation operation to be performed. The upsampled image is produced by repeating Equation 17.12 in all pixel locations, that is, for $r = 1,2,...,K_1$ and $s = 1,2,...,K_2$.

To avoid various visual impairments in the enlarged image, Equation 17.12 is an *unbiased estimator*, thus implying that the weights $w^{a,b}_{(p,q)}$ associated with the input color components $x_{(p,q)}$, for $(p,q) \in \zeta$, are normalized and satisfy $\sum_{(p,q)\in\zeta} w^{a,b}_{(p,q)} = 1$. The contribution of the

input samples to the interpolated output depends on both the input pixel location $(r,s) \in \zeta$ and the interpolation location denoted by the upsampling factor τ and block indices a and b. Therefore, Equation 17.12 uses τ^2 different weight vectors $\mathbf{w}^{a,b} = \{w^{a,b}_{(p,q)}; (p,q) \in \zeta\}$. Each of the weight vectors assigns the largest weight to the window center (r,s) in order to emphasize the central pixel due to its highest relevance to the interpolated outputs, thus ensuring the fidelity of the interpolation process. In addition, nonzero weights are assigned to the neighboring samples in the relevant interpolation direction to ensure smooth edges in the interpolated image. Finally, it should be noted that in order to produce visually pleasing images, weight vectors satisfy the following symmetry constraints: $\mathbf{w}^{a,b} \leftrightarrow \mathbf{w}^{a,\tau-b+1}$, $\mathbf{w}^{a,b} \leftrightarrow \mathbf{w}^{\tau-a+1,b}$, and $\mathbf{w}^{a,b} \leftrightarrow \mathbf{w}^{\tau-a+1,\tau-b+1}$ where $\leftrightarrow$ denotes the symmetry relation.

17.3.3.1 Spatial Interpolation

Since the window has to be centered in (r,s) and its size should extend beyond the size of $\tau \times \tau$ blocks, even values of τ suggest that spatial interpolation is performed using $(\tau+1) \times (\tau+1)$ windows described by $\zeta = \{(r+i,s+j); -\tau/2 \le i \le \tau/2, -\tau/2 \le j \le \tau/2\}$. If odd values of τ are requested, then $(\tau+2) \times (\tau+2)$ windows with $\zeta = \{(r+i,s+j); -\tau/2 - 0.5 \le i \le \tau/2 + 0.5, -\tau/2 - 0.5 \le j \le \tau/2 + 0.5\}$ should be used.

Enlarging the image using Equation 17.12 with the standard upsampling factor $\tau = 2$ implies 3×3 windows described by $\zeta = \{(r+i,s+j); -1 \le i \le 1, -1 \le j \le 1\}$. Since the spatial interpolation process is similar to *low-pass filtering*, the weight vectors $\mathbf{w}^{a,b}$ can follow the characteristics of a *Gaussian* low-pass filter with $\sigma = 1$. To simplify the implementation of the spatial interpolation solution with such characteristics, the weights can be approximated using the following integer coefficients:

$$\begin{aligned} \mathbf{w}^{1,1} &= \tfrac{1}{10}\begin{bmatrix} 1 & 2 & 0 \\ 2 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{w}^{1,2} = \tfrac{1}{10}\begin{bmatrix} 0 & 2 & 1 \\ 0 & 5 & 2 \\ 0 & 0 & 0 \end{bmatrix} \\ \mathbf{w}^{2,1} &= \tfrac{1}{10}\begin{bmatrix} 0 & 0 & 0 \\ 2 & 5 & 0 \\ 1 & 2 & 0 \end{bmatrix}, \quad \mathbf{w}^{2,2} = \tfrac{1}{10}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 2 \\ 0 & 2 & 1 \end{bmatrix} \end{aligned} \tag{17.13}$$

which constitute the weight vectors that satisfy the symmetry constraints.

17.3.3.2 Joint Spatial Interpolation and Edge Enhancement

In addition to the spatial interpolation operations, the framework presented in Reference [8] can enhance image structural content such as edges and fine details. In order to avoid blur due to low-pass filtering characteristics, the weights in Equation 17.12 should allow for *simultaneous* low-pass and high-pass filtering. Performing spatial interpolation with such hybrid (low-pass and high-pass) characteristics requires using a larger area of support, namely $(\tau+3) \times (\tau+3)$ windows described by $\zeta = \{(r+i,s+j); -\tau/2 - 1 \le i \le \tau/2 + 1, -\tau/2 - 1 \le j \le \tau/2 + 1\}$ for even values of τ, and $(\tau+4) \times (\tau+4)$ windows described by $\zeta = \{(r+i,s+j); -\tau/2 - 1.5 \le i \le \tau/2 + 1.5, -\tau/2 - 1.5 \le j \le \tau/2 + 1.5\}$ for odd values of τ.

For instance, $\tau = 2$ implies 5×5 windows with $\zeta = \{(r+i,s+j); -2 \le i \le 2, -2 \le j \le 2\}$. Since *Laplacian filtering* is one of the most efficient high-pass filters which operate using integer coefficients, the Laplacian filter coefficients can be combined with Gaussian-

like weights in Equation 17.13 to obtain a solution which is simple and easy to implement as follows:

$$\mathbf{w}^{1,1} = \frac{1}{80}\begin{bmatrix} -1 & -3 & -3 & -2 & 0 \\ -3 & 7 & 24 & -7 & 0 \\ -3 & 24 & 75 & -7 & 0 \\ -2 & -7 & -7 & -5 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{w}^{1,2} = \frac{1}{80}\begin{bmatrix} 0 & -2 & -3 & -3 & -1 \\ 0 & -7 & 24 & 7 & -3 \\ 0 & -7 & 75 & 24 & -3 \\ 0 & -5 & -7 & -7 & -2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{w}^{2,1} = \frac{1}{80}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -2 & -7 & -7 & -5 & 0 \\ -3 & 24 & 75 & -7 & 0 \\ -3 & 7 & 24 & -7 & 0 \\ -1 & -3 & -3 & -2 & 0 \end{bmatrix}, \quad \mathbf{w}^{2,2} = \frac{1}{80}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -5 & -7 & -7 & -2 \\ 0 & -7 & 75 & 24 & -3 \\ 0 & -7 & 24 & 7 & -3 \\ 0 & -2 & -3 & -3 & -1 \end{bmatrix} \tag{17.14}$$

The presence of the negative weights in Equation 17.14 suggests the edge enhancement ability of the considered framework (Figure 17.3e). Similar to the conventional Laplacian high-pass filter, the summation of the weights in each weight vector listed in Equation 17.14 is unity. Moreover, as typical for the presented framework, the weight vectors in Equation 17.14 emphasize the central pixel and the relevant interpolation direction, and satisfy the symmetry constraint.

17.3.4 Edge-Adaptive Resizing

Another way of avoiding blurred edges and aliasing artifacts in the resized images is to analyze the local structure of the input image prior to resizing it. Unlike solutions presented in Sections 17.3.1 to 17.3.3 which apply the resizing function indiscriminately to the whole image, *edge-adaptive* frameworks adjust the resizing function according to the presence and orientation of image structures in order to interpolate along edges, thus preserving structural content. Examples of edge-adaptive resizing include solutions controlled by edge-sensing parameters [9], [10], solutions which use different resizing functions with different areas of support [11], [12], [13], and solutions which operate under certain optimality criterion [14], [15], [16], [17]. This section focuses on the first category which constitutes an approach often of interest due to its good performance (Figure 17.3f and Figure 17.4e) and reasonable computational efficiency.

Popular edge-sensing algorithms process each color channel of the demosaicked image separately using Equation 17.8 with the normalized weights $w_{(p,q)}$ determined via

$$w_{(p,q)} = w'_{(p,q)} / \sum_{(p,q)\in\zeta} w'_{(p,q)} \tag{17.15}$$

where $w'_{(p,q)}$ is an edge-sensing weight commonly calculated based on the *inverse gradient* concept. Large image gradients usually indicate that the corresponding input $x_{(p,q)k}$ in Equation 17.8 is located across an edge and therefore $x_{(p,q)k}$ should not contribute much to the output in order to reduce the processing error. This constraint is realized by calculating the weights inversely proportional to gradient values.

For example, a simple way of calculating the weights $w'_{(p,q)}$, for $(p,q) \in \zeta = \{(r-1,s-1),(r-1,s+1),(r+1,s-1),(r+1,s+1)\}$ associated with the four samples surrounding

the location under consideration (r,s) in vertical and horizontal directions, is as follows:

$$\begin{aligned} w'_{(r-1,s)} &= 1/(1+|x_{(r-1,s)k}-x_{(r+1,s)k}|+|x_{(r-2,s)k}-x_{(r,s)k}|) \\ w'_{(r,s-1)} &= 1/(1+|x_{(r,s-1)k}-x_{(r,s+1)k}|+|x_{(r,s-2)k}-x_{(r,s)k}|) \\ w'_{(r,s+1)} &= 1/(1+|x_{(r,s+1)k}-x_{(r,s-1)k}|+|x_{(r,s+2)k}-x_{(r,s)k}|) \\ w'_{(r+1,s)} &= 1/(1+|x_{(r+1,s)k}-x_{(r-1,s)k}|+|x_{(r+2,s)k}-x_{(r,s)k}|) \end{aligned} \tag{17.16}$$

whereas for four samples in diagonal directions with $\zeta = \{(r-1,s-1),(r-1,s+1),(r+1,s-1),(r+1,s+1)\}$ the weights can be obtained as:

$$\begin{aligned} w'_{(r-1,s-1)} &= 1/(1+|x_{(r-1,s-1)k}-x_{(r+1,s+1)k}|+|x_{(r-2,s-2)k}-x_{(r,s)k}|) \\ w'_{(r-1,s+1)} &= 1/(1+|x_{(r-1,s+1)k}-x_{(r+1,s-1)k}|+|x_{(r-2,s+2)k}-x_{(r,s)k}|) \\ w'_{(r+1,s-1)} &= 1/(1+|x_{(r+1,s-1)k}-x_{(r-1,s+1)k}|+|x_{(r+1,s-1)k}-x_{(r,s)k}|) \\ w'_{(r+1,s+1)} &= 1/(1+|x_{(r+1,s+1)k}-x_{(r-1,s-1)k}|+|x_{(r+1,s+1)k}-x_{(r,s)k}|) \end{aligned} \tag{17.17}$$

where the absolute value terms denote gradients calculated in specific directions. More sophisticated edge-sensing solutions can be found in References [9], [10], and [17].

In Equations 17.16 and 17.17, extreme observations in the interpolator's input still contribute to the output. Contributions from samples defined across an edge can be eliminated by thresholding the corresponding weighting coefficient as follows:

$$w'_{(p,q)} = \begin{cases} w'_{(p,q)} & \text{if } w'_{(p,q)} \geq \xi \\ 0 & \text{otherwise} \end{cases} \tag{17.18}$$

where ξ is a threshold value which provides the end-user with the means to regulate edge sensitivity. In the most extreme way, resizing operations can be limited to the direction with the most dominant weight:

$$w'_{(p,q)} = \begin{cases} 1 \text{ if } w'_{(p,q)} = \max\{w'_{(i,j)};\ (i,j) \in \zeta\} \\ 0 \text{ otherwise} \end{cases} \tag{17.19}$$

thus preventing smoothing effects in the resized image. Since Equation 17.19 selects one input sample as the output, the accuracy of the resizing process critically depends on the weighting function formulation.

17.3.5 Vector Processing Solutions

Besides the pixel replication / omission and nearest neighbor solutions, the resampling techniques presented in Sections 17.3.1 to 17.3.4 resize each color plane of the demosaicked image separately (Figure 17.5a). Since a typical natural image exhibits significant spectral correlation, *componentwise processing* solutions can introduce various color shifts and spectral artifacts into the resized demosaicked images [5], [18]. To avoid the problem, *vector processing* operators rely on the trichromatic theory of color and process the pixels in the demosaicked image as a set of vectors (Figure 17.5b).

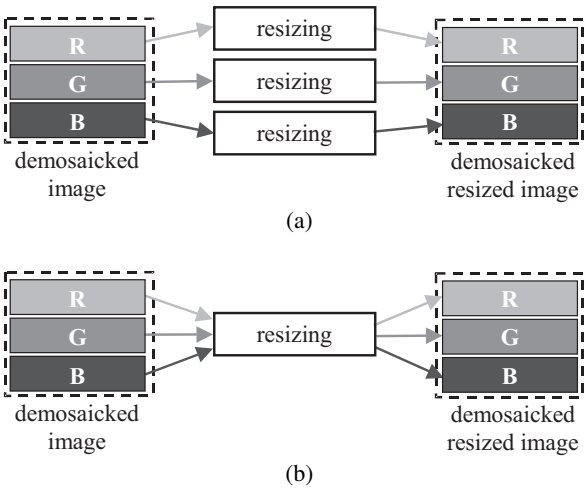


FIGURE 17.5

Typical color image resizing frameworks: (a) componentwise processing, (b) vector processing.

17.3.5.1 Pixel-Selective Framework

The solutions designed within the *pixel-selective* vector processing framework produce the output color pixel $\mathbf{x}'_{(r,s)}$ as the vector $\mathbf{x}_{(i,j)}$, for $(i,j) \in \zeta$, minimizing the *aggregated distance* or *similarity* criterion to other color vectors inside the shape mask ζ:

$$\mathbf{x}'_{(r,s)} = \arg\min_{\mathbf{x}_{(i,j)}} \sum_{(p,q)\in\zeta} D\left(\mathbf{x}_{(i,j)}, \mathbf{x}_{(p,q)}\right) \tag{17.20}$$

where $D\left(\mathbf{x}_{(i,j)}, \mathbf{x}_{(p,q)}\right)$ denotes the distance between two vectors $\mathbf{x}_{(i,j)}$ and $\mathbf{x}_{(p,q)}$. Since each vector is uniquely defined by its length (magnitude) and orientation (direction) in the vector space [19], Equation 17.20 can be realized using the magnitude distances, the directional distances, or their combination.

In the case of the magnitude processing, $D(\cdot,\cdot)$ is usually defined using the Minkowski metric:

$$D\left(\mathbf{x}_{(i,j)}, \mathbf{x}_{(p,q)}\right) = \left(\sum_{k=1}^{3} \left|x_{(i,j)k} - x_{(g,h)k}\right|^L\right)^{\frac{1}{L}} \tag{17.21}$$

where L denotes the norm parameter; for example, $L = 1$ for the city-block distance and $L = 2$ for the Euclidean distance. Equation 17.20 with the Euclidean distance constitutes the *vector median operator* (Figure 17.4f) [20]. Since magnitude relates to the luminance component of color images, vector processing solutions operating in the magnitude domain preserve the luminance characteristics of $\mathbf{x}$.

In the case of directional processing, the most common form of $D(\cdot,\cdot)$ is the angular distance:

$$\mathrm{D}(\mathbf{x}_{(i,j)}, \mathbf{x}_{(p,q)}) = \arccos\left(\frac{\mathbf{x}_{(i,j)} \cdot \mathbf{x}_{(p,q)}}{|\mathbf{x}_{(p,q)}||\mathbf{x}_{(p,q)}|}\right) \tag{17.22}$$

which makes Equation 17.20 equivalent to the solution which is well-known as the *vector directional operator* [21]. Such a solution outputs the color vector that is the most simi-

lar, in terms of the directional characteristics of $\mathbf{x}$, to other color vectors in the localized neighborhood. Note that the vector's direction denotes the chrominance characteristics of the color pixel. Therefore, vector processing solutions operating in the directional domain preserve color chromaticity.

Both the luminance and color chromaticity are essential for human perception. Therefore, some vector solutions such as those presented in Reference [22] use both magnitude and directional information during processing. Note that Equation 17.20 can operate using various distance and similarity measures. The interested reader can find the suitable solutions in Reference [19].

17.3.5.2 Data-Adaptive Framework

Due to the selective nature and the use of the minimization concept, Equation 17.20 never introduces new samples. Since the output of the vector operators discussed above is the color pixel from the localized neighborhood determined by ζ, Equation 17.20 often results in the uniform neighborhoods, thus producing images with the block effects. To avoid these effects, the contribution of the adjacent color vectors $\mathbf{x}_{(p,q)}$, for $(p,q) \in \zeta$, to the output $\mathbf{y}_{(r,s)}$ should be proportionally represented.

In the vector *data-adaptive* framework [19], the output color vector is obtained as follows:

$$\mathbf{x}'_{(r,s)} = f\left(\sum_{(p,q)\in\zeta} w_{(p,q)}\mathbf{x}_{(p,q)}\right) \tag{17.23}$$

where $f(\cdot)$ is a function that operates over the weighted average of the input color vectors. The term $w_{(p,q)}$ associated with the input color vector $\mathbf{x}_{(p,q)}$ denotes the weight normalized according to Equation 17.15.

Similar to Equation 17.20, the data-adaptive framework can use various $D(\cdot,\cdot)$ measures to obtain color output. In addition, the framework can operate using various $f(\cdot)$ functions. For example, weights $w'_{(p,q)}$ from Equation 17.15 can be achieved in each pixel location using functions of $D(\cdot,\cdot)$ in order to determine $w_{(p,q)}$ from Equation 17.23. Since the relationship between distances measured in physical units and perception is generally exponential [23], the *sigmoidal membership function* can be used to define $w'_{(p,q)}$, for $(p,q) \in \zeta$, as follows:

$$w'_{(p,q)} = \beta\left(1 + \sum_{(p,q)\in\zeta} D\left(\mathbf{x}_{(p,q)}, \mathbf{x}_{(i,i)}\right)\right)^{-\gamma} \tag{17.24}$$

where β is a normalizing constant and γ is a parameter adjusting the weighting effect of the membership function. Thus, replacing the membership function or the actual form of $f(\cdot)$ in Equation 17.23 changes not only the design philosophy, but the performance characteristics and computational complexity of the processing solution.

17.3.6 Image Resizing in the Compressed Domain

In addition to previously discussed solutions which perform resizing operations in the spatial domain, the dimensions of a digital image can also be changed by operating in the *compressed domain*. The approach is suitable for imaging pipelines which store images in the compressed format, as it reduces the computational overhead associated with coding and decoding of the compressed stream.

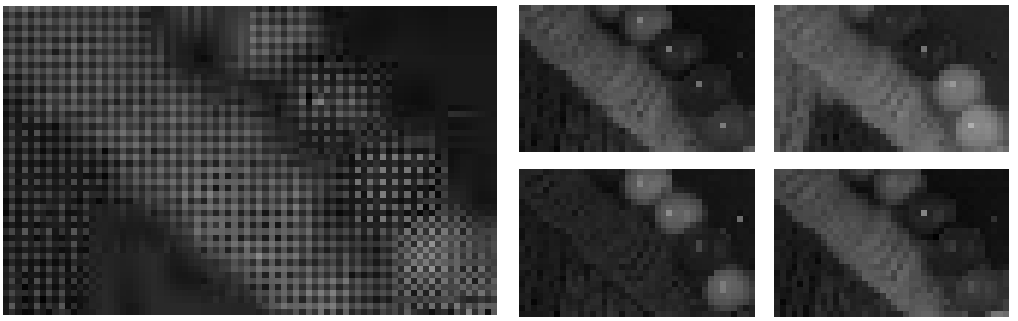
Most demosaicked images are stored using the *Joint Photographic Experts Group* (JPEG) coding scheme which employs the *discrete cosine transform* (DCT) to represent the visual data in the *YUV* color space [24]. Since *YUV* constitutes a decorrelated color space, each color channel of the YUV image can be resized separately although the different sampling frequency of the luminance (Y) and chrominance (U, V) components, usually 4:1:1 in standard JPEG, makes the design of resizing algorithms challenging. Existing DCT-based resizing solutions can basically be divided into three frameworks [25]. Namely, one framework exploits linear, distributive and unitary transform properties to accomplish image resizing by manipulating matrix multiplications in the DCT domain. Another framework uses convolution-multiplicative properties of trigonometric transforms to perform resizing operations through low-pass filtering of DCT coefficients. Finally, through subband DCT computations the third framework exploits spatial relationships of the block DCTs to alter the image dimensions. Additional information on the above frameworks and a comprehensive survey of DCT resizing techniques can be found in Reference [25].

To improve visual quality while obtaining higher compression ratios, enhanced color space flexibility and better metadata support, different variants of the *JPEG 2000* coding scheme [26], with reduced computational complexity and memory requirements, are under consideration for inclusion in the imaging pipeline [27]. Since JPEG 2000-like coding schemes use subband decompositions to obtain multiresolution image representation, images with different spatial resolutions can easily be obtained by combining different subbands or resizing subband images using standard spatial transforms.

17.4 Color Filter Array Image Resizing

17.4.1 Structure Conversion-Based Approaches

Although a typical natural color image exhibits significant correlation in both spatial and spectral senses, the neighboring pixels in the CFA image are acquired using different color filters and therefore they often significantly differ in their intensities. Thus, resizing mosaic data directly using techniques for grayscale and color imaging without considering the spectral sensitivities of the corresponding color filters in CFA greatly reduces the accuracy of interpolated values. To overcome this problem, the most natural way is to group the pixels corresponding to the same color filter in order to create *subimages* that exhibit much higher spatial correlation and therefore are more suitable for subsequent resizing [12], [28]. Note that this approach is widely used in *single-sensor image compression* to increase coding efficiency [29], [30]. A survey of existing *structure conversion* approaches can be found in Chapter 15.



(a) (b)

FIGURE 17.6

Structure conversion: (a) Bayer CFA image and (b) its corrresponding four subimages.

The simplest method of structure conversion for images captured by the Bayer CFA (Figure 1.3) is to pack the acquired sensor readings as follows:

$$\begin{array}{cc} z^1_{(r,s)} = z_{(2r-1,2s-1)}; & z^2_{(r,s)} = z_{(2r-1,2s)}; \\ z^3_{(r,s)} = z_{(2r,2s-1)}; & z^4_{(r,s)} = z_{(2r,2s)} \end{array} \tag{17.25}$$

where $r = 1,2,...,K_1/2$ and $s = 1,2,...,K_2/2$. Pixels $z^1_{(\cdot,\cdot)}$ and $z^4_{(\cdot,\cdot)}$ constitute the two subimages z^1 and z^4 containing G components, $z^2_{(\cdot,\cdot)}$ constitute a subimage z^2 created using R components, and $z^3_{(\cdot,\cdot)}$ constitute a subimage z^3 created using B components from the CFA image z. Thus, each of the four subimages consists of $K_1/2 \times K_2/2$ pixels.

Figure 17.6 demonstrates the difference between the CFA image and its subimages. The CFA image has the mosaic appearance and contains artificial high-frequencies between the neighboring pixels acquired using different color filters. On the other hand, the corresponding subimages contain more natural edges and transitions between the flat regions. Therefore, performing resizing operations on subimages rather than directly on the mosaic image can reduce the processing error.

The structure conversion approach can be used to support both upsampling and downsampling operations (Figure 17.7a and Figure 17.8a). Resizing subimages using the spatial correlation as $z'^i = f^\tau_\phi(z^i)$ or using both spatial and spectral correlations as $z'^i = f^\tau_\phi(z^1, z^2, z^3, z^4)$, for $i = 1,2,...,4$, produces four subimages with the resolution of $\tau K_1/2 \times \tau K_2/2$ pixels. The resized CFA image z' with the resolution of $\tau K_1 \times \tau K_2$ pixels is obtained from resized subimages z'^1, z'^2, z'^3, and z'^4 using the inverse structure conversion method as follows:

$$\begin{array}{cc} z'_{(2r-1,2s-1)} = z'^1_{(r,s)}; & z'_{(2r-1,2s)} = z'^2_{(r,s)}; \\ z'_{(2r,2s-1)} = z'^3_{(r,s)}; & z'_{(2r,2s)} = z'^4_{(r,s)} \end{array} \tag{17.26}$$

where $r = 1,2,...,\tau K_1/2$ and $s = 1,2,...,\tau K_2/2$ denote the spatial location in the resized subimages. Similar to standard color imaging, the resulting processing error depends on the ability of $f^\tau_\phi(\cdot)$ to follow both structural and spectral characteristic of z during the resizing process.

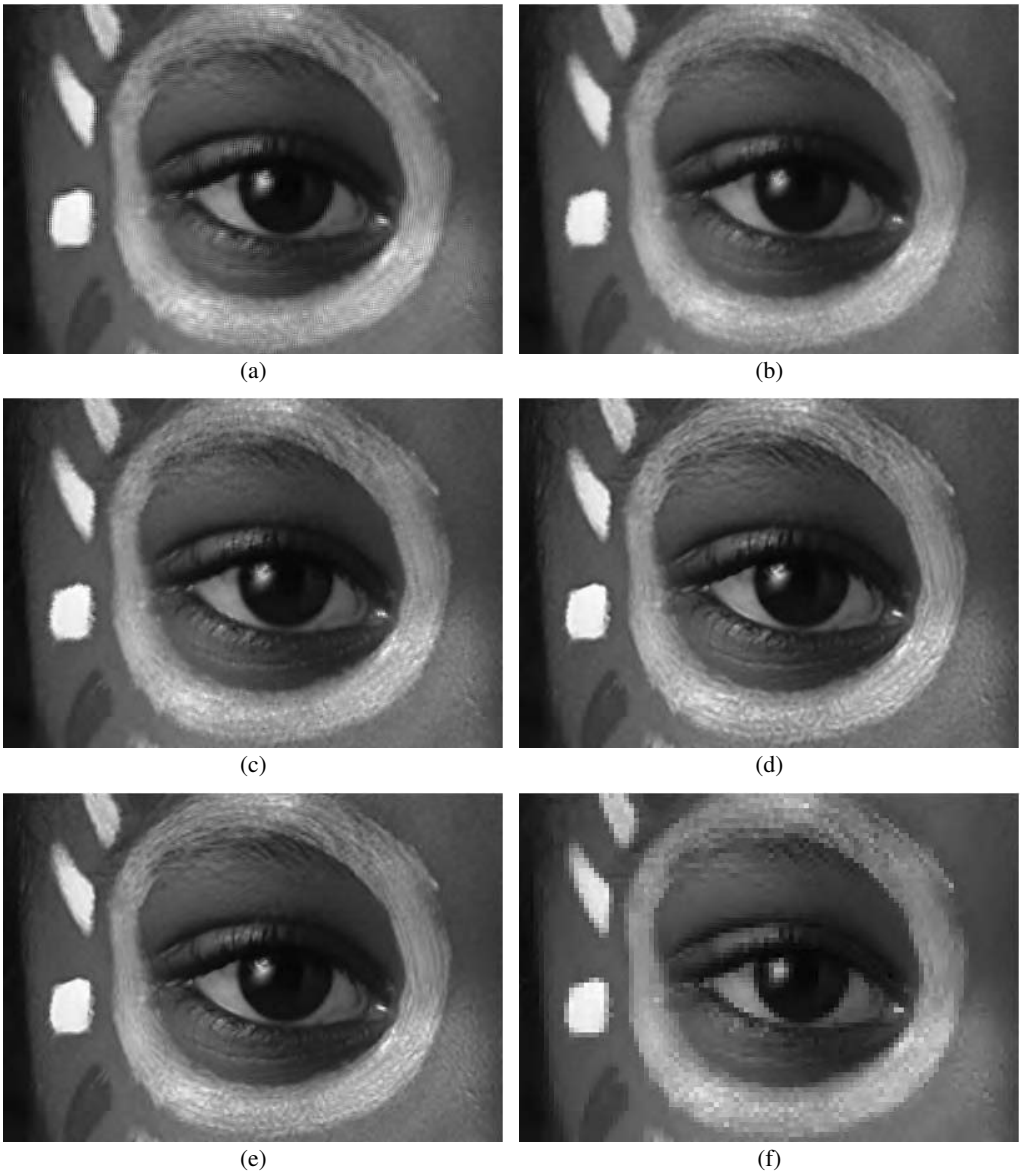
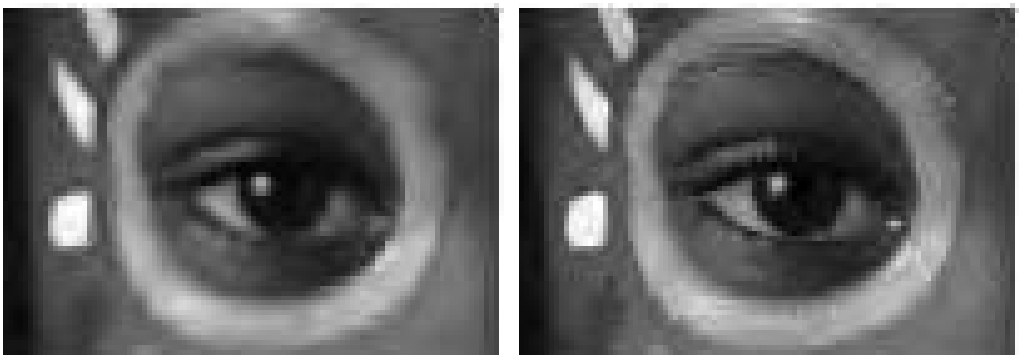


FIGURE 17.7 (See color insert.)

Camera image upsampling with $\tau = 2$. Cropped 280×400 areas from images obtained by (a) structure conversion-based CFA image resizing, (b) pixel mapping-based CFA image resizing using spatial correlations, (c) pixel mapping-based CFA image resizing using spatial and spectral correlations, (d) partially integrated demosaicking / resizing via mixing the upsampled demosaicked green channel and two upsampled color difference images, (e) partially integrated demosaicking / resizing by sharing edge orientation estimates, and (f) joint demosaicking / resizing.

17.4.2 Pixel Mapping-Based Approaches

Another approach to increasing the resolution of CFA images is based on the *mapping* of the acquired CFA sensor readings into the enlarged image array. Since, depending on the



(a) (b)

FIGURE 17.8 (See color insert.)

Camera image downsampling with $\tau = 1/2$. Cropped 70×100 areas from images obtained by (a) structure conversion-based CFA image resizing, and (f) joint demosaicking / resizing.

resizing factor τ, the mapping step may not fill all pixel locations in the enlarged image, an interpolation step is often needed to complete the resizing process.

Unfortunately, Figure 17.9 shows that the conventional pixel mapping $z'_{(\tau(r-1)+1,\tau(s-1)+1)} = z_{(r,s)}$ can be inappropriate for upsampling mosaic images with certain resizing factors as it destroys the underlying structure of the CFA and thus prohibits the subsequent utilization of standard demosaicking algorithms. For example, using the most common setting $\tau = 2$ the pixels from the original Bayer CFA image (Figure 17.9a) are mapped into positions reserved for G samples in the enlarged mosaic image (Figure 17.9b). Therefore, in order to zoom the mosaic data and at the same time preserve the basic structuring assumption behind the CFA, the original sensor data should be assigned unique positions. Following the approach from Reference [31], the mapping step for Bayer CFA images and upsampling factor $\tau = 2$ can be performed as follows (Figure 17.9c):

$$\left.\begin{array}{l} z'_{(2r-1,2s)} \\ z'_{(2r,2s-1)} \\ z'_{(2r-1,2s-1)} \end{array}\right\} = z_{(r,s)} \quad \begin{array}{l} \text{for } r \text{ odd and } s \text{ even} \\ \text{for } r \text{ even and } s \text{ odd} \\ \text{otherwise} \end{array} \tag{17.27}$$

where $r = 1,2,...,K_1$ and $s = 1,2,...,K_2$ denote coordinates in the original CFA image z.

After the mapping step has been completed, depending on the value of the resizing factor τ, missing pixels can be obtained in several iterations. The spatial distance between the interpolation location (r,s) and the location (i,j) of the available color component in the local neighborhood ζ reduces with the iteration index [31]. For example, following the pixel arrangement from Figure 17.9c, it is natural to first estimate G components located in the center of $\zeta = \{(r-2,s),(r,s-2),(r,s+2),(r+2,s)\}$ and R and B components located in the center of $\zeta = \{(r-2,s-2),(r-2,s+2),(r+1,s-2),(r+2,s+2)\}$. The process forms new arrangements and thus, in the second iteration, missing G components are located in the center of $\zeta = \{(r-1,s-1),(r-1,s+1),(r+1,s-1),(r+1,s+1)\}$ whereas R and B components are located in the center of $\zeta = \{(r-2,s),(r,s-2),(r,s+2),(r+2,s)\}$.

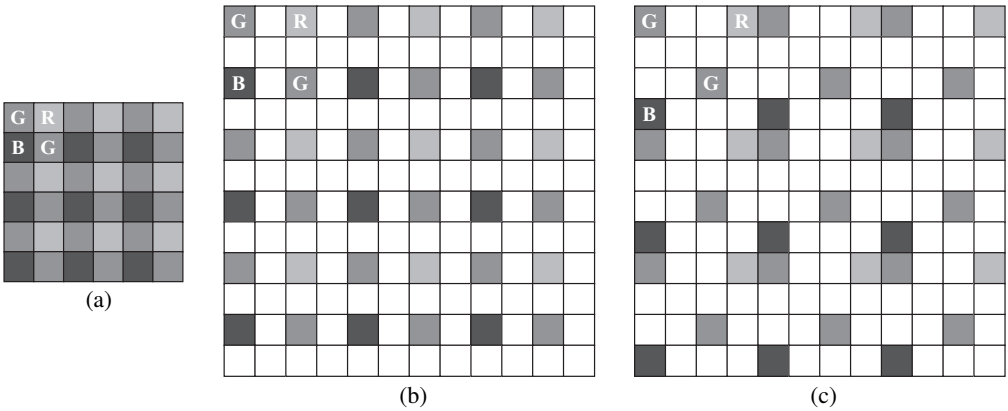


FIGURE 17.9

CFA pixel mapping solutions for image upsampling: (a) Bayer CFA, (b) conventional mapping, and (c) CFA-specific mapping.

17.4.2.1 Interpolation Using Spatial Correlation

The easiest way of interpolating pixels in the resized CFA image is to estimate them in a componentwise manner, that is, using the interchannel correlation only [32]:

$$z'_{(r,s)} = \sum_{(i,j)\in\zeta} w_{(i,j)} z_{(i,j)} \tag{17.28}$$

where $w_{(i,j)} = w'_{(i,j)} / \sum_{(p,q)\in\zeta} w'_{(p,q)}$ is a normalized weight.

In the case of simple linear interpolation, all weights $w'_{(i,j)}$ have the same value resulting in $w_{(i,j)} = 1/|\zeta|$. The structural content of the CFA image can be preserved by using $w'_{(i,j)}$ determined adaptively based on the image statistics. To reduce memory requirements, it was suggested in Reference [31] to calculate the weights as follows:

$$w'_{(i,j)} = \left(1 + \sum_{(p,q)\in\zeta} |z'_{(i,j)} - z'_{(p,q)}|\right)^{-1} \tag{17.29}$$

where $\sum_{(p,q)\in\zeta} |z'_{(i,j)} - z'_{(p,q)}|$ is the aggregated absolute difference between the available CFA components inside ζ.

17.4.2.2 Interpolation Using Spatial and Spectral Correlations

Due to the lack of spectral information, the componentwise approach presented above can introduce processing errors which are amplified by subsequent demosaicking (Figure 17.7b). To reduce the number of color shifts and artifacts primarily caused by insufficient performance of the CFA resizing solution in the areas with edges and fine details, the interpolation output should be formed using both spectral and spatial image characteristics (Figure 17.7c). Taking advantage of spectral modelling principles reviewed in Chapter 1, this can be done by performing interpolation on spectral quantities rather than on pixel intensities.

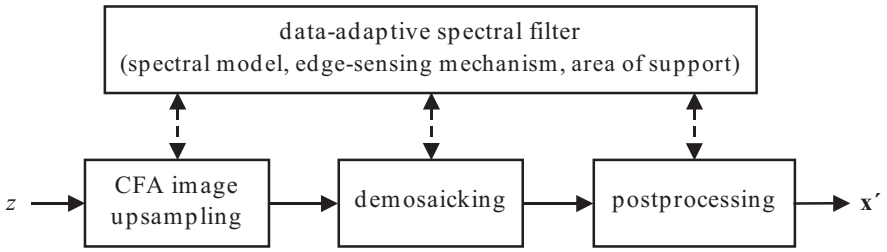


FIGURE 17.10

Partially integrated demosaicking and resizing using a series of unified processing steps.

For instance, the CFA upsampling solutions presented in References [31] and [32] first produce all missing G components in a componentwise manner and then estimate the missing R and B components as follows:

$$z'_{(r,s)} = z_{(r,s-1)} + \sum_{(i,j)\in\zeta} \{w_{(i,j)}(z'_{(i,j)} - z'_{(i,j-1)})\} \tag{17.30}$$

$$z'_{(r,s)} = z_{(r-1,s)} + \sum_{(i,j)\in\zeta} \{w_{(i,j)}(z_{(i,j)} - z_{(i-1,j)})\} \tag{17.31}$$

where the values $z'_{(i,j-1)}$ and $z'_{(i-1,j)}$ represent the spatially shifted neighboring locations used to overcome the lack of spectral information in location (i, j). The components $z_{(r-1,s)}$ and $z_{(r,s-1)}$ normalize the output of the interpolation process performed on color difference quantities $z'_{(i,j)} - z'_{(i,j-1)}$ and $z_{(i,j)} - z_{(i-1,j)}$. It should be noted that the above formulas can use any of two component spectral models presented in Chapter 1.

17.5 Joint Image Resizing and Demosaicking

17.5.1 Partially Integrated Demosaicking and Resizing

Solutions presented in References [33], [34], [35], and [36] are not true joint demosaicking / resizing schemes which produce the resized demosaicked image $\mathbf{x}'$ from the CFA image z in a single processing step. However, these solutions direct towards the joint process by *partially* integrating image resizing and demosaicking steps.

In References [33] and [34], image upsampling, demosaicking and postprocessing are performed using a series of *unified processing steps*. As shown in Figure 17.10, each of these steps employs the same data-adaptive spectral filterdata-adaptive spectral filter defined as follows:

$$\mathbf{x}'_{(r,s)} = \Lambda^{-1}\left(\mathbf{x}_{(r+a,s+b)}, \sum_{(i,j)\in\zeta} \{w_{(i,j)}\Lambda(\mathbf{x}'_{(i,j)}, \mathbf{x}'_{(i+a,j+b)})\}\right) \tag{17.32}$$

where $\Lambda(\cdot)$ and $\Lambda^{-1}(\cdot)$ denote, respectively, two-component spectral modelling and inverse spectral modelling functions described in detail in Chapter 1. The term $w_{(i,j)}$ represents the

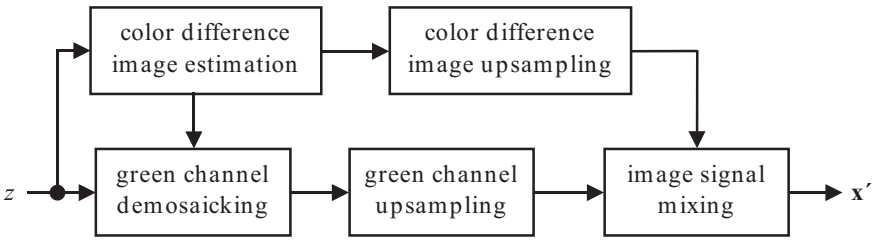


FIGURE 17.11

Partially integrated demosaicking and resizing by mixing the upsampled demosaicked green channel and two upsampled color difference images.

weight associated with the pixel location (i, j) in the neighborhood ζ. The weights are normalized versions of the edge-sensing weights calculated by following the rationale behind Equation 17.29. The nonzero terms a and b indicate the shift on the image lattice to compensate for the lack of samples in the closest neighborhood of the location under consideration (r,s) during the CFA image upsampling process (see Section 17.4.2.2). Unlike this initial processing step, subsequent demosaicking and demosaicked image postprocessing operations use $a = 0$ and $b = 0$. Algorithms designed within this framework are computationally efficient; however, they may produce color artifacts due to using spatially shifted color components in the upsampling step.

A computationally demanding but with significant performance improvements alternative to the above framework is presented in Reference [35]. As depicted in Figure 17.11, this solution forms two luminance-chrominance difference images to restore the luminance (green) channel in its original spatial resolution using demosaicking-like procedure. Then, both the demosaicked luminance image and two difference images are separately upsampled to the desired resolution. Finally, resized difference images are added to the resized luminance channel to produce the chrominance (red and blue) channels of the output image $\mathbf{x}'$. In each processing steps high-quality interpolation outputs (Figure 17.7d) are obtained by *fusing estimates* from two orthogonal directions as follows:

$$\mathbf{x}'_{(r,s)} = (w^{D_2}\mathbf{x}'^{D_1}_{(r,s)} + w^{D_1}\mathbf{x}'^{D_2}_{(r,s)})/(w^{D_1} + w^{D_2}) \tag{17.33}$$

where w^{D_1} and w^{D_2} are edge-sensing weights corresponding to directions D_1 and D_2, respectively. The weights can be expressed in the form of sample variances or their approximations and are solely calculated using the samples from the considered direction. Additional information on the estimation framework in Equation 17.33 can be found in Chapter 18.

Finally, the method presented in Reference [36] and summarized in Figure 17.12 demosaicks the green plane of the captured image using samples located along the edge as follows:

$$\mathbf{x}'_{(r,s)} = \begin{cases} f(\mathbf{x}'_{(r,s-1)}, \mathbf{x}'_{(r,s+1)}) & \text{for horizontal edges} \\ f(\mathbf{x}'_{(r-1,s)}, \mathbf{x}'_{(r+1,s)}) & \text{for vertical edges} \\ f(\mathbf{x}'_{(r-1,s)}, \mathbf{x}'_{(r,s-1)}, \mathbf{x}'_{(r,s+1)}, \mathbf{x}_{(r+1,s)}) & \text{for diagonal edges} \end{cases} \tag{17.34}$$

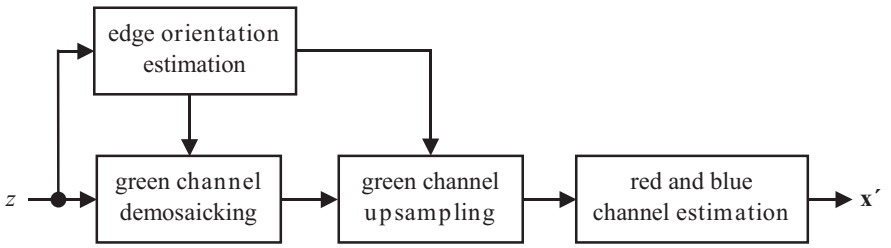


FIGURE 17.12

Partially integrated demosaicking and resizing by sharing edge orientation estimates.

where $f(\cdot)$ is an interpolation function. The demosaicking direction is determined as one minimizing local intensity gradients and local variances of color-difference values. In the next step, the *same edge orientation estimates* are used to direct the image upsampling process on the demosaicked green plane in order to perform demosaicking and zooming consistently and efficiently. After that, the red and the blue planes are obtained by estimating and adding local color difference signals to the green components in the enlarged demosaicked image (Figure 17.7e).

17.5.2 Fully Integrated Demosaicking and Resizing

Unlike solutions described in Section 17.5.1 which are limited to the upsampling operation, the framework presented in Reference [37] is a *true joint resizing / demosaicking* solution which allows for both image upsampling and downsampling and produces the resized demosaicked image (Figure 17.7f and Figure 17.8b) in a single processing step.

The framework operates by *partitioning* the input CFA image into blocks which are processed sequentially. For each block, an edge indicator is obtained in the form of variances or gradients of red, green, and blue CFA values. If any of the computed indicators exceeds a predetermined threshold meaning the presence of significant edges, the current block is further partitioned into smaller blocks for which new indicators have to be obtained. If none of the computed red, green, and blue indicators exceeds the threshold, the mean values of red, green and blue components are mapped into all pixel locations in the corresponding block in the resized image.

Figure 17.13 summarizes the algorithmic steps. After partitioning the CFA image z into a block $\{z_{(i,j)}; M_1 \leq i < M_2, N_1 \leq j < N_2\}$ of $(M_2 - M_1) \times (N_2 - N_1)$ CFA pixels $z_{(i,j)}$, color components are extracted using Equation 17.25-like procedure to produce a set of R

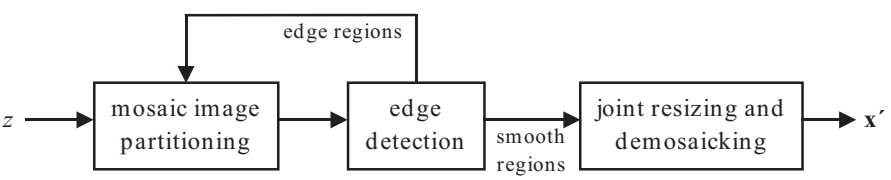


FIGURE 17.13

True joint demosaicking and resizing.

components, a set of B components and two sets of G components. For these four sets, the corresponding sample mean values μ_R, μ_{G_1}, μ_{G_2}, μ_B and sample variances υ_R, υ_{G_1}, υ_{G_2}, υ_B are calculated. Comparing the indicators, such as sample variances in this example, with the predetermined threshold ξ constitutes the constraint of the joint resizing and demosaicking operation:

$$\begin{array}{ll} \{\mathbf{x}'_{(i,j)} = [\mu_R, (\mu_{G_1} + \mu_{G_2})/2, \mu_B]^T; (i,j) \in \Omega\} & \text{if } \max\{\upsilon_R, \upsilon_{G_1}, \upsilon_{G_2}, \upsilon_B\} \leq \xi \\ \text{partition } \{z_{(i,j)}; M_1 \leq i < M_2, N_1 \leq j < N_2\} & \text{otherwise} \end{array} \tag{17.35}$$

where τ is a resizing factor and $\Omega = \{\tau M_1 \leq i < \tau M_2, \tau N_1 \leq j < \tau N_2\}$.

17.6 Conclusion

Image resizing solutions for digital cameras equipped with a color filter array placed on top of a single image sensor were the main focus of this chapter. Taking into consideration the position of the resizing step with respect to the demosaicking step in the imaging pipeline, as well as the nature of the resizing operations, the chapter provided a taxonomy of solutions for camera image upsampling and downsampling.

Efficient techniques which produce resized images of reasonable visual quality were identified for each of the three basic frameworks which can alter the spatial resolution of the image before or after demosaicking or even in a joint demosaicking / resizing process. The trade-off between performance and efficiency makes most solutions presented in this chapter indispensable tools for single-sensor imaging and its applications.

Such solutions constitute a basis for achieving the desired spatial resolution to view, manipulate, and print digital photos in a display and printer agnostic way. Moreover, in downsampling mode, they are essential for efficient storage and transmission of visual data whereas in upsampling mode, they can compensate for limitations in the camera's optical capabilities and the sensor's resolution. Thus, image resizing solutions have an extremely valuable position in digital imaging.

References

[1] R. Lukac and K.N. Plataniotis, *Color Image Processing: Methods and Applications*, ch. Single-sensor camera image processing, R. Lukac and K.N. Plataniotis (eds.), Boca Raton, FL: CRC Press / Taylor & Francis, October 2006, pp. 363–392.

[2] K. Fant, "A nonaliasing, real-time spatial transform technique," *IEEE Computer Graphics and Applications*, vol. 6, no. 1, pp. 71–80, January 1986.

[3] T. Sakamoto, C. Nakanishi, and T. Hase, "Software pixel interpolation for digital still cameras suitable for a 32-bit MCU," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 4, pp. 1342–1352, November 1998.

[4] W.T. Freeman, "Median filter for reconstructing missing color samples," U.S. Patent 4 724 395, February 1988.

[5] R. Lukac, K.N. Plataniotis, B. Smolka, and A.N. Venetsanopulos, "Vector operators for color image zooming," in *Proceedings of the IEEE International Symposium on Industrial Electronics*, Dubrovnik, Croatia, June 2005, vol. III, pp. 1273–1277.

[6] R.G. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, December 1981.

[7] S.E. Reichenbach and F. Geng, "Two-dimensional cubic convolution," *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 857–865, August 2003.

[8] R. Lukac, "Methods for fast enlargement of digital images," U.S. Patent, submitted, December 2007.

[9] J.W. Hwang and H.S. Lee, "Adaptive image interpolation based on local gradient features," *IEEE Signal Processing Letters*, vol. 11, no. 3, pp. 359–362, March 2004.

[10] Y. Cha and S. Kim, "The error-amended sharp edge (EASE) scheme for image zooming," *IEEE Transactions on Image Processing*, vol. 16, no. 6, pp. 1496–1505, June 2007.

[11] A.M. Darwish, M.S. Bedair, and S.I. Shaheen, "Adaptive resampling algorithm for image zooming," *IEE Proceedings - Vision, Image, Signal Processing*, vol. 144, no. 4, pp. 207–212, August 1997.

[12] S. Battiato, G. Gallo, and F. Stanco, "A locally adaptive zooming algorithm for digital images," *Image and Vision Computing*, vol. 20, no. 11, pp. 805–812, September 2002.

[13] Q. Wang and R.K. Ward, "A new orientation-adaptive interpolation method," *IEEE Transactions on Image Processing*, vol. 16, no. 4, pp. 889–900, April 2007.

[14] S. Thurnhofer and S.K. Mitra, "Edge-enhanced image zooming," *Optical Engineering*, vol. 35, no. 7, pp. 1862–1869, July 1996.

[15] X. Li and M.T. Orchard, "New edge-directed interpolation," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1521–1527, October 2001.

[16] D.D. Muresan and T.W. Parks, "Adaptively quadratic (AQua) image interpolation," *IEEE Transactions on Image Processing*, vol. 13, no. 5, pp. 690–698, May 2004.

[17] L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2226–2238, August 2006.

[18] N. Herodotou and A.N. Venetsanopoulos, "Colour image interpolation for high resolution acquisition and display devices," *IEEE Transactions on Consumer Electronics*, vol. 41, no. 4, pp. 1118–1126, November 1995.

[19] R. Lukac, B. Smolka, K. Martin, K.N. Plataniotis, and A.N. Venetsanopulos, "Vector filtering for color imaging," *IEEE Signal Processing Magazine; Special Issue on Color Image Processing*, vol. 22, no. 1, pp. 74–86, January 2005.

[20] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proceedings of the IEEE*, vol. 78, no. 4, pp. 678–689, April 1990.

[21] P.E. Trahanias and A.N. Venetsanopoulos, "Vector directional filters: A new class of multichannel image processing filters," *IEEE Transactions on Image Processing*, vol. 2, no. 4, pp. 528–534, October 1993.

[22] R. Lukac and K.N. Plataniotis, *Advances in Imaging and Electron Physics*, ch. A taxonomy of color image filtering and enhancement solutions, pp. 187–264, San Diego, CA: Elsevier / Academic Press, February/March 2006.

[23] K.N. Plataniotis, D. Androutsos, and A.N. Venetsanopoulos, "Adaptive fuzzy systems for multichannel signal processing," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1601–1622, September 1999.

[24] ISO/IEC, "Information technology - digital compression and coding of continuous-tone still images: Requirements and guidelines." ISO/IEC International Standard 10918-1, ITU-T Recommendation T.81, 1994.

[25] J. Mukherjee and S.K. Mitra, *Color Image Processing: Methods and Applications*, ch. Resizing color images in the compressed domain, R. Lukac and K.N. Plataniotis (eds.), Boca Raton, FL: CRC Press / Taylor & Francis, October 2006, pp. 129–156.

[26] ISO/IEC, "Information technology - JPEG 2000 image coding system." ISO/IEC International Standard 15444-1, ITU Recommendation T.800, 2000.

[27] S. Battiato, A.R. Bruna, A. Buemi, and A. Castorina, "Analysis and characterization of JPEG 2000 standard for imaging devices," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 773–779, November 2003.

[28] T. Acharya and W. Metz, "Scaling algorithm for efficient color representation recovery in video," U.S. Patent 6 236 433, May 2001.

[29] C.C. Koh, J. Mukherjee, and S.K. Mitra, "New efficient methods of image compression in digital cameras with color filter array," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1448–1456, November 2003.

[30] R. Lukac and K.N. Plataniotis, "Single-sensor camera image compression," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 299–307, May 2006.

[31] R. Lukac, K.N. Plataniotis, and D. Hatzinakos, "Color image zooming on the Bayer pattern," *IEEE Transactions on Circuit and Systems for Video Technology*, vol. 15, no. 11, pp. 1475–1492, November 2005.

[32] R. Lukac and K.N. Plataniotis, "Digital zooming for color filter array based image sensors," *Real-Time Imaging, Special Issue on Spectral Imaging II*, vol. 11, no. 2, pp. 129–138, April 2005.

[33] R. Lukac, K. Martin, and K.N. Plataniotis, "Digital camera zooming based on unified CFA image processing steps," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 15–24, February 2004.

[34] R. Lukac and K.N. Plataniotis, "Camera image processing system for Bayer CFA based imaging devices," in *Proceedings of the 2004 International Workshop on Multimedia Signal Processing*, Siena, Italy, September 2004, pp. 247–250.

[35] L. Zhang and D. Zhang, "A joint demosaicking-zooming scheme for single chip digital color cameras," *Computer Vision and Image Understanding*, vol. 107, no. 1-2, pp. 14–25, July/August 2007.

[36] K.H. Chung and Y.H. Chan, "A low-complexity joint color demosaicking and zooming algorithm for digital camera," *IEEE Transactions on Image Processing*, vol. 16, no. 7, pp. 1705–1715, July 2007.

[37] I. Baharav, R. Kakarala, and D. Vook, "Adaptive color plane interpolation in single sensor color electronic camera," U.S. Patent 6 989 862, January 2006.

18

Video-Demosaicking

Lei Zhang and Wei Lian

18.1 Introduction

With advances in hardware and software, more and more digital cameras allow for the recording of digital video. Popular digital still image and video cameras use cost-effective single-sensor technology which acquires the visual information using a color filter array (CFA) [1], [2]. Since the original color information is subsampled by the CFA, a process called demosaicking [3], [4] has to be employed in the camera imaging pipeline to reconstruct the color information from the acquired CFA sensor readings which constitute a mosaic-like image. Refer to Chapter 1 for details on single-sensor imaging fundamentals.

Many color demosaicking algorithms proposed in the past are intended for digital still images [5], [6], [7], [8], [9]. However, digital video represents a three-dimensional image

signal or a time sequence of two-dimensional images (frames). By omitting the essential temporal characteristics, even sophisticated spatial demosaicking methods often produce an output image sequence with motion artifacts. To overcome the limitation of spatial demosaicking, additional knowledge and constraints of the original color signals are needed during video-demosaicking. In particular, the temporal dimension of a sequence of CFA images often reveals new valuable information about the color composition of the scene which would be unavailable in the spatial domain of individual CFA frames. Thus, in addition to spatial and spectral correlations used commonly by today's spatial demosaicking solutions, the temporal correlation of adjacent frames should be exploited by estimating the camera and object motion in order to aid the video-demosaicking process to produce high-quality color video.

Several works have been reported on demosaicking of spatiotemporal CFA data. In Reference [10], a hybrid multi-frame color demosaicking and super-resolution algorithm was proposed to reconstruct a high resolution color image sequence from a set of low resolution CFA frames by using a multi-term cost function and minimizing iteratively the estimation error. Another super-resolution scheme, presented in Reference [11], uses a regularized optimization approach to simultaneously recover the missing color components and improve the spatial resolution of the video sequence. The methods introduced in References [12] and [13] are true video-demosaicking solutions. They use multistage spatiotemporal processing concepts to restore the full-color information without motion estimation efficiently and in reasonable visual quality. The solution in Reference [12] uses unidirectional supporting windows to obtain the spectral estimates in the spatial and temporal directions. Using the adaptive, edge-sensing weights, the unidirectional estimates are combined to produce a final, demosaicked output. This concept was extended in Reference [13] by producing spectral estimates using higher-order, bidirectional windows in a refined multi-stage architecture. Since bidirectional windows allow for additional spatial information to be used to populate the spatiotemporal CFA sequences, higher accuracy of the video-demosaicking process can be achieved while keeping the computational complexity acceptable.

This chapter presents a motion estimation and optimal — in the sense of minimum mean-squared error (MSE) — data fusion based framework for video-demosaicking. First the sequence of CFA frames can be demosaicked in a spatial manner using solely the information from the actual frame. Since a digital video is a three-dimensional signal correlated in both spatial and temporal senses, the spatial demosaicking solution can be extended to spatiotemporal video-demosaicking by exploiting the temporal redundancy. In particular, motion estimation is implemented to compute the relative displacement between camera and object and to align the adjacent frames so that the temporal correlation can be better utilized for improving accuracy of the video-demosaicking process. After motion compensation, the spatially demosaicked results in adjacent frames will be optimally fused to obtain the temporally enhanced demosaicking output.

This chapter is structured as follows. Section 18.2 introduces the concepts of spatial, temporal and spatiotemporal processing for video sequence. Section 18.3 briefly describes the CFA image/sequence acquisition and representation. Section 18.4 discusses CFA video-demosaicking without motion compensation. Section 18.5 presents the motion compensated video-demosaicking framework as well as the experimental results. Section 18.6 concludes this chapter.

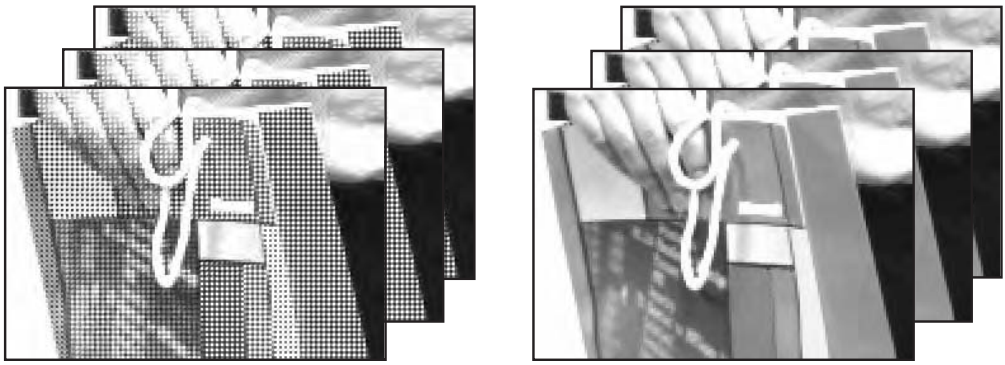


FIGURE 18.1

The concept of video-demosaicking. The picture on the left shows the sequence of mosaic frames. The picture on the right shows the sequence of demosaicked, full-color frames.

18.2 Fundamentals of Color Filter Array Image Sequence Processing

A color filter array (CFA) image sequence is a three-dimensional signal or a time sequence of two-dimensional gray-scale mosaic-like frames captured by a single-sensor digital camera. Each pixel within a CFA sequence can be denoted as $\boldsymbol{C}_k(n,m)$, where subscript k represents its temporal index and pair (n,m) represents its spatial location in that frame. Demosaicking pixels $\boldsymbol{C}_k(n,m)$ restores the two color components per spatial location, resulting in the sequence of full-color frames with pixels $\boldsymbol{F}_k(n,m,c)$, where c denotes the color channel. The demosaicked video has higher dimensionality than CFA video because each demosaicked pixel is a vector of color components, usually red, green and blue components as opposed to a single CFA sensor reading per spatial location. Figure 18.1 demonstrates differences between the CFA image sequence and its demosaicked version.

Since the natural CFA video is a highly non-stationary three-dimensional image signal, in order to reduce the demosaicking errors and color artifacts, video-demosaicking uses localized operations inside the area of support centered in the location under consideration. Based on the dimensionality of the area of support, there are three basic paradigms for video-demosaicking. Namely, temporal demosaicking employs a one-dimensional processing window oriented along the temporal axis, thus using one pixel value from each frame. Spatial demosaicking processes each frame separately using a two-dimensional area of support within the actual frame. Finally, spatiotemporal demosaicking combines the two above approaches by using a series of spatial windows placed along the temporal axis, thus forming a three-dimensional area of support. Because video usually exhibits significant motion characteristics caused by the moving objects and/or camera motion, estimating motion, at the expense of increased computational complexity, can enhance the performance of temporal and spatiotemporal demosaicking by increasing the correlation between the inputs. More detailed information on video-demosaicking paradigms follows below.

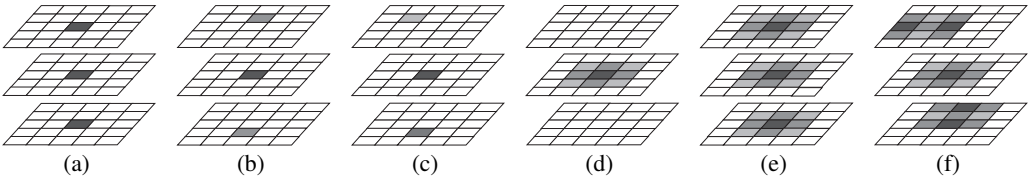


FIGURE 18.2 (See color insert.)

Examples of basic windows used in video-demosaicking: (a) non-motion compensated temporal window, (b, c) motion compensated temporal windows, (d) spatial window, (e) non-motion compensated spatiotemporal window, and (f) motion compensated spatiotemporal window.

18.2.1 Temporal Demosaicking

As mentioned above, temporal demosaicking uses one-dimensional area of support. Operating without motion estimation implies that temporal demosaicking should be performed using pixels occupying the same spatial location (n,m) in several consecutive CFA frames. However, since all such pixels are captured using the same color filter (e.g., blue in Figure 18.2a), the two missing color components cannot be obtained. Symmetric temporal windows (Figure 18.2b) placed along the estimated motion vector cannot completely overcome the problem, as the pixels inside such windows correspond to two of three necessary types of color filters. This is not the case for motion compensated asymmetric windows (Figure 18.2c) which contain all three color components. Unfortunately, this often leads to processing errors, in particular if the motion estimation is not accurate or the relative motion between adjacent frames is complex. Due to its insufficient performance, temporal video-demosaicking is rarely used in practice.

18.2.2 Spatial Demosaicking

To overcome the lack of spectral information during demosaicking, the supporting window should contain all types of color components. As shown in Figure 18.2d, two-dimensional windows used in spatial demosaicking satisfy the constraint. Only the pixels within the window in the current frame are used to demosaick the underlying pixel. Most of the existing techniques, such as those described in Chapters 7 to 9, are spatial demosaicking methods. Spatial demosaicking solutions process each frame of CFA video separately, that is, without exploiting the temporal correlations. They generally have reasonable performance, although the omission of temporal characteristics of the captured video in the demosaicking process often results in various visual impairments such as motion blur and demosaicking artifacts in areas with abrupt color transitions or sharp edges.

18.2.3 Spatiotemporal Demosaicking

Spatiotemporal demosaicking exploits all the three types of correlations (spatial, temporal, and spectral) available in a CFA video sequence. Therefore, the accuracy of spatiotemporal demosaicking is generally higher than that of spatial and temporal demosaicking. On the other hand, spatiotemporal demosaicking solutions have naturally higher computational cost and memory requirements than their spatial and temporal counterparts.

As in temporal demosaicking, spatiotemporal demosaicking solutions can operate with and without motion estimation. In the former case, a number of spatial windows centered in the same spatial location in several consecutive CFA frames are used to estimate the color components missing at the location under consideration (Figure 18.2e). In the latter case, windows are centered along the direction matching the estimated motion vector (Figure 18.2f) to increase the correlation among the inputs of the demosaicking solution. It should be noted that the performance of the latter approach highly depends on the accuracy of motion estimation. Though many motion estimation techniques have been presented in the literature [14], [15], [16], [17], [18], most of them were designed for standard (non-mosaic) video. Unfortunately, so far there is no systematic study of motion estimation on single-sensor camera captured video. One possibility of performing motion compensated CFA video processing is to demosaick the green channel of CFA frames by using spatial demosaicking first and then to apply the conventional motion estimation techniques to demosaicked green channels to estimate the motion vectors.

18.3 Spatial Demosaicking By Fusing the Directional Signals

Directional filtering is a popular approach to spatial demosaicking due to its good performance and relative computational simplicity. Many demosaicking solutions designed within this framework follow the rationale behind the method in Reference [5]. Namely, the second order gradients of blue and red samples and the first order gradient of green samples are used to demosaick the green channel. Then, the red and blue samples are demosaicked similarly with the correction of the second order gradients of the green samples.

An advanced approach to directional demosaicking was proposed in Reference [9]. To reduce the processing errors, the demosaicking outputs are formed by fusing the directional signals using the linear minimum mean square error estimation method. The details are presented below.

18.3.1 Primary Difference Signal Formulation

For ease of presentation and without loss of generality, we examine the case depicted in Figure 18.3a. In this figure, a column and a row of alternating green and red components intersect at a red sampling position with the missing green component to be demosaicked. The symmetric case of estimating the missing green components at the blue positions of the Bayer pattern can be handled in the same way. Denote by R_0 the red sample at the center of the window. Its interlaced red and green neighbors in the horizontal direction are labeled as R_i^h with $i \in \{\cdots,-2,2,\cdots\}$ and G_i^h with $i \in \{\cdots,-1,1,\cdots\}$, respectively. Similarly, the red and green neighbors of R_0 in the vertical direction are R_j^v with $j \in \{\cdots,-2,2,\cdots\}$ and G_j^v with $j \in \{\cdots,-1,1,\cdots\}$, respectively. The sample R_0 at the intersection can be taken as R_0^h or R_0^v at will.

Most spatial-spectral demosaicking methods, such as those presented in References [5], [8], and [9], are based on an assumption that the difference between the green channel and

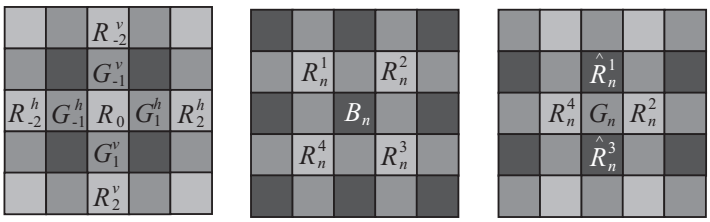


FIGURE 18.3

Pixel configurations used in demosaicking: (a) a row and a column of mosaic data that intersect at a red sampling position, (b) a blue sample and its four nearest red neighbors, and (c) a green sample and its two original and two estimated red neighbors.

the red/blue channel is a low-pass signal. Let $\Delta_0 = G_0 - R_0$ be the unknown difference between green and red channels at the sample position of R_0. The idea is to obtain an estimate $\widehat{\Delta}_0$ of Δ_0 and then recover the missing green sample as $G_0 \approx R_0 + \widehat{\Delta}_0$. The reason for estimating the color difference signal rather than the green intensity signal directly from its neighbors is that the color difference signal is much smoother than the original intensity signal. Therefore, performing demosaicking operations on the color difference domain of the captured images can greatly reduce the processing error compared to demosaicking on the intensity domain.

The differences between the green and red channels form the so-called *primary difference signal*:

$$\Delta_{g,r}(n) = G_n - R_n \tag{18.1}$$

where n is the position index of the pixels.

Coarse measurements of $\Delta_{g,r}$ can be obtained using an arbitrary demosaicking method. For example, employing the second-order Laplacian filter of Reference [5], the missing green components corresponding to the original red components R^h_i or R^v_j are obtained as follows:

$$\widehat{G}^h_i = \frac{1}{2}(G^h_{i-1} + G^h_{i+1}) + \frac{1}{4}(2 \cdot R^h_i - R^h_{i-2} - R^h_{i+2}) \tag{18.2}$$

$$\widehat{G}^v_j = \frac{1}{2}(G^v_{j-1} + G^v_{j+1}) + \frac{1}{4}(2 \cdot R^v_j - R^v_{j-2} - R^v_{j+2}) \tag{18.3}$$

Similarly, for any original green component G^h_i or G^v_j, the corresponding red component is demosaicked as follows:

$$\widehat{R}^h_i = \frac{1}{2}(R^h_{i-1} + R^h_{i+1}) + \frac{1}{4}(2 \cdot G^h_i - G^h_{i-2} - G^h_{i+2}) \tag{18.4}$$

$$\widehat{R}^v_j = \frac{1}{2}(R^v_{j-1} + R^v_{j+1}) + \frac{1}{4}(2 \cdot G^v_j - G^v_{j-2} - G^v_{j+2}) \tag{18.5}$$

The demosaicked values from Equations 18.2 to 18.5 can be used to obtain two estimates of $\Delta_{g,r}$ in horizontal and vertical directions as follows:

$$\widehat{\Delta}^h_{g,r}(i) = \begin{cases} \widehat{G}^h_i - R^h_i, \text{ in R CFA locations} \\ G^h_i - \widehat{R}^h_i, \text{ in G CFA locations} \end{cases}, \quad \widehat{\Delta}^v_{g,r}(i) = \begin{cases} \widehat{G}^v_i - R^v_i, \text{ in R CFA locations} \\ G^v_i - \widehat{R}^v_i, \text{ in G CFA locations} \end{cases} \tag{18.6}$$

The estimation errors associated with $\widehat{\Delta}^h_{g,r}$ and $\widehat{\Delta}^v_{g,r}$ are:

$$\varepsilon^h_{g,r} = \Delta_{g,r} - \widehat{\Delta}^h_{g,r}; \quad \varepsilon^v_{g,r} = \Delta_{g,r} - \widehat{\Delta}^v_{g,r} \tag{18.7}$$

Since $\widehat{\Delta}^h_{g,r}$ and $\widehat{\Delta}^v_{g,r}$ can be seen as two observations of $\Delta_{g,r}$, and accordingly $\varepsilon^h_{g,r}$ and $\varepsilon^v_{g,r}$ as the corresponding *directional demosaicking noise terms*, Equation 18.7 can be rewritten as:

$$\widehat{\Delta}^h_{g,r} = \Delta_{g,r} - \varepsilon^h_{g,r}; \quad \widehat{\Delta}^v_{g,r} = \Delta_{g,r} - \varepsilon^v_{g,r} \tag{18.8}$$

Now, the demosaicking problem can be formulated as the derivation of the missing green samples using Equation 18.1 based on the estimation of $\Delta_{g,r}$ from the two observation sequences $\{\widehat{\Delta}^h_{g,r}\}$ and $\{\widehat{\Delta}^v_{g,r}\}$.

18.3.2 Directional Estimation of Primary Difference Signals

To simplify the notation, let x be the true primary difference signal $\Delta_{g,r}$, the term y be the associated observation $\widehat{\Delta}^h_{g,r}$ or $\widehat{\Delta}^v_{g,r}$, and υ denote the associated demosaicking noise $\varepsilon^h_{g,r}$ or $\varepsilon^v_{g,r}$. Obviously, the relationship between the above signals is additive:

$$y(n) = x(n) + \upsilon(n) \tag{18.9}$$

which allows $\widehat{x} = E[x/y] = \int xp(x/y)dx$ being the optimal minimum mean square error estimation (MMSE) of x. However, estimating x under the MMSE constraint is very difficult, if possible at all, because p(x/y) is seldom known in practice. To overcome the problem, x can be estimated from y using the linear MMSE (LMMSE) framework which provides a good approximation to the MMSE formulation and can be relatively efficiently implemented. Particularly, if $x(n)$ and $\upsilon(n)$ are locally Gaussian processes (a reasonable assumption for natural images and videos), then the spatially adaptive LMMSE will be equivalent to MMSE. The LMMSE of x is computed as:

$$\widehat{x} = E[x] + \frac{\mathrm{Cov}(x,y)}{\mathrm{Var}(y)}(y - E[y]) \tag{18.10}$$

It was shown in Reference [9] that υ represents a zero-mean random process and that υ is almost uncorrelated with $\Delta_{g,r}$. Therefore, Equation 18.10 can be rewritten as follows:

$$\widehat{x} = \mu_x + \frac{\sigma_x^2}{(\sigma_x^2 + \sigma_\upsilon^2)}(y - \mu_x) \tag{18.11}$$

where $\mu_x = E[x]$, $\sigma_x^2 = \mathrm{Var}(x)$, and $\sigma_\upsilon^2 = \mathrm{Var}(\upsilon)$.

From Equation 18.11, it can be seen that the calculation of the LMMSE estimate $\widehat{x}(n)$ requires to determine μ_x, σ_x and σ_υ from observation data $y(n)$. In order to make the estimate $\widehat{x}(n)$ spatially adaptive, these parameters should be estimated locally in the neighborhood of $y(n)$.

By examining the power spectrum density functions of $x(n)$ and $\upsilon(n)$ with real images, it is found that the power of x is concentrated in low-frequency bands, whereas the power of υ is spread in relatively high-frequency bands [9]. Since x and υ have distinct power

spectrum, passing y through a low-pass filter can remove the noise effectively. Denoting by $\{h(k)\}$ the response sequence of a low-pass filter provides

$$y_s(n) = (y * h)(n) = \sum_{k=-\infty}^{\infty} y(n-k) \cdot h(k) \tag{18.12}$$

where $*$ is the convolution operator. In general, $\{h(k)\}$ can be the Gaussian smoothing filter with coefficients $h(k) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{k^2}{2\sigma^2}}$ and parameter σ which is used to control the shape of the filter response.

Assuming that the random process $x(n)$ is ergodic and stationary, its mean value $\mu_x(n)$ can be estimated by the neighboring data of $y(n)$. The low-pass filter output $y_s(n)$ is a weighted average of $y(n)$ and its neighbors, and it is much closer to $x(n)$ than $y(n)$. For a $2L+1$ dimensional vector $\boldsymbol{Y}_n^s = [y_s(n-L) \cdots y_s(n) \cdots y_s(n+L)]$ centered at $y_s(n)$, the value of $\mu_x(n)$ can be estimated as follows:

$$\mu_x(n) = \frac{1}{2L+1} \sum_{k=1}^{2L+1} \boldsymbol{Y}_n^s(k) \tag{18.13}$$

and the value of $\sigma_x^2(n)$ can be obtained as

$$\sigma_x^2(n) = \frac{1}{2L+1} \sum_{k=1}^{2L+1} (\boldsymbol{Y}_n^s(k) - \mu_x(n))^2 \tag{18.14}$$

Since $y_s(n)$ is an approximation of $x(n)$, it follows that $y(n) - y_s(n)$ is an approximation of $\upsilon(n)$ and thus $\sigma_\upsilon^2(n)$ can be estimated as follows:

$$\sigma_\upsilon^2(n) = \frac{1}{2L+1} \sum_{k=1}^{2L+1} (\boldsymbol{Y}_n^s(k) - \boldsymbol{Y}_n(k))^2 \tag{18.15}$$

where $\boldsymbol{Y}_n = [y(n-L) \cdots y(n) \cdots y(n+L)]$ is a $2L+1$ dimensional vector centered at $y(n)$.

For each sample $x(n)$ to be estimated, the corresponding parameters $\mu_x(n)$, $\sigma_x^2(n)$ and $\sigma_\upsilon^2(n)$ are computed and substituted into Equation 18.11 to yield $\widehat{x}(n)$ which is the nearly LMMSE estimate of $x(n)$. Assuming that $\widetilde{x}(n) = x(n) - \widehat{x}(n)$ is the estimation error, the variance of $\widetilde{x}(n)$ can be expressed as follows:

$$\sigma_{\widetilde{x}}^2(n) = E[\widetilde{x}^2(n)] = \sigma_x^2(n) - \frac{\sigma_x^2(n)}{(\sigma_x^2(n) + \sigma_\upsilon^2(n))} \tag{18.16}$$

18.3.3 Demosaicking the Green Channel

By applying the approach from Section 18.3.2 to image rows and columns, two LMMSE estimates, $\widehat{x}_h(n)$ and $\widehat{x}_v(n)$, of the primary difference signal $x(n)$ are obtained in each demosaicking location. Using the corresponding demosaicking errors, $\widehat{x}_h(n)$ and $\widehat{x}_v(n)$, implies:

$$\begin{cases} \widehat{x}_h(n) = x(n) - \widetilde{x}_h(n) \\ \widehat{x}_v(n) = x(n) - \widetilde{x}_v(n) \end{cases} \tag{18.17}$$

Either $\widehat{x}_h(n)$ or $\widehat{x}_v(n)$ exploits the correlation of $x(n)$ with its neighbors in a particular direction. A more accurate estimate of $x(n)$ can be obtained by fusing the two directional LMMSE estimates as follow:

$$\widehat{x}_w(n) = w_h(n)\widehat{x}_h(n)w_v(n)\widehat{x}_v(n) \tag{18.18}$$

where $w_h(n) + w_v(n) = 1$. The weights $w_h(n)$ and $w_v(n)$ are determined to minimize the mean square error of $\widehat{x}_w(n)$: $\sigma^2_{\widetilde{x}_w}(n) = E[\widetilde{x}^2_w(n)] = E[(x(n) - \widehat{x}_w(n))^2]$, that is:

$$\sigma^2_{\widetilde{x}_w}(n) = w^2_h(n)\sigma^2_{\widetilde{x}_h}(n) + w^2_v(n)\sigma^2_{\widetilde{x}_v}(n) + 2w_h(n)w_v(n)E[\widetilde{x}_h(n)\widetilde{x}_v(n)] \tag{18.19}$$

where $\sigma^2_{\widetilde{x}_h}(n)$ and $\sigma^2_{\widetilde{x}_v}(n)$ are the variances of estimation errors $\widetilde{x}_h(n)$ and $\widetilde{x}_v(n)$.

Generally, the correlation between variables $\widetilde{x}_h$ and $\widetilde{x}_v$ is weak in edge areas of natural images. Even for highly correlated $\widetilde{x}_h$ and $\widetilde{x}_v$ (meaning that the two estimates $\widehat{x}_h$ and $\widehat{x}_v$ are close to each other), there exists still some small difference between w_h and w_v. Assuming that $\widetilde{x}_h$ and $\widetilde{x}_v$ are approximately uncorrelated, the optimal weights can be expressed as:

$$w_h(n) = \frac{\sigma^2_{\widetilde{x}_v}(n)}{\sigma^2_{\widetilde{x}_h}(n) + \sigma^2_{\widetilde{x}_v}(n)}, \quad w_v(n) = \frac{\sigma^2_{\widetilde{x}_h}(n)}{\sigma^2_{\widetilde{x}_h}(n) + \sigma^2_{\widetilde{x}_v}(n)} \tag{18.20}$$

The method presented above requires calculating the directional weighted estimates of the green-red primary difference signal $\Delta_{g,r}(n)$ in each red pixel position $\boldsymbol{R}_n$ and the green-blue primary difference signal $\Delta_{g,b}(n)$ in each blue pixel position $\boldsymbol{B}_n$. The green channel of the captured image is restored by demosaicking the missing green samples as follows:

$$\widehat{\boldsymbol{G}}_n = \boldsymbol{R}_n + \Delta_{g,r}(n) \quad \text{or} \quad \widehat{\boldsymbol{G}}_n = \boldsymbol{B}_n + \Delta_{g,b}(n) \tag{18.21}$$

18.3.4 Demosaicking the Red and Blue Channels

Following standard practice [4], [5], in order to avoid color shifts and artifacts the demosaicked green channel is used to restore the red and blue channel. This is accomplished in two steps; first the red (blue) components are demosaicked in the blue (red) CFA locations and then both red and blue components are obtained in the green CFA locations.

Figure 18.3b depicts the scenario for the former case where four red components $\boldsymbol{R}^1_n$, $\boldsymbol{R}^2_n$, $\boldsymbol{R}^3_n$ and $\boldsymbol{R}^4_n$ are surrounding the blue CFA location with the component $\boldsymbol{B}_n$ in the diagonal directions. Using the samples from the demosaicked green plane, the corresponding difference values $\Delta^1_{n,gr}$, $\Delta^2_{n,gr}$, $\Delta^3_{n,gr}$ and $\Delta^4_{n,gr}$ can be obtained to estimate the green-red primary difference signal $\Delta_{n,gr} = (\Delta^1_{n,gr} + \Delta^2_{n,gr} + \Delta^3_{n,gr} + \Delta^4_{n,gr})/4$. This signal is added to the demosaicked green component $\widehat{\boldsymbol{G}}_n$ to restore the missing red component as $\widehat{\boldsymbol{R}}_n = \widehat{\boldsymbol{G}}_n - \Delta_{n,gr}$. Similar approach is used to demosaick the blue component at the red CFA location as $\widehat{\boldsymbol{B}}_n = \widehat{\boldsymbol{G}}_n - \Delta_{n,gb}$ with $\Delta_{n,gb} = (\Delta^1_{n,gb} + \Delta^2_{n,gb} + \Delta^3_{n,gb} + \Delta^4_{n,gb})/4$.

After the above step has been completed, the process continues by demosaicking the red and blue components in the green CFA locations. Figure 18.3c shows the configuration with the available red components (original or estimated) located in horizontal and vertical directions. Note that at those positions we have already interpolated the green component.

Since green components are available at all pixel locations, the primary difference signals can be estimated as $\Delta_{n,gr} = (\Delta^1_{n,gr} + \Delta^2_{n,gr} + \Delta^3_{n,gr} + \Delta^4_{n,gr})/4$. The missing red component is then demosaicked as $\widehat{\boldsymbol{R}}_n = \boldsymbol{G}_n - \Delta_{n,gr}$. Similarly, the blue components are demosaicked as $\widehat{\boldsymbol{B}}_n = \boldsymbol{G}_n - \Delta_{n,gb}$ using the corresponding primary difference signal $\Delta_{n,gb} = (\Delta^1_{n,gb} + \Delta^2_{n,gb} + \Delta^3_{n,gb} + \Delta^4_{n,gb})/4$. Completing this step restores the full-color information in the demosaicked image.

18.4 Motion Compensated Video-Demosaicking

In References [19] and [20], motion compensated video-demosaicking algorithms were proposed by using motion estimation and data fusion. In this section, we describe in detail this scheme. Since natural video usually exhibits significant correlations between the frames, it is highly desired to incorporate the temporal or motion characteristics into the video-demosaicking process. In single-sensor imaging, motion characteristics can be basically estimated by operating directly on the CFA mosaic data or on the demosaicked frames.

18.4.1 Motion Estimation in the Mosaic Domain

Although motion estimation is one of the most researched problems in video processing, the lack of motion estimation methods for CFA mosaic data still persists. This is caused by the increased complexity of motion estimation in CFA video compared to standard (non-mosaic) gray-scale or color video as well as due to the fact that video-demosaicking became the subject of academic research only recently.

Few motion estimation algorithms designed for CFA sequences have been reported. In Reference [11], the motion estimation problem in CFA video was treated from a super-resolution reconstruction perspective. The motion vector is calculated by minimizing a similarity function $C(s)$, which is equivalent to searching for the minimum of the sum of squared differences (SSD) between the observed CFA image and the simulated CFA image with motion vector s. Since estimating motion directly in the CFA domain is rather complex, it may be more suitable to obtain the motion vectors in the demosaicked domain.

18.4.2 Motion Estimation in the Demosaicked Domain

Traditional techniques of motion estimation for gray-scale and color image sequences, such as techniques presented in References [14], [15], [16], [17], and [18], cannot be directly applied to mosaic data due to the underlying mosaic layout of the CFA. However, they can be applied to demosaicked frames serving as intermediate signals for motion compensated video-demosaicking which can be implemented through the registration of the frames along estimated motion vectors or enhancement of spatially demosaicked frames using temporal characteristics of the acquired video. To demonstrate the concept of motion compensated video-demosaicking, the block-based motion estimation technique [18],

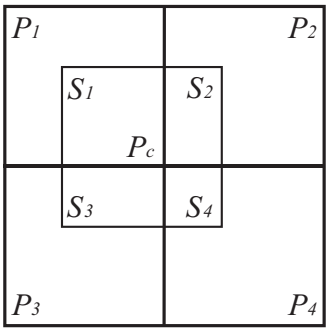


FIGURE 18.4

Arrangement of the pixels after motion compensation. © 2006 IEEE

which is widely used in MPEG 2/4 and other video coding standards due to its good balance between estimation accuracy and computational complexity, will be employed below. The cost of processing can be reduced by estimating the motion on the demosaicked green channel only, as most CFAs contains more green filters than red and blue filters. The green channel can be restored using any demosaicking algorithm; however, spatial algorithms such as one presented in Section 18.3 usually produce good intermediate results for subsequent temporal enhancement.

Let G be the original green sample and $\widehat{G}$ its estimated version obtained by the spatial demosaicking solution. The terms $\boldsymbol{M}_0$ and $\boldsymbol{M}_{i,j}$ denote, respectively, a block in the current frame and a block in a reference frame with an integer displacement (i, j). Let $\boldsymbol{M}_c$ stand for the matched block to $\boldsymbol{M}_0$ from the reference frame and (τ_x, τ_y) be the motion vector. If the motion vectors are desired to be in the whole pixel precision, then (τ_x, τ_y) will be searched from (i, j) within a search range. If (τ_x, τ_y) are supposed to follow the subpixel precision, $\boldsymbol{M}_c$ should be re-sampled from the reference frame. In the literature the value of a pixel is commonly modelled as the integral of the light over a unit square. Let P_c be a pixel in $\boldsymbol{M}_c$ and the associated unit square of P_c will overlap with four unit squares, which are associated with P_1, P_2, P_3 and P_4 which are four pixels in the reference frame (Figure 18.4). Since the areas of the overlaps S_1, S_2, S_3 and S_4 associated with P_1, P_2, P_3 and P_4, respectively, can be computed from the fractional part of the real valued (τ_x, τ_y), the value of P_c can be calculated as $P_c = \sum_{i=1}^{4} S_i P_i$.

Due to the structure of the sampling grid of the green channel, two of the four squares P_1, P_2, P_3 and P_4 are the original green samples G and the other two are the demosaicked green samples $\widehat{G}$. To emphasize original samples G over demosaicked samples $\widehat{G}$, each P_i term can be associated with a weight c_i leading to $P_c = \sum_{i=1}^{4} c_i S_i P_i$ where $c_i > 1$ if P_i is an original green sample, $c_i < 1$ if P_i is a demosaicked green sample, and $\sum_{i=1}^{4} c_i = 4$.

18.4.3 Video-Demosaicking by Temporal Enhancement

Following the procedure shown in Figure 18.5, the motion characteristics extracted by operating on spatially demosaicked green channels of the captured video are used to register the adjacent frames. To improve the accuracy of the restored green channel, original

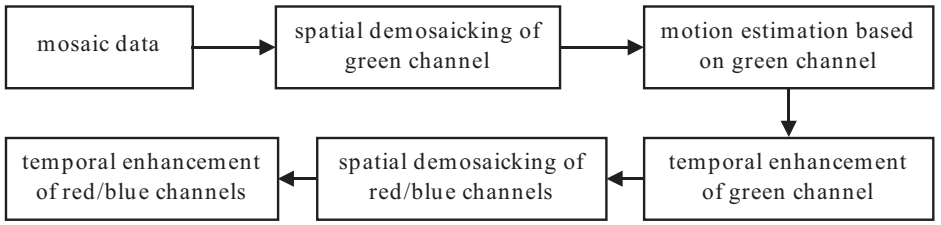


FIGURE 18.5

Video-demosaicking scheme with temporal enhancement of spatially demosaicked samples. © 2006 IEEE

green samples from adjacent frames are fused along the temporal axis with the spatially demosaicked samples from the current frame. The resulting improved green channel will serve as an anchor to demosaick the red and blue channels using both the intraframe and interframe information. The red and blue channels are first restored spatially with the help of temporally enhanced green channel, and then enhanced by fusing the reference red/blue components with the spatially demosaicked red/blue sample using the estimated motion vectors. The details of the procedure are provided below.

18.4.3.1 Temporal Enhancement of the Green Channel

Estimating the motion in the demosaicked green channel allows for a matched block of the current block $\boldsymbol{M}_0$ in each reference frame. For K reference frames, $\{\boldsymbol{M}_i\}_{i=1,2,\ldots,K}$ denotes the set of the K matched blocks. Assuming that $\widehat{G}_0$ is a spatially demosaicked green sample in $\boldsymbol{M}_0$ and that G denotes the unknown true green sample corresponding to $\widehat{G}_0$ and $\widehat{G}_i$ denotes the associated matched sample in $\boldsymbol{M}_i$, the following can be written:

$$\widehat{G}_i = G + e_i, \quad i = 0, 1, \ldots, K \tag{18.22}$$

where e_i is the processing errors of $\widehat{G}_i$ in the spatial demosaicking process, nearly uncorrelated with G. Since $\widehat{G}_i$ are independent observations of G in different frames, it is reasonable to assume that e_i, for $i = 0, 1, \ldots, K$, are mutually nearly uncorrelated. Indeed, if e_i are significantly correlated, then the observations $\widehat{G}_i$ are very similar and the reference frames cannot offer a significant amount of new information. Therefore, in this case temporal enhancement cannot improve spatial demosaicking.

A more robust estimate of G can be obtained by fusing all the measurements $\widehat{G}_i$ as follows:

$$\overline{G} = \sum_{i=0}^{K} w_i \widehat{G}_i \tag{18.23}$$

where weights $\sum_{i=0}^{K} w_i = 1$. The criterion of determining w_i is to minimize the MSE $\{w_i\} = \arg\min_{\sum_i w_i = 1} E[(\overline{G} - G)^2]$ of $\overline{G}$, where E is the expectation operator. In one possible implementation, E can be expressed as follows:

$$\Omega = E[(\overline{G} - G)^2] = E\left[\left(\sum_{i=1}^{K} w_i e_i + (1 - \sum_{i=1}^{K} w_i) e_0\right)^2\right] \tag{18.24}$$

Differentiating Ω with respect to e_i, $i = 1, 2, \ldots, K$, and setting the partial derivatives to zero results for $E[e_i e_j]|_{i \neq j} \approx 0$ to the following expression:

$$\frac{\partial \Omega}{\partial w_i} = w_i \sigma_i^2 - \left(1 - \sum_{j=1}^{K} w_j\right) \sigma_0^2 = 0 \tag{18.25}$$

where σ_i^2 denotes the variance of error e_i. The optimal weights for the estimates in the K reference frames constitute the column vector $\boldsymbol{w} = \{w_1, w_2, \ldots, w_K\}$ which can be obtained as:

$$\boldsymbol{w} = \boldsymbol{S}^{-1}\mathbf{1} \tag{18.26}$$

where $\mathbf{1}$ is a column vector whose elements are all ones and $\boldsymbol{S}$ is the $K \times K$ matrix defined as follows:

$$\boldsymbol{S} = \begin{bmatrix} 1+\sigma_1^2/\sigma_0^2 & 1 & \cdots & 1 \\ 1 & 1+\sigma_2^2/\sigma_0^2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1+\sigma_K^2/\sigma_0^2 \end{bmatrix} \tag{18.27}$$

Solving Equation 18.26 for w_i, the fused estimate of G is then computed via Equation 18.23.

18.4.3.2 Estimation of Error Variances

To implement the above algorithm, the error variances σ_i^2 need to be estimated. Since $\widehat{G}_i = G + e_i$ and $E[e_i e_j]|_{i \neq j} \approx 0$, the following can be written:

$$d_{i,j} = E[(\widehat{G}_i - \widehat{G}_j)^2] = \sigma_i^2 + \sigma_j^2, \quad i, j = 0, 1, \ldots, K \text{ and } i \neq j \tag{18.28}$$

The values of $d_{i,j}$ can be estimated adaptively from blocks $\boldsymbol{M}_i$ and $\boldsymbol{M}_j$, for $i \neq j$, as follows:

$$d_{i,j} = \frac{1}{L} \sum_{\widehat{G}_i \in \boldsymbol{M}_i, \widehat{G}_j \in \boldsymbol{M}_j} (\widehat{G}_i - \widehat{G}_j)^2 \tag{18.29}$$

where L is the total number of missing green samples in blocks $\boldsymbol{M}_i$.

If the variance σ_0^2 of e_0 in the current block $\boldsymbol{M}_0$ is known, then the values of σ_i^2 for other values of i can be calculated using Equations 18.28 and 18.29. Otherwise, the σ_i^2 terms can be estimated using the procedure described below.

Let

$$\boldsymbol{\sigma} = \{\sigma_0^2, \ldots, \sigma_K^2\} \tag{18.30}$$

be a $K+1$-dimensional column vector that consists of all the σ_i^2 terms, and let

$$\boldsymbol{d} = \{d_{0,1}, \ldots, d_{0,K}, d_{1,2}, \ldots, d_{1,K}, \ldots, d_{K-1,K}\} \tag{18.31}$$

be a $K(K+1)/2$-dimensional column vector that is comprised of all the $d_{i,j}$ terms. Then, there exists a $K(K+1)/2 \times (K+1)$ matrix $\boldsymbol{H}$ such that

$$\boldsymbol{d} = \boldsymbol{H}\boldsymbol{\sigma} \tag{18.32}$$

where $\boldsymbol{h}_{i,j}$ is the row in $\boldsymbol{H}$ such that $d_{i,j} = \boldsymbol{h}_{i,j}\boldsymbol{\sigma}$. Since only the i^{th} and j^{th} elements in $\boldsymbol{h}_{i,j}$ are 1 and all other elements are zeros, $\boldsymbol{\sigma}$ can be estimated by the least square estimation technique as follows:

$$\boldsymbol{\sigma} = (\boldsymbol{H}'\boldsymbol{H})^{-1}\boldsymbol{H}'\boldsymbol{d} \tag{18.33}$$

(a) (b)

FIGURE 18.6

Selected frames from the image sequences used in the experimentation: (a) test color sequence, and (b) real-life mosaic sequence.

18.4.3.3 Reconstruction of Red and Blue Channels

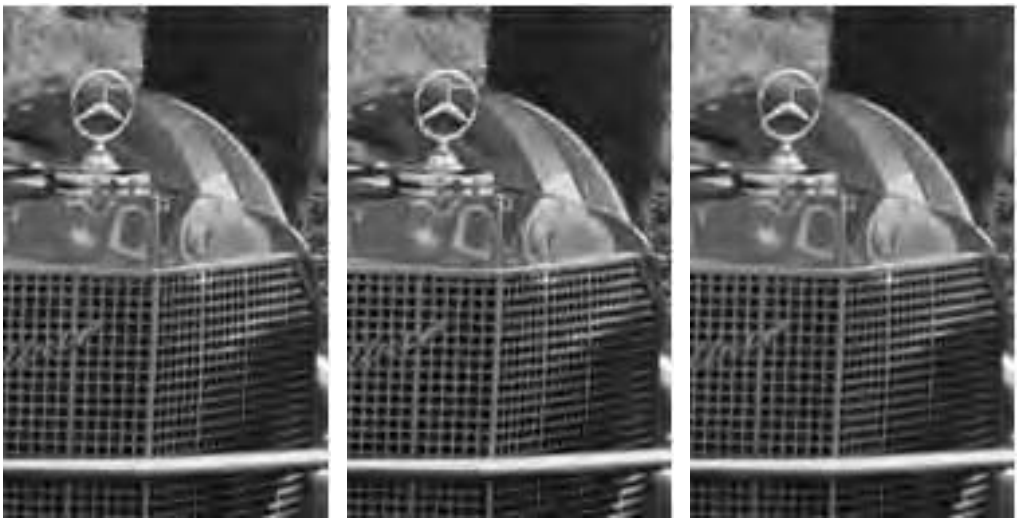
After the green channel has been spatially restored and then temporally enhanced in each frame of the captured video, the video-demosaicking process continues with spatially restoring the red and blue channels in each green demosaicked frame. Any spatial demosaicking algorithm can be used to complete this task. However, reasonable results can be obtained using a procedure described in Section 18.3.4 with temporally enhanced green values by the procedure described in Section 18.4.3.1.

Restored red and blue channels can be enhanced via motion estimation and data fusion by following the procedure described in Section 18.4.3.1. Since green components are usually less affected by the demosaicking error due to their higher amount in the CFA data captured using Bayer CFA, motion vectors should be estimated on the green channel instead of red and blue channels.

18.5 Examples

The spatial method described in Section 18.3 and the spatiotemporal method presented in Section 18.4 were evaluated in terms of the demosaicking performance. In the first experiment, a color video sequence, originally captured on film and then digitized by high-resolution scanner at a 24 frames per second rate with the spatial resolution of 1948×1280 pixels was subsampled using the Bayer CFA to generate the mosaic data. Figure 18.6a shows one frame from this test video sequence. The second experiment was conducted using real (non-simulated) single-sensor video sequence recorded at the rate of 25 frames per second and spatial resolution of 640×480 pixels. An example frame is shown in Figure 18.6b. In both experiments, spatiotemporal demosaicking was performed using the information from five consecutive frames.

Figure 18.7a shows a complex edge area cropped from the original frame in Figure 18.6a. The result shown in Figure 18.7b was obtained via spatial LMMSE demosaicking [9]. In this demosaicked image, the grill area of the car suffers from significant color artifacts. In



(a) (b) (c)

FIGURE 18.7 (See color insert.)

Experimentation using test image sequence from Figure 18.6a: (a) original image, (b) image demosaicked using the spatial method described in Section 18.3, and (c) image demosaicked using the spatiotemporal method presented in Section 18.4.

(a) (b) (c)

FIGURE 18.8 (See color insert.)

Experimentation using real single-sensor data: (a) original mosaic image, (b) image demosaicked using the spatial method described in Section 18.3, and (c) image demosaicked using the spatiotemporal method presented in Section 18.4.

addition, strong zipper effects can be seen along the boundary of the red and silver colors and on the emblem of the car. It was observed that spatial LMMSE demosaicking usually fails in the areas with sharp edges of highly saturated colors where both spatial and spectral correlations are weak, and the color difference signal exhibits abrupt changes. As shown in Figure 18.7c, spatiotemporal demosaicking can overcome the problem and produce images of higher visual quality. Due to the use of temporal information, spatiotemporal LMMSE demosaicking can significantly reduce the amount of color artifacts and nicely reconstruct difficult edge areas, even those with sharp edges and fine details that are blurred or distorted by spatial demosaicking.

In the second clip, the camera is still but the man is moving with a colorful bag in hand. Figure 18.8a shows a zoom-in portion of the mosaic image, where sharp edges of saturated

colors exist. Figure 18.8b is the result by the spatial demosaicking method. We can see many artifacts associated with sharp edges. Figure 18.8c is the result by the spatiotemporal demosaicking method. It yields much better visual quality than the spatial demosaicking technique.

18.6 Conclusion

Video-demosaicking of spatiotemporal mosaic data captured by single-sensor digital camera was the topic of this chapter. Three basic paradigms — temporal, spatial, and spatiotemporal demosaicking — were identified based on the dimensionality of the area of support in the video-demosaicking process. Among these, spatiotemporal solutions are most suitable for the task at hand due to their ability to utilize temporal, spatial-spectral and even motion characteristics when estimating and compensating the motion during video-demosaicking.

As an example of motion compensated approach to spatiotemporal demosaicking, a video-demosaicking scheme with temporal enhancement of spatially demosaicked samples was presented. Spatially demosaicked estimates were fused and temporally enhanced using a motion-compensated linear minimum mean square error estimation (LMMSE) framework. In this way, visually pleasing video sequences can be produced, as shown in the experiments on both simulated and real mosaic video data. Spatiotemporal demosaicking requires a fairly large buffer to hold multiple reference frames, and involves quite extensive computations compared with the spatial demosaicking solutions. However, the complexity can be reduced by invoking spatiotemporal demosaicking judiciously to the areas with complex edges and fine details while restoring smooth regions via spatial demosaicking.

Video-demosaicking constitutes a promising research and application field, as video-capturing capabilities have become an important feature in various consumer electronic products such as camera phones, webcams, video-conference equipment, and digital cinema. As demonstrated in this chapter, even sophisticated spatial demosaicking techniques may not be able to produce the demosaicked video of high visual quality. Fortunately, spatiotemporal demosaicking combined with motion compensation can overcome this drawback. Therefore, it is believed that advanced spatiotemporal video-demosaicking constitutes an attractive solution for modern single-sensor imaging devices and applications.

Acknowledgment

Figure 18.4 and Figure 18.5 are reprinted from Reference [20] with the permission of IEEE.

References

[1] B.E. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 1976.

[2] R. Lukac and K.N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1260–1267, November 2005.

[3] R. Lukac and K.N. Plataniotis, *Color Image Processing: Methods and Applications*, ch. Single-sensor camera image processing, R. Lukac and K.N. Plataniotis (eds.), Boca Raton, FL: CRC Press / Taylor & Francis, 2006, pp. 363–392.

[4] B.K. Gunturk, J. Glotzbach, Y. Altunbasak, R.W. Schafer, and R.M. Mersereau, "Demosaicking: Color filter array interpolation in single-chip digital cameras," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 44–54, January 2005.

[5] J.E. Adams, "Design of practical color filter array interpolation algorithms for digital cameras," *Proceedings of the SPIE*, vol. 3028, pp. 117–125, April 1997.

[6] K. Hirakawa and T.W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 360–369, March 2005.

[7] X. Li, "Demosaicing by successive approximation," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 370–379, March 2005.

[8] R. Lukac and K.N. Plataniotis, "Normalized color-ratio modeling for CFA interpolation," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 737–745, May 2004.

[9] L. Zhang and X. Wu, "Color demosaicking via directional linear minimum mean square-error estimation," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2167–2178, December 2005.

[10] S. Farsiu, M. Elad, and P. Milanfar, "Multiframe demosaicing and super-resolution of color images," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 141–159, January 2006.

[11] T. Gotoh and M. Okutomi, "Direct super-resolution and registration using raw CFA images," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Washington, D.C., USA, June / July 2004, vol. 2, pp. 600–607.

[12] R. Lukac and K.N. Plataniotis, "Fast video demosaicking solution for mobile phone imaging applications," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 675–681, May 2005.

[13] R. Lukac and K.N. Plataniotis, "Adaptive spatiotemporal video demosaicking using bidirectional multistage spectral filters," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 651–654, May 2006.

[14] R.R. Schultz, L. Meng, and R.L. Stevenson, "Subpixel motion estimation for super-resolution image sequence enhancement," *Journal of Visual Communication and Image Representation*, vol. 9, no. 1, pp. 38–50, March 1998.

[15] C. Stiller and J. Konrad, "Estimating motion in image sequence," *IEEE Signal Processing Magazine*, vol. 16, no. 4, pp. 70–91, July 1999.

[16] B.C. Tom and A.K. Katsaggelos, "Resolution enhancement of monochrome and color video using motion compensation," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 278–287, February 2001.

[17] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: A review and a new contribution," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 858–876, June 1995.

[18] P. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*. Boston, MA: Kluwer Academic Publishers, 1999.

[19] X. Wu and L. Zhang, "Color video demosaicking via motion estimation and data fusion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 231–240, February 2006.

[20] X. Wu and L. Zhang, "Improvement of color video demosaicking in temporal domain," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 3138–3151, October 2006.

19

Simultaneous Demosaicking and Resolution Enhancement from Under-Sampled Image Sequences

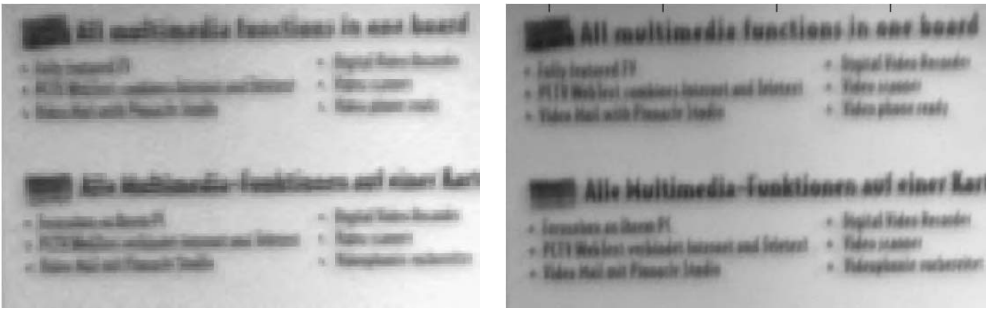
Sina Farsiu, Dirk Robinson, Michael Elad, and Peyman Milanfar

19.1 Introduction

Single-sensor imaging provides a convenient, yet low-cost solution for the problem of capturing color images at multiple wavelengths. To capture color, single-sensor cameras add a color filter array (CFA) [1], [2] on top of the photo-detector array. The CFA provides a simple mechanism for sampling different color channels in a multiplexed fashion.[1] Using a CFA trades-off the spatial sampling resolution of the three-sensor system, to achieve multispectral sampling. Refer to Chapters 1 and 5 for details.

Visual inspection of a real image shown in Figure 19.1a reveals various image quality issues encountered in single-sensor imaging. These shortcomings motivated the development of image processing algorithms to improve image quality for single-sensor imaging

[1] Design, properties, and performance characteristics of many CFA patterns are discussed in Reference [2].



(a) (b)

FIGURE 19.1 (See color insert.)

(a) Example showing the typical image quality problems associated with inexpensive single-sensor imaging devices. The image is noisy with evident color artifacts and has limited overall resolution as evidenced by the difficulty in reading the text numbers. (b) Image enhanced using a multiframe color supperresolution approach.

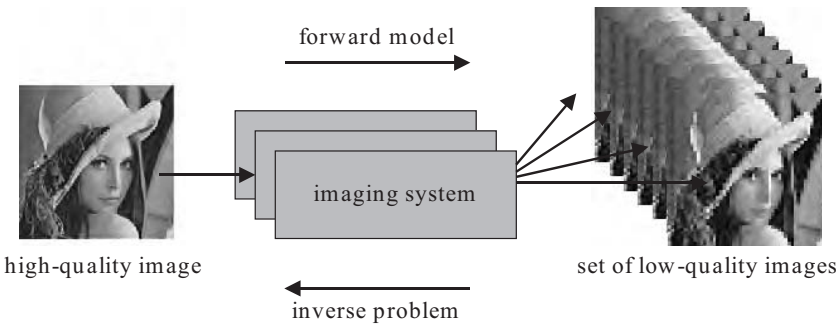


FIGURE 19.2

A block diagram representation of the multiframe image processing. The forward model is a mathematical description of the image degradation process. The multiframe inverse problem addresses the issue of retrieving (or estimating) the original scene from a *set* of low-quality images.

systems. Arguably, the most powerful image processing methods are the multiframe image processing approaches. The multiframe approaches combine the information from multiple images to produce higher quality images (Figure 19.1b). Figure 19.2 provides a general picture of the multiframe imaging approach to mitigate the shortcomings of imaging systems.

The need to interpolate the missing values in CFA images calls for a model that ties the full-color images in the sequence to the measurements obtained. The mathematical model of the forward imaging model for single-sensor imaging is

$$\mathbf{y}(k) = \mathbf{ADHF}(k)\mathbf{x} + \mathbf{v}(k) \tag{19.1}$$

where $\mathbf{y}$ is a vector containing all of the captured pixel values for a single image. The vector $\mathbf{x}$ contains the pixel values of the unknown high-resolution, noise-free image. Conceptually,

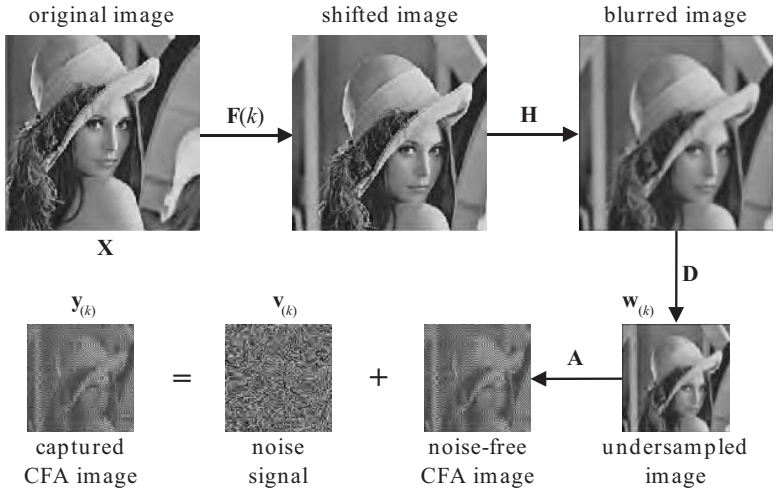


FIGURE 19.3

Block diagram representing the image formation model considered in this chapter, where **x** is the perfect intensity image of the scene, **v** is the additive noise, and **y** is the resulting color-filtered low-quality image. The operators **F**, **H**, **D**, and **A** are representatives of the warping, blurring, downsampling, and color-filtering processes, respectively.

we can consider this vector to be comprised of the three unknown color channel images

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_R \\ \mathbf{x}_G \\ \mathbf{x}_B \end{pmatrix}. \tag{19.2}$$

The matrix **F** represents a motion operator capturing the image motion induced by the relative motion between the imaging system and the camera. The matrix **H** captures the blurring effects due to the camera optics, and **D** represents the downsampling effect due to limited resolution of the sensor array. These three operators (**F**, **H**, and **D**) are applied to each of the color channels, either separately or jointly. In this work we assume that their effect is separated to each color channel.

The matrix **A** represents the sampling effect due to the CFA. This again stands for an operator that operates on the three color channels, sampling each differently, based on the Bayer pattern. The vectors **v** represent the additive random noise inherent in the imaging process. For the multiframe imaging case, the variable k indexes the N captured frames. Notice that in our proposed model, only the motion operators are assumed to vary as a function of k, implying that the blur, downsampling, and CFA sampling are all independent of time. An extension of this model could be proposed, where the various operators vary in time, and operate on the RGB in a mixed way.

Figure 19.3 depicts the image capture model for multiframe image processing. The goal of multiframe processing is to estimate the high resolution image **x** from a collection of low-resolution images $\{\mathbf{y}(k)\}$. In general, the problem of multiframe processing for single-sensor imaging systems is complicated. Earlier approaches to this problem sequentially solved single frame demosaicking, followed by multiframe resolution enhancement or su-

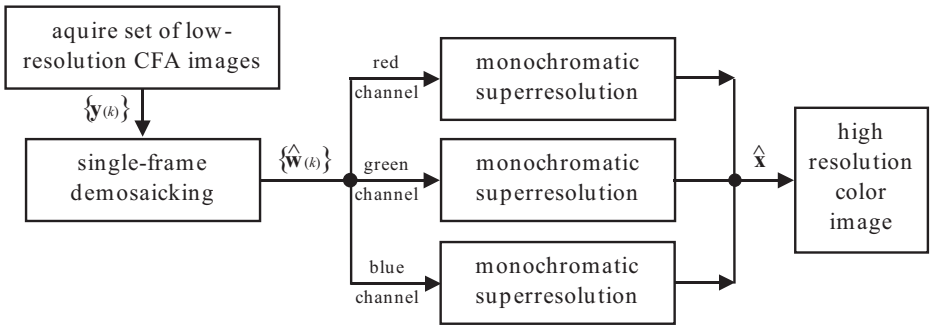


FIGURE 19.4

Block diagram representing the traditional two-step approach to the multiframe reconstruction of color images. First, a demosaicking algorithm is applied to each of the CFA images independently. Second, a superresolution algorithm is applied to each of the resulting color channel image sets independently.

perresolution. The proposed approach, a multiframe demosaicking scheme, solves both problems in a joint fashion offering improved performance. In Sections 19.2 and 19.3 we describe both of these approaches in more detail. In Section 19.4, we describe a fast implementation of the joint approach enabling multiframe enhancement of single-sensor video sequences. Concluding remarks of this chapter are given in Section 19.5.

19.2 Sequential Demosaicking and Superresolution

The classic approach for multiframe processing of single-sensor color images involves two stages of processing. First, single frame demosaicking interpolates the missing color pixel values in the mosaic data. This single frame demosaicking is applied to a collection of images independently, producing a set of full-color low-resolution images. Second, these restored low-resolution color images are processed using multiframe resolution enhancement or *superresolution* algorithms. The superresolution algorithms are either applied to each color channel independently [3], [4], or to the luminance (grayscale) components of the color images [5]. This traditional two-step approach is illustrated in Figure 19.4.

Producing a high resolution image requires a smart combination of data processing and prior knowledge on the spatial and spectral (color) properties of images. The prior information about the images characterizes both the color and the spatial properties of images. In the following sections, we review the various approaches to single-frame demosaicking and superresolution, including analysis of the different forms of prior information needed to solve these two problems.

19.2.1 Single-Frame Demosaicking

Single-frame demosaicking algorithms, or just demosaicking algorithms, attempt to restore the missing color components eliminated by using a color filter array. Essentially,

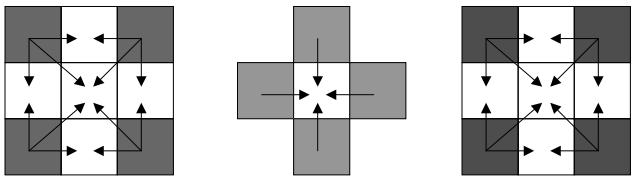


FIGURE 19.5

Illustration of the basic interpolation problem in single-frame demosaicking. The demosaicking algorithm must estimate the missing color pixel value using the observed neighboring values. Individual pixel arrangements correspond to: (left) blue, (middle) green, and (right) red channel of Bayer CFA captured images.

demosaicking algorithms try to estimate a low-resolution full-color image $\mathbf{w}(k)$ defined as

$$\mathbf{w}(k) = \mathbf{DHF}(k)\mathbf{x}. \tag{19.3}$$

The full-color image $\mathbf{w}(k)$ has three color values for every pixel location, whereas the captured image $\mathbf{y}(k) = \mathbf{A}\mathbf{w}(k) + \mathbf{v}(k)$ contains only one color value per pixel location. The single-frame demosaicking algorithm is applied to each captured image $\mathbf{y}(k)$ independently.

Numerous demosaicking methods have been proposed over the years to solve this underdetermined problem and in this section we review some of the more popular approaches. The simplest approach estimates the unknown pixel values using linear interpolation of the known color values in the neighboring pixel locations. Figure 19.5 illustrates the standard neighborhood interpolation problem of single-frame demosaicking.

The straightforward interpolation approach ignores some important information about the correlation between the color channel images and often produces serious color artifacts. It can be shown that the red and blue channels are related to the green channel via the approximate equality of the ratios $\frac{red}{green}$ and $\frac{blue}{green}$. As observed in Figure 19.5, the green channel has twice the number of pixels, compared to the red and blue channels. Thus, using the correlations between the color channels just described allows us to provide better interpolation of the red and blue channels using the more reliably interpolated green channel. This approach forms the basis of the smooth-hue transition method first discussed in Reference [6].

Note that this correlation between color channels does not hold across edges in an image. Consequently, although the smooth-hue transition algorithm works for smooth regions of the image, it does not work in the high-frequency (edge) areas. Considering this fact, gradient-based methods, first addressed in Reference [7], do not perform interpolation across the edges of an image. This noniterative method [7] uses the second derivative of the red and blue channels to estimate the edge direction in the green channel. Later, the green channel is used to compute the missing values in the red and blue channels.

A variation of this method was later proposed in Reference [8], where the second derivative of the green channel and the first derivative of the red (or blue) channels are used to estimate the edge direction in the green channel. Both the smooth hue and edge-based methods were later combined in Reference [9]. In this iterative method, the smooth hue interpolation is performed along the local image gradients computed in eight directions about

a pixel of interest. A second stage using anisotropic inverse diffusion further enhances the quality of the reconstructed image. This two step approach of interpolation followed by an enhancement step has been used in many other settings. In Reference [10], spatial and spectral correlations among neighboring pixels are exploited to define the interpolation step, while adaptive median filtering is used as the enhancement step. A different iterative implementation of the median filter is used as the enhancement step of the method described in Reference [11], that takes advantage of a homogeneity assumption in the neighboring pixels.

Maximum a posteriori (MAP) methods form another important category of demosaicking methods. These MAP approaches apply global assumptions about the correlations between the color channels and the spatial correlation using a penalty function of the form

$$\Omega(\mathbf{w}) = J_0(\mathbf{w},\mathbf{y}) + P(\mathbf{w}) \tag{19.4}$$

where $J_0(\cdot,\cdot)$ captures the correspondence between the estimated image $\mathbf{w}$ and the observed data $\mathbf{y}$ and $P(\cdot)$ captures the quality of the proposed solution $\mathbf{w}$ based on the prior knowledge about the spatial and color correlations in the destination image. For example, a MAP algorithm with a smooth chrominance prior is discussed in Reference [12]. The smooth chrominance prior is also used in Reference [13], where the original image is first transformed to YIQ representation.[2] The chrominance interpolation is performed using isotropic smoothing. The luminance interpolation is done using edge directions computed in a steerable wavelet pyramidal structure.

Other examples of popular demosaicking methods available in published literature are [15], [16], [17], [18], [19], [20], [21], and [22]. For additional information on demosaicking refer to Chapters 1, 3, and 6 to 9. We also note that a few postdemosaicking image enhancement techniques have been proposed (e.g., References [23] and [24]) to reduce artifacts introduced through the color-interpolation process. Almost all of the above demosaicking methods are based on one or more of these following assumptions:

1. In the measured image with the mosaic pattern, there are more green sensors with regular pattern of distribution than blue or red ones (in the case of Bayer CFA there are twice as many greens than Red or Blue pixels and each is surrounded by four green pixels).
2. The CFA pattern is most likely the Bayer pattern.
3. For each pixel, one and only one color band value is available.
4. The color pattern of available pixels does not change through the measured image.
5. The human eye is more sensitive to the details in the luminance component of the image than the details in chrominance component [13].
6. The human eye is more sensitive to chromatic changes in the low-spatial frequency regions than the luminance changes [18].
7. Interpolation should be performed along and not across the edges.
8. Different color bands are correlated with each other.
9. Edges should align between color channels.

[2] YIQ is the standard color representation used in broadcast television (NTSC systems) [14].

FIGURE 19.6

A high-resolution image (a) captured by a three-CCD sensor camera is (b) downsampled by a factor of four. In (c) the original three-sensor image is blurred by a Gaussian kernel before downsampling by a factor of four. The images in (a-c) are color-filtered and then demosaicked by the method of Reference [9] to produce the results shown in (d-f), respectively. © 2007 IEEE

The accuracy of these single-frame demosaicking algorithms depends to a large degree on the resolution of the single-sensor imaging system. For example, Figure 19.6a shows a high-resolution image captured by a three-sensor camera. If we had captured this image instead using a single-sensor camera and then applied the single frame demosaicking algorithm of Reference [9], we would obtain the image depicted in Figure 19.6d. This demosaicked image shows negligible color artifacts. Figure 19.6b shows the same scene from a simulated three-sensor camera which is undersampled by a factor of four. Figure 19.6e shows the single-frame demosaicking result, where the color artifacts in this image are

much more evident than Figure 19.6d. In this case, inadequate sampling resolution and the subsequent aliasing produces severe color artifacts. Such severe aliasing happens in cheap commercial still or video digital cameras, with small number of sensor pixels [25].

Some imaging systems attempt to avoid these aliasing artifacts by adding optical low-pass filters to eliminate high spatial frequency image content prior to sampling. Refer to Chapter 4 for details. Such optical low-pass filters not only increase system cost but lower image contrast and sharpness. These optical low-pass filters can remove some, but not all color artifacts from the demosaicking process. Figure 19.6c shows a simulated low-resolution image which was presmoothed by a Gaussian low-pass digital filter simulating the effects of an optical low-pass filter prior to downsampling. The demosaicked version of this captured image is shown in Figure 19.6f. The demosaicked image contains fewer color artifacts than Figure 19.6e; however, it has lost some high-frequency details.

The residual aliasing artifacts amplified by the demosaicking algorithm motivated the application of multiframe resolution enhancement algorithms. We describe these algorithms in the next section.

19.2.2 Multiframe Superresolution

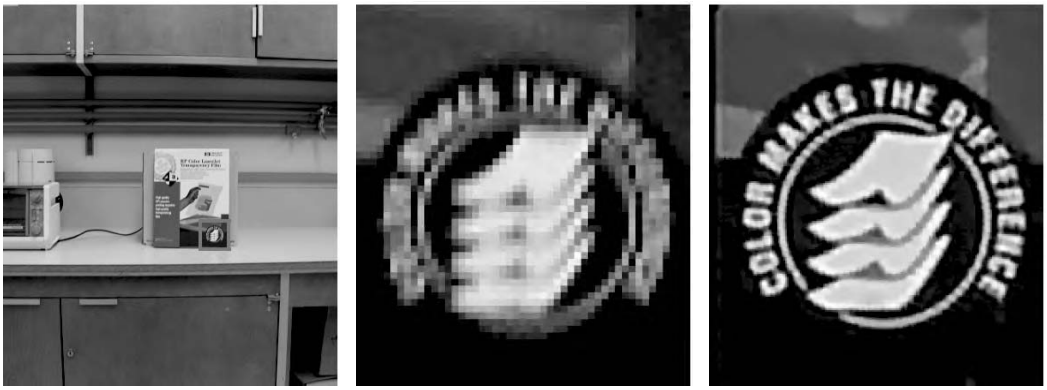
A relatively recent approach to dealing with aliasing in imaging systems is to combine multiple aliased images with some phase modulation between them to reconstruct a single high resolution image free of (or with significantly less) aliasing artifacts with improved SNR.[3] Such methods have been studied extensively for monochromatic imaging [26], [27], [28] and are commonly known as multiframe resolution enhancement (superresolution) algorithms, which produce sharp images with less noise and higher *spatial resolution* than the capture images.

The standard model for such algorithms assumes that a single monochromatic low-resolution aliased image in a set of N images is defined by

$$\mathbf{m}(k) = \mathbf{DHF}(k)\mathbf{g} + \mathbf{v}(k), k = 0 \ldots N-1 \tag{19.5}$$

where $\mathbf{m}(k)$ is the kth captured monochrome image. In this case, $\mathbf{g}$ represents the unknown high resolution monochromatic image. In the context of the sequential multiframe demosaicking approach, the captured images $\mathbf{m}(k)$ are obtained by extracting one of the color channels from the demosaicked images. In other words, $\mathbf{m}(k)$ is either $\hat{\mathbf{w}}_R(k)$ or $\hat{\mathbf{w}}_G(k)$ or $\hat{\mathbf{w}}_B(k)$ and $\mathbf{g}$ is either $\mathbf{x}_R$, or $\mathbf{x}_G$, or $\mathbf{x}_B$. As before, $\mathbf{F}(k)$ is a warping operator capturing the relative motion between the kth frame and the anchor frame $k = 0$ and $\mathbf{v}(k)$ represents the additive noise in the image system. The matrix $\mathbf{H}$ represents the blurring associated with the optical point spread function and $\mathbf{D}$ represents the downsampling operator reflecting the limited spatial resolution associated with the image sensor. In effect, the downsampling operator is the cause of the aliasing artifacts observed in the final image. While we do not address the motion estimation in full detail in this chapter, an appendix provides a brief discussion about this topic, with an emphasis on the need to estimate the cross-motion between the N frames jointly to avoid accumulated error.

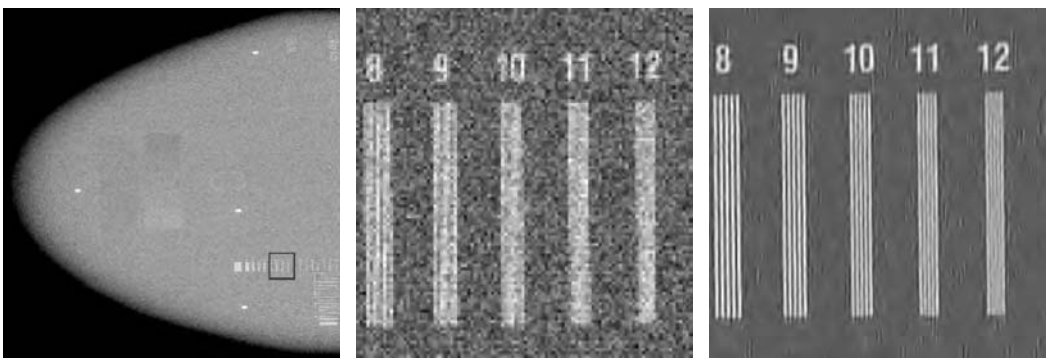
[3] Signal to noise ratio (SNR) is defined as $10\log_{10}\frac{\sigma^2}{\sigma_n^2}$, where σ^2, σ_n^2 are variance of a clean frame and noise, respectively.



(a) (b) (c)

FIGURE 19.7

Superresolution experiment on real image data. Twenty-six low quality images were combined to produce a higher quality image. One captured image is shown in (a). The red square section of (a) is zoomed in (b). Superresolved image in (c) is the high quality output image.



(a) (b) (c)

FIGURE 19.8

Superresolution experiment on real image data. Seventeen low quality images from a digital X-ray machine were combined to produce a higher quality image. One captured image is shown in (a). The red square section of (a) is zoomed in (b). Superresolved image in (c) is the high quality output image.

Early studies of superresolution showed that the aliasing artifacts in the low-resolution images enable the recovery of the high-resolution fused image, provided that a relative sub-pixel motion exists between the under-sampled input images [29].

Figure 19.7 shows an example of the resulting image after applying superresolution processing to a collection of images taken by a commercial digital still camera. The application of superresolution is not restricted to optical imaging. An example is illustrated in Figure 19.8, where seventeen images captured by a Siemens digital X-ray imaging system were fused to create the shown high resolution X-ray image [30]. Figure 19.8a shows one of the input images, a selected region of which is zoomed in Figure 19.8b for a closer examination. A factor of three superresolved image of Figure 19.8a is zoomed in Figure 19.8c, showing a significant reduction in aliasing artifacts.

The multiframe superresolution problem was first addressed in Reference [29], where the authors proposed a frequency domain approach, later extended by others [31]. Although the frequency domain methods are computationally cheap, they are very sensitive to noise and modelling errors [26]. Also, by design, these approaches handle only pure translational motion.

Another popular class of methods solves the problem of resolution enhancement in the spatial domain. The noniterative spatial domain data fusion approaches are studied in References [32], [33], and [34]. The iterative methods are based on construction of a cost function, which is minimized in an iterative function minimization approach. The cost function is of the form

$$\Omega(\mathbf{g}) = J_0(\mathbf{g}, \{\mathbf{m}(k)\}) + P(\mathbf{g}) \tag{19.6}$$

where $J_0(\mathbf{g}, \{\mathbf{m}(k)\})$ is the data penalty term that measures the closeness of the observed data $\{\mathbf{m}(k)\}$ to the predicted measurements using an estimate of the image $\mathbf{g}$. The $P(\mathbf{g})$ term represents the prior information about the unknown signal $\mathbf{g}$, as described before.

The most common data fidelity term is based on minimizing the L_2 difference between the observed data and the predicted data for an estimated image $\mathbf{g}$. The cost function looks like

$$J_0(\mathbf{g}, \{\mathbf{m}(k)\}) = \sum_{k=0}^{N-1} \|\mathbf{DHF}(k)\mathbf{g} - \mathbf{m}(k)\|_2^2. \tag{19.7}$$

The solution to this penalty function is equivalent to the least squares estimate of the image $\mathbf{g}$, and is the optimal maximum likelihood (ML) solution [35] when noise follows the white Gaussian model, and when the motion vectors implicit in $\mathbf{F}(k)$ are assumed to be known exactly, and a priori.

One distinct advantage of the spatial-domain approaches is the ability to incorporate complicated prior information into the estimation process. The most common form of prior information arises as a type of *regularization* to the commonly ill-conditioned or ill-posed estimation problem. This regularization or prior information takes the form of some assumptions about the spatial variability of the unknown high resolution image $\mathbf{g}$. Many of the early superresolution algorithms, however, applied rudimentary regularization such as Tikhonov regularization [36], which describes the smoothness of the image $\mathbf{g}$. A Tikhonov regularization has the form

$$P(\mathbf{g}) = \lambda \|\Lambda \mathbf{g}\|_2^2 \tag{19.8}$$

where Λ represents a spatial high-pass operator applied to the image $\mathbf{g}$. In other words, the estimated image will have a reduced high spatial frequency content, depending on the magnitude of the weighting parameter λ.

The early iterative algorithms used search techniques to minimize cost functions. The simplest approach to minimize a cost function of the form of Equation 19.6 is to use the steepest descent (SD) algorithm. In SD, the derivative of Equation 19.6 is computed with respect to the unknown parameter vector $\mathbf{g}$ to update the estimate. The estimate for the $n+1$th iteration is updated according to

$$\widehat{\mathbf{g}}^{n+1} = \widehat{\mathbf{g}}^{n} - \beta \nabla_x \Omega(\widehat{\mathbf{g}}^{n}) \tag{19.9}$$

where β is the SD step size. This approach is similar to the iterative back-projection method proposed in References [5] and [37].

These spatial domain methods discussed so far were often computationally expensive, especially when solved via explicit construction of the modelling matrices of Equation 19.5. The authors of Reference [38] introduced a block circulant preconditioner for solving the Tikhonov regularized superresolution problem formulated in Reference [36]. In addition, they addressed the calculation of regularization factor for the under-determined case[4] by generalized cross-validation presented in Reference [39].

Later, a very fast superresolution algorithm for pure translational motion and common space invariant blur was developed in Reference [33]. This fast approach was based on the observation that the deblurring can be separated from the merging process of the low-resolution images, and that merging of the images is obtained simply by shift-and-add operation. Using these, it was shown that the result is nevertheless ML optimal. Another interesting observation in this work is the fact that the matrices $\mathbf{H}$, Λ, $\mathbf{D}$, etc., can be applied directly in the image domain using standard operations such as convolution, masking, downsampling, and shifting. Implementing the effects of these matrices as a sequence of operators on the images eliminates the need to explicitly construct the matrices. As we shall see momentarily, this property helps us develop fast and memory efficient algorithms.

A different fast spatial domain method was recently suggested in Reference [40], where low-resolution images are registered with respect to a reference frame defining a nonuniformly spaced high-resolution grid. Then, an interpolation method called Delaunay triangulation is used for creating a noisy and blurry high-resolution image, which is subsequently deblurred. All of the above methods assumed the additive Gaussian noise model. Furthermore, regularization was either not implemented or it was limited to Tikhonov regularization. Another class of superresolution techniques uses the projection onto convex sets (POCS) formulation [41], [42], [43]. The POCS approach allows complicated forms of prior information such as set inclusions to be invoked.

Recently, a powerful class of superresolution algorithms leverage sophisticated *learned*-based prior information. In these algorithms, the prior penalty functions $P(\mathbf{g})$ are derived from collections of training image samples [44], [45]. For example, in Reference [45] an explicit relationship between low-resolution images of faces and their known high-resolution image is learned from a face database. This learned information is later used in reconstructing face images from low-resolution images. Due to the need for gathering a vast number of examples, often these methods are only effective when applied to very specific scenarios, such as faces or text.

Another class of superresolution algorithms focuses on being robust to modelling errors and different noise. These methods implicitly [46] or explicitly [47], [48] take advantage of L_1 distance metrics when constructing the cost functions. Compared to Equation 19.7, the methods in References [47] and [48] apply the L_1 metric as the data fidelity term

$$J_0(\mathbf{g}, \{\mathbf{m}(k)\}) = \sum_{k=0}^{N-1} \|\mathbf{DHF}(k)\mathbf{g} - \mathbf{m}(k)\|_1, \tag{19.10}$$

[4] In under-determined case, the number of nonredundant low-resolution frames is smaller than the square of resolution enhancement factor. A resolution enhancement factor of r means that low-resolution images of dimension $Q_1 \times Q_2$ produce a high-resolution output of dimension $rQ_1 \times rQ_2$. Scalars Q_1 and Q_2 are the number of pixels in the vertical and horizontal axes of the low-resolution images, respectively.

(a) (b) (c)

FIGURE 19.9

The image (a) shows an example low-resolution CFA image processed by the single frame demosaicking algorithm of Reference [9]. The image (b) shows the resulting color image after applying the robust superresolution algorithm [47] to each of the color channels independently completing the sequential multiframe processing algorithm, which shows improved resolution with significantly reduced artifacts. Some residual color artifacts, however, are apparent along the fence. Better reconstruction results are shown in (c), where the multiframe demosaicking method presented in Section 19.3 has fused the information of ten CFA frames. © 2007 IEEE

which is the optimal ML solution in the presence of additive white Laplacian noise.

In Reference [47], in the spirit of the total variation criterion [49], [50] and a related method called the bilateral filter [51], [52], the L_1 distance measure was also incorporated into a prior term called the bilateral total variation (BTV) regularization penalty function, defined as

$$P(\mathbf{g}) = \lambda \sum_{l=-L_{max}}^{L_{max}} \sum_{m=-M_{max}}^{M_{max}} \alpha^{|m|+|l|} \|\mathbf{g} - \mathbf{S}_x^l \mathbf{S}_y^m \mathbf{g}\|_1. \tag{19.11}$$

where S_x^l and S_y^m are the operators corresponding to shifting the image represented by $\mathbf{g}$ by l pixels in the horizontal and m pixels in the vertical directions, respectively. The parameters M_{max} and L_{max} define the size of the corresponding bilateral filter kernel. The scalar weight α, $0 < \alpha \leq 1$, is applied to give a spatially decaying effect to the summation of the regularization terms.

Alternative robust superresolution approaches were presented in References [53] and [54] based on normalized convolution and nonparametric kernel regression techniques. To achieve robustness with respect to errors in motion estimation, Reference [55] proposed a novel solution based on modifying camera hardware. Finally, in References [42], [56], and [57] the superresolution framework has been applied to reduce the quantization noise resulting from video compression. More comprehensive surveys of monochromatic multiframe superresolution methods can be found in References [26], [58], [59], and [60].

19.2.3 Sequential Processing Example

Figure 19.9 shows an example of the sequential multiframe processing. In this example, ten CFA images were synthetically generated using the high resolution image shown in Figure 19.6a. To produce the simulated CFA images, each color channel of the high resolution image was blurred by a 5×5 pixel Gaussian blurring kernel with unit standard deviation. Then, the ten images were taken from this blurry image by subsampling by a factor of four starting at randomly offset pixels. White Gaussian noise was added to each of the simulated low-resolution images such that the image had an SNR of 30dB. Each of the simulated CFA images was processed by the single frame demosaicking algorithm described in Reference [9].

Figure 19.9a shows an example low-resolution image after single frame demosaicking. Then, the robust monochromatic superresolution algorithm of Reference [47] was applied to each of the three color channels independently. The resulting high resolution image is shown in Figure 19.9b, which shows improvement in resolution and noise reduction. The high resolution image still, however, contains some color artifacts and has over-smoothing in some regions. The weakness of these results motivated the use of multiframe demosaicking algorithms, an example of which is shown in Figure 19.9c. These techniques are described in detail in the next section.

19.3 Multiframe Demosaicking

The sequential multiframe processing approach works reasonably well when the amount of aliasing artifacts is small. With significant color aliasing artifacts, however, the sequential approach fails to adequately correct all of the color artifacts. Basically, the failure is traced to the fact that all of the color prior information is applied at the individual (single) frame processing level during demosaicking. A more complicated, yet robust method performs the demosaicking and superresolution *simultaneously*. This approach is known as *multiframe demosaicking* and has been developed relatively recently [61], [62], [63], [64], [65], [66], [67]. Note that some of these methods (e.g., References [63] and [66]), not motivated by superresolution enhancement, were designed primarily for video-demosaicking, which is described in detail in Chapter 18.

19.3.1 Modelling the Problem

The multiframe demosaicking approach tries to estimate the full-color high resolution image $\mathbf{x}$ using the the *set* of unprocessed CFA images $\mathbf{y}(k)$. In this case, the complete forward data model given by Equation 19.1 is used to reconstruct the full-color high resolution image $\mathbf{x}$.

The generic MAP multiframe demosaicking approach estimates $\mathbf{x}$ by minimizing a cost function of the form

$$\Omega(\mathbf{x}) = J_1(\mathbf{x}, \{\mathbf{y}(k)\}) + P_1(\mathbf{x}) + P_2(\mathbf{x}) + P_3(\mathbf{x}). \tag{19.12}$$

This cost function is similar to the superresolution cost function of Equation 19.6 with the exception that now we have three prior information terms. These multiple terms include both the spatial and color prior information constraints. In Reference [65], the MAP-based penalty terms are described as:

1. $J_1(\cdot,\cdot)$: A penalty term to enforce similarities between the raw data and the high-resolution estimate (data fidelity penalty term).
2. $P_1(\cdot)$: A penalty term to encourage sharp edges in the luminance component of the high-resolution image (spatial luminance penalty term).
3. $P_2(\cdot)$: A penalty term to encourage smoothness in the chrominance component of the high-resolution image (spatial chrominance penalty term).
4. $P_3(\cdot)$: A penalty term to encourage homogeneity of the edge location and orientation in different color bands (spectral dependencies penalty term).

These terms combine several forms of prior information, forming a powerful constraint when performing multiframe demosaicking. Details about these terms are provided below:

• *Data Fidelity Penalty Term*
In the multiframe demosaicking case, the data fidelity term must include all three color channels. Considering the general motion and blur model of Equation 19.1, a reasonable L_2-based multiframe data fidelity penalty term is given by:

$$J_1(\mathbf{x},\{\mathbf{y}(k)\}) = \sum_{i=R,G,B}\sum_{k=0}^{N-1} \|\mathbf{ADHF}(k)\mathbf{x}_i - \mathbf{y}_i(k)\|_2^2. \tag{19.13}$$

Essentially, this functional penalizes estimates of the high resolution color image that, when passed through the forward model of Equation 19.1, differs from the observed image. The data penalty function, Equation 19.13, works well when the model is accurate (e.g., the motion estimation produces accurate estimates of the forward imaging model, and the blur is the correct one). When these are not true, the forward model contains errors, and a more robust form of Equation 19.13 could be proposed, based on the L_1 norm:

$$J_1(\mathbf{x},\{\mathbf{y}(k)\}) = \sum_{i=R,G,B}\sum_{k=0}^{N-1} \|\mathbf{ADHF}(k)\mathbf{x}_i - \mathbf{y}_i(k)\|_1. \tag{19.14}$$

This is the full-color generalization of the robust monochromatic superresolution penalty term of Equation 19.10. As before, this penalty term provides robustness at the expense of slight increase in computational complexity, and reduced denoising efficiency when applied to Gaussian noise.

• *Spatial Luminance Penalty Term*
The human eye is more sensitive to the details in the luminance component of an image than the details in the chrominance components [13]. It is important, therefore, that the edges in the luminance component of the reconstructed high-resolution image look sharp. As explained in Reference [47], the BTV functional of Equation 19.11 produces images with sharp edges. For the multiframe demosaicking application, we apply the BTV cost

function to the reconstruction of the luminance component. The luminance image can be calculated as the weighted sum $\mathbf{x}_L = 0.299\mathbf{x}_R + 0.597\mathbf{x}_G + 0.114\mathbf{x}_B$, [14]. This luminance image is the Y component of the YIQ image format commonly found in video coding. Using λ_1 to denote the strength of the image prior, the BTV luminance regularization term is then defined as

$$P_1(\mathbf{x}) = \lambda_1 \sum_{l=-L_{max}}^{L_{max}} \sum_{m=-M_{max}}^{M_{max}} \alpha^{|m|+|l|} \|\mathbf{x}_L - \mathbf{S}_x^l \mathbf{S}_y^m \mathbf{x}_L\|_1. \tag{19.15}$$

• *Spatial Chrominance Penalty Term*
Spatial regularization is required also for the chrominance channels. However, since the human visual system is less sensitive to the resolution of these bands, we can apply a simpler regularization functional, based on the L_2 norm, in the form of

$$P_2(\mathbf{x}) = \lambda_2 \left(\|\Lambda \mathbf{x}_I\|_2^2 + \|\Lambda \mathbf{x}_Q\|_2^2 \right), \tag{19.16}$$

where the images $\mathbf{x}_I$ and $\mathbf{x}_Q$ are the I and Q portions of the YIQ color representation and λ_2 defines the strength of the image prior. As before, Λ is a spatial high-pass operator such as derivative, Laplacian, or even identity matrix. Such a penalty term encourages the two chrominance components to vary smoothly over the high resolution image.

• *Spectral Dependency Penalty Term*
This penalty term characterizes the spectral or interchannel correlation between the three channels of the original color images. This penalty function minimizes the mismatch between locations or orientations of edges across the color bands. Following Reference [12], minimizing the vector product norm of any two adjacent color pixels forces different bands to have similar edge location and orientation. Based on the theoretical justifications in Reference [68], the authors of Reference [12] suggest a pixelwise spectral dependencies cost function to be minimized. This term has the vector outer product norm of all pairs of neighboring color pixels. With some modifications to what was proposed in Reference [12], our spectral dependencies penalty term is a differentiable cost function

$$P_3(\mathbf{x}) = \lambda_3 \sum_{l,m=-1}^{1} \Big[\|\mathbf{x}_G \odot \mathbf{S}_x^l \mathbf{S}_y^m \mathbf{x}_B - \mathbf{x}_B \odot \mathbf{S}_x^l \mathbf{S}_y^m \mathbf{x}_G\|_2^2 +$$
$$\|\mathbf{x}_B \odot \mathbf{S}_x^l \mathbf{S}_y^m \mathbf{x}_R - \mathbf{x}_R \odot \mathbf{S}_x^l \mathbf{S}_y^m \mathbf{x}_B\|_2^2 + \|\mathbf{x}_R \odot \mathbf{S}_x^l \mathbf{S}_y^m \mathbf{x}_G - \mathbf{x}_G \odot \mathbf{S}_x^l \mathbf{S}_y^m \mathbf{x}_R\|_2^2 \Big], \tag{19.17}$$

where $\odot$ is the element by element multiplication operator. The λ_3 defines the strength of the image prior.

As with the monochromatic superresolution case, we minimize the generic full-color cost function from Equation 19.12 using a type of SD. In each step, the derivative will be computed with respect to one color channel assuming the other color channels are fixed. Thus, for the $n+1$ iteration, the algorithm updates the ith color channel according to

$$\widehat{\mathbf{x}}_i^{n+1} = \widehat{\mathbf{x}}_i^n - \beta \nabla_{x_i} \Omega(\mathbf{x}^n) \qquad i = R, G, B \tag{19.18}$$

where the scalar β is the steepest descent step size. This minimization approach converges relatively quickly for most image sequences, providing satisfactory results after ten to fifteen such iterations.

(a) (b)

FIGURE 19.10 (See color insert.)

Multiframe color superresolution applied to a real data sequence. (a) One of thirty-one low-resolution input images of size $141 \times 147 \times 3$ demosaicked by the single-frame method of Reference [7]. (b) Resulting image after multiframe demosaicking using the robust data penalty term of Equation 19.14. © 2007 IEEE

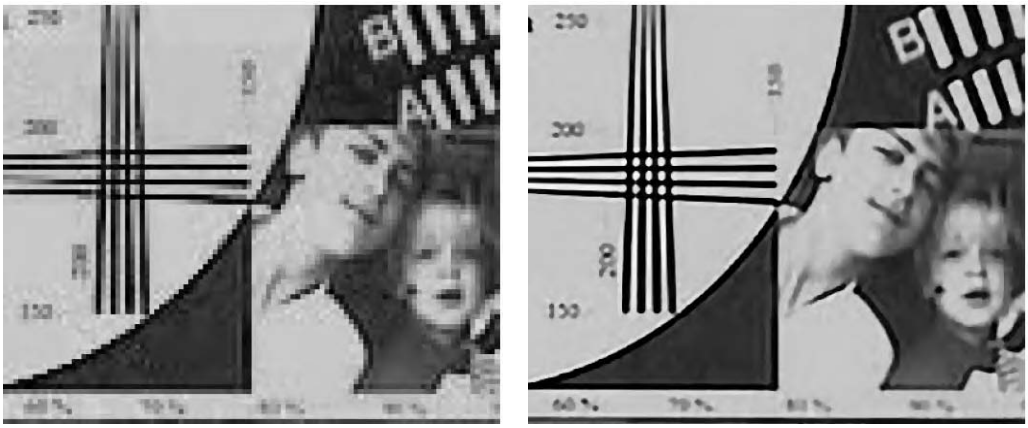
19.3.2 Multiframe Demosaicking Examples

In this section, we present some visual results of the multiframe demosaicking algorithm described in the previous section. The first example demonstrates the advantages of the multiframe demosaicking algorithm when dealing with raw CFA images taken directly from an image sensor. In this example, we acquired thirty-one uncompressed raw CFA images from a two megapixel CMOS sensor. The (unknown) camera point spread function (PSF) was modelled as a tapered 5×5 disk PSF. In all examples, to compute the unknown motion matrices, the raw CFA images were first demosaicked by the single-frame method from Reference [7]. Then, the luminance component of these images was registered in a pairwise fashion using the motion estimation algorithm[5] described in Reference [70].

Figure 19.10a shows a single low-resolution image after applying the single-frame demosaicking method of Reference [7]. The single-frame demosaicking method exhibits the typical problems of low-resolution, color aliasing, and sensor noise. Figure 19.10b shows the resulting image after applying the multiframe demosaicking algorithm directly to the set of low-resolution CFA images to increase the spatial resolution by a factor of three.

In the second example, we repeated the above experiment using a different camera, capturing twenty-six color filtered images. Figure 19.1a shows a single low-resolution image after applying more complex single-frame demosaicking from Reference [9] and Figure 19.1b shows the resulting image after applying the multiframe demosaicking algorithm with the data penalty term of Equation 19.14 directly to the set of low-resolution CFA images to increase the spatial resolution by a factor of three. Both examples depicted in Figure 19.1b and Figure 19.10b show improved resolution and almost no color artifacts.

[5] Application of an accurate multiframe registration technique (see Appendix) for directly registering the CFA data is described in Reference [69].



(a) (b)

FIGURE 19.11 (See color insert.)

(a) One of forty captured images which are demosaicked and compressed in an unknown fashion by the imaging system. These images were used to reconstruct the higher resolution image in (b) by exploiting the multiframe demosaicking algorithm.

In the third example, we apply the multiframe demosaicking algorithm to a set of color images which have already undergone single-frame demosaicking and compression. Figure 19.11a shows an example low-resolution image which has undergone an unknown demosaicking algorithm followed by compression. Forty such images were combined using the multiframe demosaicking algorithm to produce the image on the right. This resulting image demonstrates the capability of the multiframe demosaicking approach to handle images later in the imaging pipeline after standard compression. While the results would presumably be better if the multiframe algorithm had direct access to the raw CFA images, the final image shows significant improvement over the input image.

19.4 Fast and Dynamic Multiframe Demosaicking

In this section, first we describe the conditions under which the multiframe demosaicking algorithm described in the previous section may be implemented very efficiently (Section 19.4.1). Using this fast formulation, in Section 19.4.2 we extend the multiframe demosaicking algorithm to *dynamic* multiframe processing which differs from basic multiframe demosaicking in that the final result is a high resolution image sequence or video. This is also known as video-to-video superresolution, which has been addressed in the literature for the grayscale [71], [72], [73] and color (demosaicked) [74] sequences.

19.4.1 Fast Multiframe Demosaicking

While the iterative method described in the previous section is fast enough for many practical applications, speeding up the implementation is desirable when handling large

amounts of data. In this section, we describe a fast, two-stage approach to multiframe demosaicking, in the spirit of the method developed in Reference [75]. The two stage approach works only when two conditions are satisfied. First, the common system PSF is spatially invariant. Second, the motion between the low-resolution frames (at least locally) is translational in nature. When these two conditions are satisfied, the shifting operator $\mathbf{F}(k)$ and the blurring operator $\mathbf{H}$ commute. We then define $\mathbf{z} = \mathbf{Hx}$ as the unknown high-resolution blurry image. In the fast approach, multiframe demosaicking is divided into the following two steps:

1. Noniterative data fusion (shift-and-add – see below) provides an estimate of $\mathbf{z}$ from the captured data. The model for this step is

$$\mathbf{y}(k) = \mathbf{ADF}(k)\mathbf{z} + \mathbf{v}(k) \tag{19.19}$$

2. Iterative deblurring and interpolation provides an estimate of $\mathbf{x}$ from the estimate $\hat{\mathbf{z}}$.

The two-step approach is fast since the first stage is noniterative. After this stage, we no longer need to store the set of capture images $\{\mathbf{y}(k)\}$. In essence, the estimate $\hat{\mathbf{z}}$ is a sufficient statistic with which we can estimate the high resolution image $\mathbf{x}$.

19.4.1.1 Data Fusion Step

The data fusion step solves the estimation problem for the model in Equation 19.19 in a noniterative fashion for each color channel independently by using an analytic minima of a simple data penalty cost function [47]. If the penalty function is based on the L_2 norm penalty function

$$\sum_k \|\mathbf{y}_i(k) - \mathbf{DF}(k)\mathbf{z}_i\|_2^2, \;\; i = R, G, B \tag{19.20}$$

the fusion step is called the shift-and-add algorithm. In this algorithm, the input images are upsampled by zero-padding, shifted by the inverse of the translation amount, and averaged over the set of N images. Figure 19.12 shows a visual example of the shift-and-add process for one color channel. This is equivalent to the ML estimate of the image $\mathbf{z}$. As shown in Reference [47], if the robust L_1 data fidelity term

$$\sum_k \|\mathbf{y}_i(k) - \mathbf{DF}(k)\mathbf{z}_i\|_1 \;\; i = R, G, B \tag{19.21}$$

is used, then a pixelwise median operation over the set of frames implements the robust penalty function of Equation 19.14. This constitutes the robust shift-and-add algorithm, the result of which is a single image containing estimates of the blurry image $\mathbf{z}$.

As apparent in the fused image in Figure 19.12, the color sampling pattern after shift-and-add can be quite arbitrary depending on the relative motion of the low-resolution images. For some pixels we may not have any value, while for some we might have multiple measurements of one, two, or even all three color bands. Because this sampling pattern is arbitrary, we rely on the iterative deblurring and interpolation algorithm to restore the missing information. As an aside, Figure 19.12 partly explains the inferiority of the sequential strategy of first demosaicking the low-resolution images followed by superresolution. Image motion between captured frames will often provide data in some of the pixel

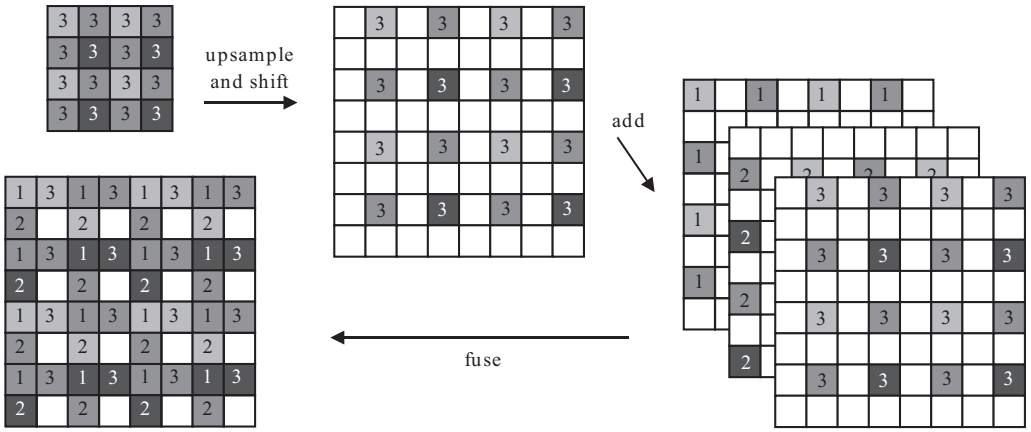


FIGURE 19.12 (See color insert.)

Diagram showing an example of the shift-and-add process. The input color channel image is upsampled by the resolution enhancement factor $r = 2$, and shifted according to the inverse of the image translation. This shifted image is added and averaged with the other $N-1$ (in this case 2) upsampled and shifted low-resolution images.

locations lacking color data in a single CFA image. Interpolating the missing pixels for each individual frame prematurely estimates the missing pixel data thereby limiting the final performance of multiframe resolution enhancement.

19.4.1.2 Deblurring and Interpolation Step

The second step uses the estimate of the blurry high resolution image $\hat{\mathbf{z}}$ to estimate the high resolution image $\mathbf{x}$. This step relies on a cost function very similar to that of Equation 19.12 with the exception that the data penalty term is replaced by

$$\Omega(\mathbf{x},\hat{\mathbf{z}}) = \sum_{i=R,G,B} \|\Phi_i(\mathbf{H}\mathbf{x}_i - \hat{\mathbf{z}}_i)\|_2^2 + P_1(\mathbf{x}) + P_2(\mathbf{x}) + P_3(\mathbf{x}) \tag{19.22}$$

The matrix Φ_i (for $i = R,G,B$) is a diagonal matrix with diagonal values equal to the square root of the number of measurements that contributed to make each element of $\hat{\mathbf{z}}_i$. In this way, the data fidelity term applies no weight to pixel locations which have no observed data. On the other hand, those pixels which represent numerous measurements have a stronger influence on the data fidelity penalty function. Here, we observe that the data fidelity term no longer requires summation over the set of captured frames, saving a considerable amount of processing time as well as memory.

19.4.2 Dynamic Multiframe Demosaicking

Armed with the fast implementation of the multiframe demosaicking algorithm, we turn to apply this algorithm to video sequences. This is called *dynamic* multiframe demosaicking that produces an image *sequence* or video with higher resolution. A naive approach to this problem is to apply the static algorithm on a set of images while changing the reference frame. However, the memory and computational requirements for the static process are so taxing as to preclude its direct application to the dynamic case.

In contrast, we present a dynamic algorithm which takes advantage of the fact that the resulting high resolution image for the previous time frame $t-1$ helps predict the solution for the current time frame t. In this section, we replace the generic frame index k with t to indicate the temporal ordering of the low-resolution frames. A simple forward model capturing the temporal relationship of the low-resolution images is

$$\mathbf{x}(t) = \mathbf{F}(t)\mathbf{x}(t-1) + \mathbf{u}(t), \tag{19.23}$$

and

$$\mathbf{y}(t) = \mathbf{ADH}(t)\mathbf{x}(t) + \mathbf{v}(t). \tag{19.24}$$

In other words, the current frame is a shifted version of the previous frame with some additive noise with covariance $\mathbf{C}_u(t)$.

The equations given above describe a system in its *state-space* form, where the state is the desired ideal image. Thus, a Kalman-filter (KF) [76] formulation can be employed to recursively compute the optimal estimates $(\mathbf{x}(t), t \in \{1,...,N\})$ from the measurements $(\mathbf{y}(t), t \in \{1,...,N\})$, assuming that $\mathbf{D}, \mathbf{H}$, $\mathbf{F}(t)$, $\mathbf{C}_u(t)$, and $\mathbf{C}_v(t)$ (covariance of $\mathbf{v}(t)$) are known [35], [71], [72]. In the following, we study the application of the causal Kalman filter, where the estimates are obtained through on-line processing of an incoming sequence.[6]

To implement the dynamic multiframe demosaicking algorithm in a fast and efficient manner, we rewrite the state space model Equations 19.23 and 19.24 in terms of $\mathbf{z}$ ($\mathbf{z}(t) = \mathbf{Hx}(t)$) as

$$\mathbf{z}(t) = \mathbf{F}(t)\mathbf{z}(t-1) + \mathbf{e}(t), \tag{19.25}$$

and

$$\mathbf{y}(t) = \mathbf{ADz}(t) + \mathbf{v}(t). \tag{19.26}$$

Note that the first of the two equations is obtained by left multiplication of both sides of Equation 19.23 by $\mathbf{H}$ and using the fact that it commutes with $\mathbf{F}(t)$. Thus, the perturbation vector $\mathbf{e}(t)$ is a colored version of $\mathbf{u}(t)$, leading to $\mathbf{C}_e = \mathbf{HC}_u\mathbf{H}^T$ as its covariance matrix.

The following defines the forward Kalman propagation and update equations [35] that account for a causal (on-line) process. We assume that at time $t-1$ we already have the mean-covariance pair, $(\hat{\mathbf{z}}(t-1), \hat{\Pi}(t-1))$, and those should be updated to account for the information obtained at time t. We start with the covariance matrix update based on Equation 19.25:

$$\tilde{\Pi}(t) = \mathbf{F}(t)\hat{\Pi}(t-1)\mathbf{F}^T(t) + \mathbf{C}_{e(t)}, \tag{19.27}$$

where $\tilde{\Pi}(t)$ is the propagated covariance matrix (initial estimate of the covariance matrix at time t). The KF gain matrix is given by

$$\mathbf{K}(t) = \tilde{\Pi}(t)(\mathbf{AD})^T[\mathbf{C}_v(t) + \mathbf{AD}\tilde{\Pi}(t)\mathbf{D}^T]^{-1}. \tag{19.28}$$

Based on $\mathbf{K}(t)$, the updated state vector mean is computed by

$$\hat{\mathbf{z}}(t) = \mathbf{F}(t)\hat{\mathbf{z}}(t-1) + \mathbf{K}(t)[\mathbf{y}(t) - \mathbf{ADF}(t)\hat{\mathbf{z}}(t-1)]. \tag{19.29}$$

[6] A closely related noncausal processing mode, where every high-resolution reconstructed image is derived as an optimal estimate incorporating information from all the frames in the sequence, is studied in Reference [74].

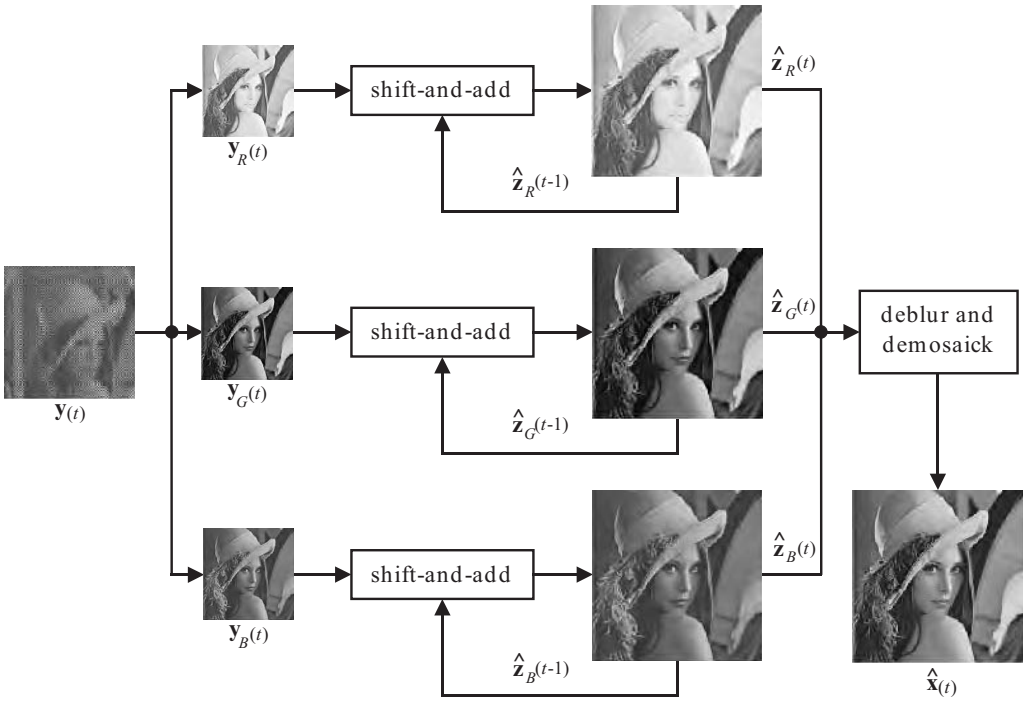


FIGURE 19.13

Block diagram representation of the fast dynamic multiframe demosaicking process.

The final stage requires the update of the covariance matrix, based on Equation 19.26 as follows:

$$\hat{\Pi}(t) = \mathrm{Cov}\,(\hat{\mathbf{z}}(t)) = [\mathbf{I} - \mathbf{K}(t)\mathbf{AD}]\tilde{\Pi}(t). \tag{19.30}$$

More on the meaning of these equations and how they are derived can be found in References [35] and [77].

While in general the above equations require the propagation of intolerably large matrices in time, if we refer to $\mathbf{C}_e(t)$ as a diagonal matrix, then $\tilde{\Pi}(t)$ and $\hat{\Pi}(t)$ are diagonal matrices. Following References [71] and [72], if we choose a matrix $\sigma_e^2\mathbf{I} \geq \mathbf{C}_e(t)$, it implies that $\sigma_e^2\mathbf{I} - \mathbf{C}_e(t)$ is a positive semi-definite matrix, and there is always a finite σ_e that satisfies this requirement. Replacing $\mathbf{C}_e(t)$ with this majorizing diagonal matrix, the new state-space system in Equations 19.25 and 19.26 simply assumes a stronger innovation process. The effect on the KF is to rely less on the temporal relation in Equation 19.25 and more on the measurements in Equation 19.26. Since $\mathbf{C}_e(t)$ is diagonal, Equations 19.27, 19.28, 19.29, and 19.30 can be implemented on an extremely fast pixel-by-pixel basis [74].

The output of the temporal Kalman filter equations (Equations 19.27, 19.28, 19.29, and 19.30) is an estimate of the blurred high resolution video sequence $\hat{\mathbf{z}}(t)$. We refer to this as the dynamic shift-and-add sequence. At this point, we apply the iterative deblurring and interpolation step described earlier. The iterative deblurring of the blurry image $\hat{\mathbf{z}}(t)$ for time t is accomplished by minimizing the cost function

$$\Omega(\mathbf{x}(t)) = J_1'(\mathbf{x}(t), \hat{\mathbf{z}}(t)) + P_1(\mathbf{x}(t)) + P_2(\mathbf{x}(t)) + P_3(\mathbf{x}(t)) \tag{19.31}$$

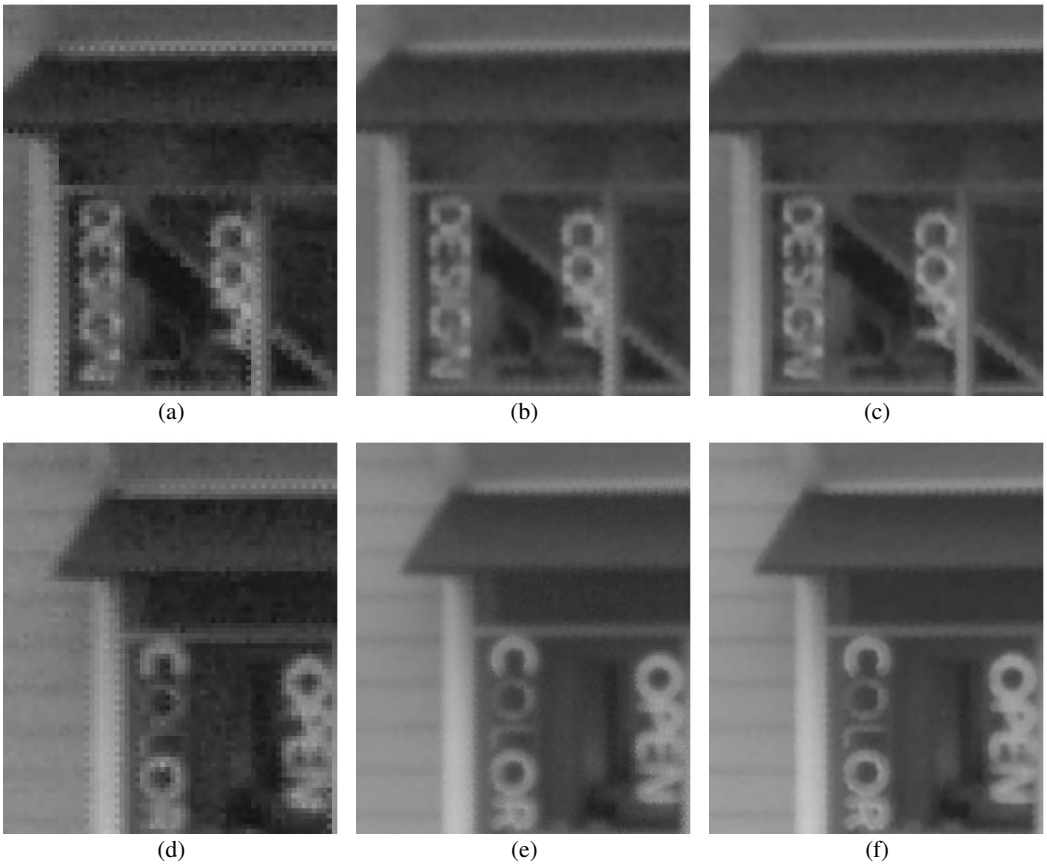


FIGURE 19.14

The images in the left column show frames #1 (top) and #69 (bottom) of the 74-frame low-resolution CFA image sequence. The second column shows the results after the recursive dynamic shift-and-add processing increasing their resolution by a factor of three in each direction. The right column shows the images after iteratively deblurring the dynamic shift-and-add images. We observe that the dynamic shift-and-add performs much of the work, while the subsequent deblurring restores the final details to the images.

using the forward shifted version of the previous estimate $\mathbf{F}(t)\hat{\mathbf{x}}(t-1)$ as the initial guess. By relying on the shifted high resolution estimate from the previous frame $\hat{\mathbf{x}}(t-1)$ as an initial guess, very few iterations are required for subsequent frames.

Figure 19.13 shows the block diagram describing fast dynamic multiframe demosaicking. The algorithm uses only the forward causal processing model; however, it can be extended to consider both forward filtering and smoothing, as described in References [65] and [78].

19.4.3 Example of Dynamic Multiframe Demosaicking

We present an example demonstrating the dynamic multiframe demosaicking algorithm. As before, we used seventy-four uncompressed, raw CFA images from a video camera (based on Zoran 2MP CMOS Sensors). The upper and lower images in the left column

of Figure 19.14 show the low-resolution frames (frames number 1, and 69, respectively) demosaicked by the method in Reference [7]. The central images show the resulting color images after the recursive dynamic shift-and-add processing [74]. Some of the resolution has been restored at this point. The images in the right column show the iteratively deblurred versions of the shift-and-add images. Overall, the final image sequences show significant improvement over the original sequence.

19.5 Conclusion

In this chapter, we discussed superresolution and demosaicking as two important inverse problems in imaging. More significantly, we presented a unified approach for simultaneously solving these two interrelated problems. We illustrated how using the robust L_1 norm for the data fidelity term makes the proposed methods robust to errors in both measurements and in modelling. Furthermore, we used the bilateral regularization of the luminance term to achieve sharp reconstruction of edges; meanwhile the chrominance and spectral dependencies cost functions were tuned to remove color artifacts from the final estimate.

While the implementation of the resulting algorithms may appear complex at first blush, from a practical point of view, all matrix-vector operations in the proposed methods are implementable as simple image operators, making the methods practically feasible and possibly useful for implementation on specialized or general purpose hardware within commercially available cameras. Furthermore, as these operations are locally performed on pixel values on the HR grid, parallel processing may also be used to further increase the computational efficiency.

As we have pointed out, accurate subpixel motion estimation is an essential part of any image fusion process such as the proposed simultaneous multiframe superresolution and demosaicking. While multiscale methods based upon the optical flow or phase correlation principles are adequate, they are not specifically tailored or appropriate for estimating motions between color-filtered images. In fact, there is currently no significant body of literature that addresses the problem of estimating subpixel motion between Bayer filtered images. This gap in the literature needs to be filled, which could result in improved versions of the algorithms suggested in this chapter. More broadly, a new paradigm (Reference [25]) along the lines is needed in which accurate subpixel motion estimation is performed jointly inside and along with the overall superresolution/demosaicking problem, instead of what has traditionally been a preprocessing step.

Acknowledgments

We would like to thank Lior Zimet and Erez Galil from Zoran Corp. for providing the camera used to produce the raw CFA images. We would also like to thank Eyal Gordon from the Technion-Israel Institute of Technology for helping us capture the raw CFA im-

ages used in the Figure 19.1 experiment. We thank Prof. Ron Kimmel of the Technion for providing us with the code that implements the method from Reference [9]. We would like to thank Prof. Joseph Y. Lo and Prof. Cynthia A. Toth for helping us capturing and processing the low-resolution images in Figure 19.8. The image sequence used in the experiment of Figure 19.11 is courtesy of Adyoron Intelligent Systems Ltd., Tel Aviv, Israel.

This work was supported in part by the U.S. Air Force under Grant F49620-03-1-0387. Sina Farsiu was supported in part by the above and by the North Carolina Biotechnology Center's Collaborative Funding Grant (CFG).

Figure 19.7 and Figure 19.14 are reprinted from [79] with the permission of SPIE. Figure 19.6, Figure 19.9, Figure 19.10, and Figure 19.11 are reprinted from [65] with the permission of IEEE.

Appendix: Motion Estimation

The first step in solving the multiframe demosaicking problem is estimating the motion between the collection of CFA images. Numerous image registration techniques have been developed throughout the years [80]. Of these, optical flow [81], [82], and correlation-based methods [83] are among the most popular.

While the detailed description of these techniques is out of the scope of this chapter, we note that these methods are mainly developed to estimate the relative motion between *a pair* of frames. For cases where several images are to be registered with respect to each other (e.g., superresolution applications), two simple strategies are commonly used. The first is to register all frames with respect to a single reference frame [47]. This may be called the *anchoring* approach, as illustrated in Figure 19.15a. The choice of a reference or anchor frame is rather arbitrary, and can have a severe effect on the overall accuracy of the resulting estimates. This caveat aside, overall, this strategy is effective in cases where the camera motion is small and random (e.g., small vibrations of a gazing camera).

The other popular strategy is the *progressive* registration method [32], [78], where images

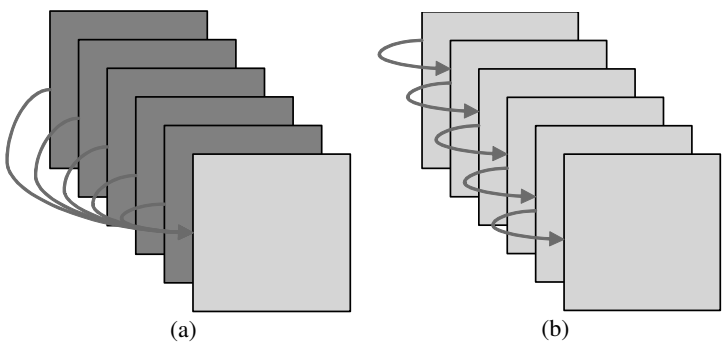


FIGURE 19.15

Common strategies used for registering frames of a video sequence. (a) Fixed reference ("anchored") estimation. (b) Pairwise ("progressive") estimation.

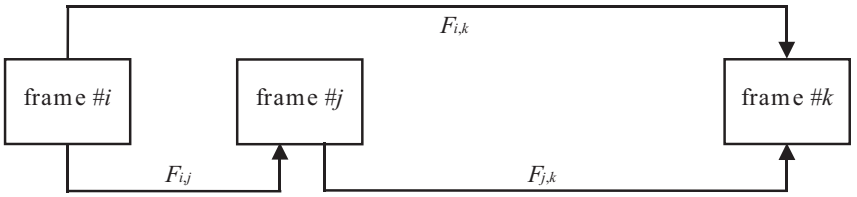


FIGURE 19.16

The consistent motion property dictates that the motion between any pair of frames must be the composition of the motion between two other pairs of frames.

in the sequence are registered in pairs, with one image in each pair acting as the reference frame. For instance, taking a causal view with increasing index denoting time, the *i*th frame of the sequence is registered with respect to the $(i+1)$th frame and the $(i+1)$th frame is registered with respect to the $(i+2)$th frame, and so on, as illustrated in Figure 19.15b. The motion between an arbitrary pair of frames is computed as the combined motion of the above incremental estimates. This method works best when the camera motion is smooth. However, in this method, the registration error between two "nearby" frames is accumulated and propagated when such values are used to compute motion between "far away" frames. Neither of the above approaches takes advantage of the important prior information available for the multiframe motion estimation problem. This prior information constrains the estimated motion vector fields as defined in the following.

To begin, let us define $\mathbf{F}_{i,j}$ as the operator which maps (registers) frames indexed i and j as follows:

$$\mathbf{y}_i = \mathbf{F}_{i,j}\{\mathbf{y}_j\},$$

where $\mathbf{y}_i$ and $\mathbf{y}_j$ are the lexicographic reordered vector representations of frames i and j. Now given a sequence of N frames, precisely $N(N-1)$ such operators can be considered. Regardless of considerations related to noise, sampling, and the finite dimensions of the data, there is an inherent intuitive relationship between these pair-wise registration operators. This condition dictates that the operator describing the motion between any pair of frames must be the composition of the operators between two other pairs of frames. More specifically, as illustrated in Figure 19.16, taking any triplet of frames i, j, and k, we have the motion consistency condition as:

$$\forall i, j, k \in \{1, ..., N\}, \quad \mathbf{F}_{i,k} = \mathbf{F}_{i,j} \circ \mathbf{F}_{j,k}. \tag{19.32}$$

Using these types of constraints provides a means for reliable estimation of the sub-pixel motions and hence the warping operators. While a detailed discussion will be out of the scope of this paper, we shall note that the constrained motion property has been successfully implemented in different guises and for various applications such as mosaicking [84], motion estimation [85], [86], [87], and superresolution/demosaicking [25], [69], [82], [88], [89], [90], [91], [92].

References

[1] B.E. Bayer, "Color imaging array," U.S. Patent 3 971 065, July 1976.

[2] R. Lukac and K.N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1260–1267, November 2005.

[3] N.R. Shah and A. Zakhor, "Resolution enhancement of color video sequences," *IEEE Transactions on Image Processing*, vol. 8, no. 6, pp. 879–885, June 1999.

[4] B.C. Tom and A. Katsaggelos, "Resolution enhancement of monochrome and color video using motion compensation," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 278–287, February 2001.

[5] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical Model and Image Processing*, vol. 53, no. 3, pp. 231-239, May 1991.

[6] D.R. Cok, "Signal processing method and apparatus for sampled image signals," U.S. Patent 4 630 307, July 1987.

[7] C.A. Laroche and M.A. Prescott, "Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients," U.S. Patent 5 373 322, December 1994.

[8] J.F. Hamilton and J.E. Adams, "Adaptive color plan interpolation in single sensor color electronic camera," U.S. Patent 5 629 734, May 1997.

[9] R. Kimmel, "Demosaicing: Image reconstruction from color CCD samples," *IEEE Transactions on Image Processing*, vol. 8, no. 9, pp. 1221–1228, September 1999.

[10] L. Chang and Y.P. Tan, "Color filter array demosaicking: New method and performance measures," *IEEE Transactions on Image Processing*, vol. 12, no. 10, pp. 1194–1210, October 2002.

[11] K. Hirakawa and T.W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," in *Proceedings of the IEEE International Conference on Image Processing*, Barcelona, Spain, September 2003, vol. III, pp. 669–672.

[12] D. Keren and M. Osadchy, "Restoring subsampled color images," *Machine Vision and Applications*, vol. 11, no. 2, pp. 197–202, December 1999.

[13] Y. Hel-Or and D. Keren, "Demosaicing of color images using steerable wavelets," Technical report HPL-2002-206R1 20020830, HP Labs, Israel, citeseer.nj.nec.com/548392.html, 2002.

[14] W.K. Pratt, *Digital Image Processing*. New York: John Wiley & Sons, 3rd ed., 2001.

[15] D. Taubman, "Generalized Wiener reconstruction of images from colour sensor data using a scale invariant prior," in *Proceedings of the IEEE International Conference on Image Processing*, Vancouver, Canada, September 2000, vol. III, pp. 801–804.

[16] D.D. Muresan and T.W. Parks, "Demosaicing using optimal recovery," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 267–278, February 2005.

[17] B.K. Gunturk, Y.A. Altunbasak, and R.M. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 997–1013, September 2002.

[18] S.C. Pei and I.K. Tam, "Effective color interpolation in CCD color filter arrays using signal correlation," *IEEE Transactions on Image Processing*, vol. 13, no. 6, pp. 503–513, June 2003.

[19] D. Alleysson, S. Süsstrunk, and J. Hérault, "Color demosaicing by estimating luminance and opponent chromatic signals in the Fourier domain," in *Proceedings of the IS&T/SID 10th Color Imaging Conference*, Scottsdale, AZ, USA, November 2002, pp. 331–336.

[20] R. Ramanath and W.E. Snyder, "Adaptive demosaicking," *Journal of Electronic Imaging*, vol. 12, no. 4, pp. 633–642, October 2003.

[21] L. Zhang and X. Wu, "Color demosaicing via directional linear minimum mean square-error estimation," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2167–2178, December 2005.

[22] R. Lukac, K.N. Plataniotis, D. Hatzinakos, and M. Aleksic, "A new CFA interpolation framework," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 86, no. 7, pp. 1559–1579, July 2006.

[23] R. Lukac, K. Martin, and K.N. Plataniotis, "Colour-difference based demosaicked image postprocessing," *Electronics Letters*, vol. 39, no. 25, pp. 1805–1806, December 2003.

[24] R. Lukac, K. Martin, and K.N. Plataniotis, "Demosaicked image postprocessing using local color ratios," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 914–920, June 2004.

[25] D. Robinson, S. Farsiu, and P. Milanfar, "Optimal registration of aliased images using variable projection with applications to superresolution," *The Computer Journal, Special Issue on Super-Resolution in Imaging and Video*, doi: 10.1093/comjnl/bxm007, accepted for publication.

[26] S. Borman and R.L. Stevenson, "Super-resolution from image sequences — A review," in *Proceedings of the 1998 Midwest Symposium on Circuits and Systems*, Notre Dame, Indiana, April 1998, vol. V, p. 374.

[27] S.C. Park, M.K. Park, and M.G. Kang, "Super-resolution image reconstruction: A technical overview," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 21–36, May 2003.

[28] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Advances and challenges in super-resolution," *International Journal of Imaging Systems and Technology*, vol. 14, no. 2, pp. 47–57, October 2004.

[29] T.S. Huang and R.Y. Tsai, Advances in Computer Vision and Image Processing, ch. Multiframe image restoration and registration, *Advances in Computer Vision and Image Processing*, Greenwich, Connecticut: JAI Press, 1984, vol. 1, pp. 317–339.

[30] D. Robinson, S. Farsiu, J.Y. Lo, P. Milanfar, and C.A. Toth, "Efficient multiframe registration of aliased X-ray images," in *Proceedings of the 41th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, November 2007, pp. 215–219.

[31] N.K. Bose, H.C. Kim, and H.M. Valenzuela, "Recursive implementation of total least squares algorithm for image reconstruction from noisy, undersampled multiframes," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, MN, USA, April 1993, vol. V, pp. 269–272.

[32] L. Teodosio and W. Bender, "Salient video stills: Content and context preserved," in *Proceedings of the First ACM International Conference on Multimedia*, Anaheim, CA, USA, August 1993, vol. X, pp. 39–46.

[33] M. Elad and Y. Hel-Or, "A fast super-resolution reconstruction algorithm for pure translational motion and common space invariant blur," *IEEE Transactions on Image Processing*, vol. 10, no. 8, pp. 1186–1193, August 2001.

[34] M.C. Chiang and T.E. Boulte, "Efficient super-resolution via image warping," *Image and Vision Computing*, vol. 18, no. 10, pp. 761–771, July 2000.

[35] S.M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[36] M. Elad and A. Feuer, "Restoration of single super-resolution image from several blurred, noisy and down-sampled measured images," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1646–1658, December 1997.

[37] S. Peleg, D. Keren, and L. Schweitzer, "Improving image resolution using subpixel motion," *Pattern Recognition Letters*, vol. 5, no. 3, pp. 223–226, March 1987.

[38] N. Nguyen, P. Milanfar, and G.H. Golub, "A computationally efficient image superresolution algorithm," *IEEE Transactions on Image Processing*, vol. 10, no. 4, pp. 573–583, April 2001.

[39] N. Nguyen and P. Milanfar, "Efficient generalized cross-validation with applications to parametric image restoration and resolution enhancement," *IEEE Transactions on Image Processing*, vol. 10, no. 9, pp. 1299-1308, September 2001.

[40] S. Lertrattanapanich and N.K. Bose, "High resolution image formation from low resolution frames using Delaunay triangulation," *IEEE Transactions on Image Processing*, vol. 11, no. 12, pp. 1427–1441, December 2002.

[41] A.J. Patti, M.I. Sezan, and A.M. Tekalp, "Superresolution video reconstruction with arbitrary sampling lattices and nonzero aperture time," *IEEE Transactions on Image Processing*, vol. 6, no. 8, pp. 1326–1333, March 1997.

[42] Y. Altunbasak, A. Patti, and R. Mersereau, "Super-resolution still and video reconstruction from MPEG-coded video," *IEEE Transactions on Circuits and System Video Technology*, vol. 12, no. 4, pp. 217–226, April 2002.

[43] O. Haik and Y. Yitzhaky, "Super-resolution reconstruction of a video captured by a vibrated TDI camera," *Journal of Electronic Imaging*, vol. 15, no. 2, pp. 023006-1-12, April-June 2006.

[44] C.B. Atkins, C.A. Bouman, and J.P. Allebach, "Tree-based resolution synthesis," in *Proceedings of the IS&T Conf. on Image Processing, Image Quality, Image Capture Systems*, Savannah, GA, USA, April 1999, pp. 405–410.

[45] S. Baker and T. Kanade, "Limits on super-resolution and how to break them," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1167–1183, September 2002.

[46] A. Zomet, A. Rav-Acha, and S. Peleg, "Robust super resolution," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Kauai, HI, USA, December 2001, vol. I, pp. 645–650.

[47] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Fast and robust multi-frame superresolution," *IEEE Transactions on Image Processing*, vol. 13, no. 10, pp. 1327–1344, October 2004.

[48] Z.A. Ivanovski, L. Panovski, and L.J. Karam, "Robust super-resolution based on pixel-level selectivity," in *Proceedings of the 2006 SPIE Conference on Visual Communications and Image Processing*, San Jose, CA, USA, January 2006, vol. 6077, pp. 607707-1–8.

[49] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, no. 1–4, pp. 259–268, November 1992.

[50] T.F. Chan, S. Osher, and J. Shen, "The digital TV filter and nonlinear denoising," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 231–241, February 2001.

[51] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the IEEE International Conference on Computer Vision*, New Delhi, India, January 1998, pp. 836–846.

[52] M. Elad, "On the bilateral filter and ways to improve it," *IEEE Transactions on Image Processing*, vol. 11, no. 10, pp. 1141–1151, October 2002.

[53] T.Q. Pham, L.J. van Vliet, and K. Schutte, "Robust fusion of irregularly sampled data using adaptive normalized convolution," *EURASIP Journal on Applied Signal Processing*, vol. 2006, id. 83268, ps. 12, 2006.

[54] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 349–366, February 2007.

[55] M.B. Ezra, A. Zomet, and S.K. Nayar, "Video super-resolution using controlled subpixel detector shifts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 977–987, June 2004.

[56] C.A. Segall, R. Molina, A. Katsaggelos, and J. Mateos, "Bayesian high-resolution reconstruction of low-resolution compressed video," in *Proceedings of the IEEE International Conference on Image Processing*, Thessaloniki, Greece, October 2001, vol. II, pp. 25–28.

[57] C.A. Segall, A.K. Katsaggelos, R. Molina, and J. Mateos, "Bayesian resolution enhancement of compressed video," *IEEE Transactions on Image Processing*, vol. 13, no. 7, pp. 898–911, July 2004.

[58] S. Borman, "Topics in multiframe superresolution restoration," University of Notre Dame, Ph.D. Thesis, Notre Dame, IN, USA, May 2004.

[59] M.G. Kang and S. Chaudhuri, "Super-resolution image reconstruction," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 21–36, May 2003.

[60] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Advances and challenges in super-resolution," *International Journal of Imaging Systems and Technology*, vol. 14, no. 2, pp. 47–57, August 2004.

[61] S. Farsiu, M. Elad, and P. Milanfar, "Multi-frame demosaicing and super-resolution from under-sampled color images," *Proceedings of the 2004 IS&T/SPIE 16th Annual Symposium on Electronic Imaging*, vol. 5299, pp. 222–233, January 2004.

[62] T. Gotoh and M. Okutomi, "Direct super-resolution and registration using raw CFA images," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Washington, D.C., USA, July 2004, vol. II, pp. 600–607.

[63] R. Lukac and K.N. Plataniotis, "Fast video demosaicking solution for mobile phone imaging applications," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 675–681, May 2005.

[64] R. Sasahara, H. Hasegawa, I. Yamada, and K. Sakaniwa, "A color super-resolution with multiple nonsmooth constraints by hybrid steepest descent method," in *Proceedings of the International Conference on Image Processing*, Genoa, Italy, September 2005, vol. I, pp. 857–860.

[65] S. Farsiu, M. Elad, and P. Milanfar, "Multiframe demosaicing and super-resolution of color images," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 141-159, January 2006.

[66] R. Lukac and K.N. Plataniotis, "Adaptive spatiotemporal video demosaicking using bidirectional multistage spectral filters," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 651–654, May 2006.

[67] P. Vandewalle, K. Krichane, D. Alleysson, and S. Süsstrunk, "Joint demosaicing and super-resolution imaging from a set of unregistered aliased images," in *IS&T SPIE Electronic Imaging: Digital Photography III*, San Jose, CA, USA, January 2007, vol. 6502, ps. 12.

[68] D. Keren and A. Gotlib, "Denoising color images using regularization and correlation terms," *Journal of Visual Communication and Image Representation*, vol. 9, no. 4, pp. 352–365, December 1998.

[69] S. Farsiu, M. Elad, and P. Milanfar, "Constrained, globally optimal, multi-frame motion estimation," in *Proceedings of the 2005 IEEE Workshop on Statistical Signal Processing*, Bordeaux, France, July 2005, pp. 1396–1401.

[70] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," *Proceedings of the European Conference on Computer Vision*, vol. 20, pp. 237–252, May 1992.

[71] M. Elad and A. Feuer, "Super-resolution reconstruction of image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 817–834, September 1999.

[72] M. Elad and A. Feuer, "Superresolution restoration of an image sequence: Adaptive filtering approach," *IEEE Transactions on Image Processing*, vol. 8, no. 3, pp. 387–395, March 1999.

[73] C. Newland, D. Gray, and D. Gibbins, "Modified Kalman filtering for image super-resolution," in *Proceedings of the 2006 Conference on Image and Vision Computing*, Great Barrier Island, New Zealand, November 2006, pp. 79–84.

[74] S. Farsiu, M. Elad, and P. Milanfar, "Video-to-video dynamic super-resolution for grayscale and color sequences," *EURASIP Journal on Applied Signal Processing*, vol. 2006, id. 61859, 2006.

[75] M. Elad and Y. Hel-Or, "A fast super-resolution reconstruction algorithm for pure translational motion and common space invariant blur," *IEEE Transactions on Image Processing*, vol. 10, no. 8, pp. 1186–1193, August 2001.

[76] B. Anderson and J. Moore, *Optimal Filtering*. Englewood Cliffs, New Jersey: Prentice-Hall, 1979.

[77] A.H. Jazwinski, *Stochastic Processes and Filtering Theory: The Visual Neurosciences*. New York: Academic Press, 1970.

[78] S. Farsiu, D. Robinson, M. Elad and P. Milanfar, "Dynamic demosaicing and color superresolution of video sequences," in *Proceedings of the SPIE Conference on Image Reconstruction from Incomplete Data III*, Denver, CO, USA, October 2004, vol. 5562, pp. 169–178.

[79] S. Farsiu, M. Elad, and P. Milanfar, "A practical approach to super-resolution," in *Proceedings of the SPIE Conference on Visual Communications and Image Processing*, San Jose, CA, USA, January 2006, vol. 6077, pp. 24–38.

[80] L.G. Brown, "A survey of image registration techniques," *ACM Computing Surveys*, vol. 24, no. 8, pp. 325–376, December 1992.

[81] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the DARPA Image Understanding Workshop*, 1981, pp. 121–130.

[82] W.Y. Zhao and H.S. Sawhney, "Is super-resolution with optical flow feasible?" in *Proceedings of the European Conference on Computer Vision*, Copenhagen, Denmark, May 2002, vol. I, pp. 599–613.

[83] M. Alkhanhal, D. Turaga, and T. Chen, "Correlation based search algorithms for motion estimation," in *Proceedings of the Picture Coding Symposium*, Portland, OR, USA, April 1999, pp. 99–102.

[84] H.S. Sawhney, S.C. Hsu, and R. Kumar, "Robust video mosaicing through topology inference and local to global alignment," in *Proceedings of the European Conference on Computer Vision*, Freiburg, Germany, June 1998, vol. II, pp. 103–119.

[85] V.M. Govindu, "Combining two-view constraints for motion estimation," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Kauai, HI, USA, July 2001, vol. II, pp. 218–225.

[86] V.M. Govindu, "Lie-algebraic averaging for globally consistent motion estimation," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Washington, D.C., USA, July 2004, vol. I, pp. 684–691.

[87] Y. Sheikh, Y. Zhai, and M. Shah, "An accumulative framework for the alignment of an image sequence," in *Proceedings of the Asian Conference on Computer Vision*, Jeju, Korea, January 2004.

[88] P. Vandewalle, S. Süsstrunk, and M. Vetterli, "A frequency domain approach to registration of aliased images with application to super-sesolution," *EURASIP Journal on Applied Signal Processing*, vol. 2006, id. 71459, 2006.

[89] H. Foroosh, J. Zerubia, and M. Berthod, "Extension of phase correlation to subpixel registration," *IEEE Transactions on Image Processing*, vol. 11, no. 3, pp. 188–200, March 2002.

[90] D. Robinson and P. Milanfar, "Statistical performance analysis of super-resolution," *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1413–1428, June 2006.

[91] N.A. Woods, N.P. Galatsanos, and A.K. Katsaggelos, "Stochastic methods for joint registration, restoration, and interpolation of multiple undersampled images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 1, pp. 201–213, January 2006.

[92] J. Chung, E. Haber, and J. Nagy, "Numerical methods for coupled super-resolution," *Inverse Problems*, vol. 22, no. 4, pp. 1261–1272, August 2006.

20

An Overview of Image / Video Stabilization Techniques

Wen-Chung Kao and Sheng-Yuan Lin

20.1 Introduction

With the advances in the technologies of lens modules, image sensors and integrated circuits, commercial high-end cameras have shrunk to a smaller size and lighter weight. However, improving the camera's portability results in problems such as unstable image

and video capture. The images or videos taken by miniature cameras usually have larger vibrations than those taken by bulky cameras. Image stabilization therefore becomes a necessary capability in many consumer digital cameras [1], [2], [3], [4], [5]. It is particularly important in situations where the user takes video in moving vehicles, records and tracks moving objects, or uses the camera in automatic machine vision applications in order to ensure desired visual quality and performance of processing solutions.

Unstable images are caused by undesired hand movements and intentional camera panning. The vibration problem becomes much worse while recording videos in moving vehicles due to road conditions [6], [7], [8]. The unwanted fluctuations of a camera result in unstable image sequences which will affect the visual quality or reduce the recognition performance for machine vision applications. The challenges of an image stabilization system are to accurately predict the camera fluctuations influenced by moving objects in the scene and to compensate for these movements in order to get a stable image or smooth image sequence. Moreover, applying image stabilization in video recording can help to reduce the bit rate of data compression because the performance of motion estimation can be significantly improved after stabilizing the successive frames.

Popular techniques include optical, electronic, and digital image stabilization. The main advantage of optical image stabilization (OIS) is that it can efficiently deal with the vibration problem for both video recoding and still image capture, while its drawback is that the production cost as well as the size of the lens module is increased. An old OIS technique uses an adjustable prism to dynamically change the refraction angle of the focused light according to the inclination of the camera [9], [10]. Two or more gyroscopic sensors are used to detect pitch and yaw of the camera, and then the vertical and horizontal angular velocity signals are passed through a low-pass filter and an analog-to-digital converter (ADC). The adjustable prism is driven using these signals to compensate the fluctuations. The two most popular OIS methods used in current digital cameras are in-lens stabilization and in-camera stabilization. For in-lens stabilization, a correction lens element is moved to counter the effect and bring the light rays back to their original positions. For in-camera stabilization, the sensor board is moved instead of moving the lens element.

Electronic image stabilization (EIS) [11], [12] and digital image stabilization (DIS) adopt similar approaches to compensate for camera vibrations. The only difference between them is they have different approaches to motion vector detection. EIS uses a motion sensor to detect the vector of camera vibration, whereas DIS calculates the global motion vector (GMV) between successive image frames using a software only approach. Based on the estimated GMVs, the vibration can be compensated by dynamically selecting the active pixel area of CCD/CMOS image sensors. Cameras with DIS do not require additional costly hardware components such as motion sensors or adjustable prisms. This is the reason that DIS has attracted many camera designers and researchers.

The most important issue of GMV estimation in a DIS system is to differentiate between hand-jitter, panning and moving objects. Both hand-jitter and panning are caused by the photographer, but only the former needs to be compensated since the latter is the desired camera motion to capture a scene with wider area. On the other hand, identifying moving objects is also important for judging the types of vibrations. The estimated GMV should not be affected by moving objects in the scene. This is the key for image stabilization in real-time video recording. The last remaining issue of DIS is how to fully utilize the

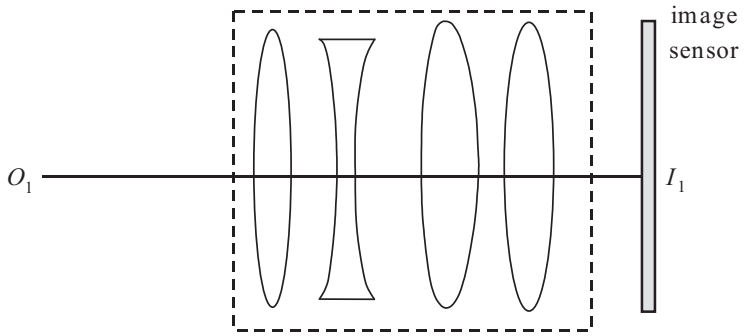


FIGURE 20.1

Normal condition of a lens module.

hardware resources to be compatible with original image/video compression flow.

This chapter presents popular image and video stabilization techniques. Section 20.2 focuses on the OIS design. Section 20.3 presents the problem statements and system architecture of the EIS and DIS systems. The design of a DIS system is discussed in detail in Sections 20.4 and 20.5. Finally, Section 20.6 offers conclusions.

20.2 Optical Image Stabilization

Increasing exposure time is a common approach to capturing still pictures in dark scenes. As the photographer presses the shutter, the camera tends to be shifted slightly due to the movement of the finger. The movement of the camera results in the problem of the image blurred due to shifting the scene to be captured. The situation becomes worse for telephoto lens with longer effective focal length (EFL), since this kind of lens tends to have narrower view angle. A small camera shift will cause larger percentage change in the image. If the scene is bright enough so that the shutter speed is fast (or called short exposure time), the effect is not obvious since the amount of shift is negligible during the exposure period.

Figure 20.1 shows the normal condition of a lens module. The image of the object O_1 is located at the position I_1 on the sensor. If the camera is tilted down while the photographer presses the shutter button, the image of O_1 falls on the lower position. The original image position of the object O_1 becomes the position of another object O_2 as shown in Figure 20.2. Apparently the solution to such an image position change is to move the image point I_1 back to the position I_2. This mechanism is called optical image stabilization (OIS).

20.2.1 Popular Solutions

In general, existing OIS systems redirect the light to the original image position or move the sensor to capture the shifted image. These concepts can be realized by three popular designs: flexible prism architecture [10], [13], in-lens stabilization [14], [15], [16], [17], and in-camera stabilization [18], [19], [20].

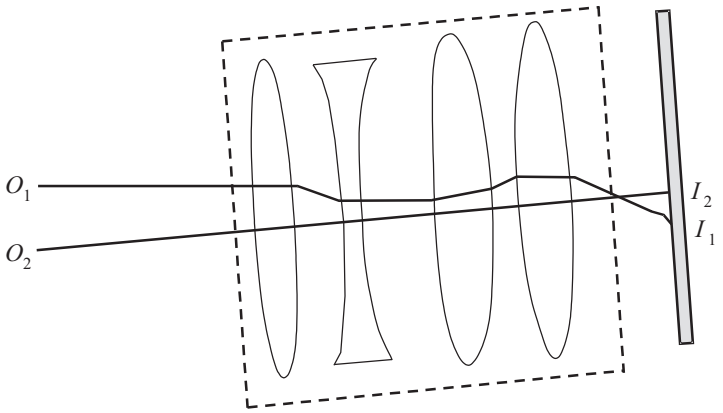


FIGURE 20.2

Lens tilted down.

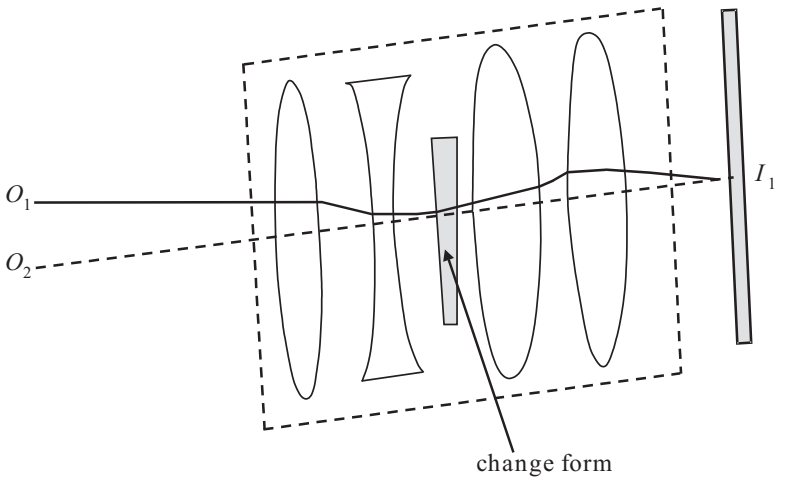


FIGURE 20.3

Fluid prism compensation.

20.2.1.1 Flexible Prism Architecture

The flexible prism architecture (also called vari-angle prism) is shown in Figure 20.3. The prism is filled with fluid and the shape can be changed in order to redirect the light passing through it. In the early days, inertia of the liquid prism was adopted in order to maintain the proper direction of the light beams and this method is deemed the passive OIS. This method has been replaced with active OIS later, in which the movement of the camera is detected with angular speed sensors and the prism is adjusted accordingly. Since the prism is of flat surface and the shift of the light rays is uniform, the performance is much better than other types of OIS system. However, as the control mechanism is more complicated and costly, this method has been replaced by other solutions.

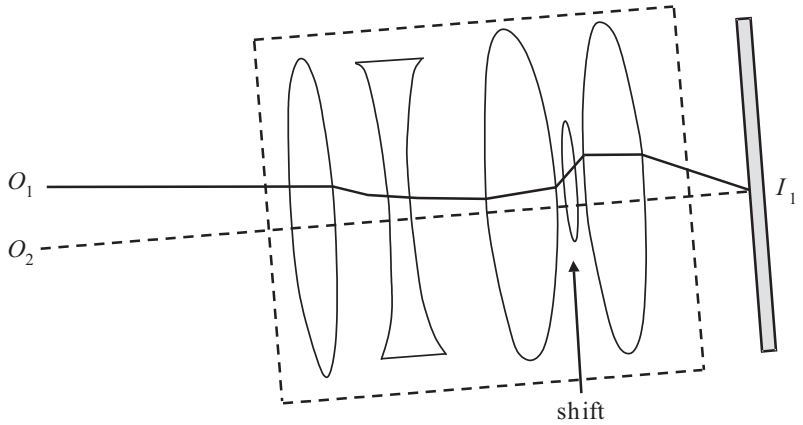


FIGURE 20.4

Lens shift compensation.

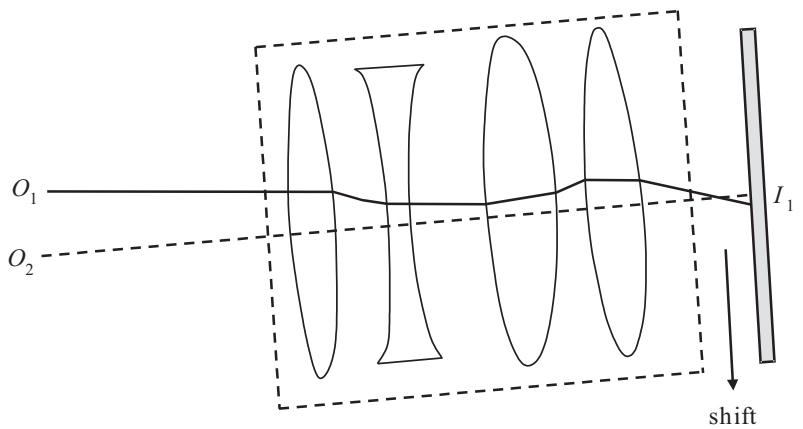


FIGURE 20.5

Sensor shift compensation.

20.2.1.2 In-Lens and In-Camera Optical Image Stabilization

The two most popular OIS methods used in current digital cameras are shown in Figure 20.4 and Figure 20.5. In Figure 20.4, a correction lens is used to control the direction of the incident light beam. Assume the direction of the optical axis is in the z-axis, the correction lens element can shift in x- and y-directions. When the angular speed sensor detects the camera shift, the embedded microprocessor in the camera calculates the exact shift in the image plane. The correction lens element is moved to counter the effect and bring the light rays back to their original positions. This method is usually called in-lens optical stabilization. Another compensation method, called in-camera OIS, is shown in Figure 20.5. This method moves the image sensor rather than the lens element.

For in-lens OIS, the correction lens element lies close to the central part of the optical path. It only needs to move slightly to compensate for camera vibrations. In addition, this single lens element is usually lighter than the entire image sensor module. Thus the

design with such a drive mechanism becomes easier to be implemented. Furthermore, the OIS mechanism can also be integrated with the lens for digital single lens reflex (SLR) cameras, the light rays passing the lens are already stabilized and the images seen through the optical view finder are also stabilized. This is quite useful with telephoto lens for the photographer to be able to view clear images through the view finder.

In-camera stabilization or sensor-shift stabilization can be used with many of the off-the-shelf lenses. For digital SLR cameras with exchangeable lenses, this requires designing a set of OIS systems inside the camera for accommodating different lenses. In reality, the system designers need also consider the image circle of the lens, which means the area on the image plane that the lens can produce a well-focused image. With in-camera stabilization, the sensor needs to be moved laterally to compensate for the image shift caused by the camera vibration. The image circle of the lens needs to be larger than regular lenses. This is not a problem for digital SLR cameras because the image circle is big enough with large lenses. However, the zoom lens of a smaller camera needs to be redesigned and the design limitations are more stringent.

The major disadvantage of a digital SLR camera with in-camera stabilization is that the view finder does not perform stabilization. This is especially troublesome with telephoto lenses. A solution to cope with this trouble is adopting electronic view finder, which relies on the image sensor to continuously capture images and display them on the color LCD for previewing the image. The actuator must be enabled all the time, resulting in unacceptable high power consumptions. Another issue is that different zoom lenses have different performance for camera shaking. The shaking effect for the lenses with longer effective focal length (EFL) is more serious compared to lenses with shorter EFL. The embedded software system of the camera needs to store many sets of calibration data for accommodating different lenses. Furthermore, as the extent for the image sensor to move is usually larger, it is more difficult for designing this driving mechanism and the power consumption becomes higher than other solutions.

On the image processing side, as the sensor is shifted, the conditions of lens fall-off and color shading effects should be adjusted according to the location of the sensor. This makes the embedded software more complicated and increases the product development time. Another issue for digital SLR camera with in-camera stabilization and exchangeable lenses is that different lenses have different optical characteristics and the camera software has to store many sets of parameters in order to achieve consistent performance over different lenses. If there are new lenses developed after the camera body is released for sale, an embedded software update in order to accommodate for new lenses is needed.

In many cases the optical stabilization can enhance the shutter time by up to three stops ($8\times$) or four stops ($16\times$). Ordinary people can hold the camera still and capture an acceptable image with setting exposure time shorter than 1/60 second. With optical image stabilization, the exposure time can be typically increased up to 1/8 or 1/4 second without blurring the images. The camera incorporated with OIS helps ordinary people to take clear pictures in most of the scenes at will.

If the image stability is of the top priority, the best solution would be using a gyroscope in the camera [21]. With the high speed spinning of the gyroscope, the camera can maintain its direction and it is more difficult to shift the camera. However, when the photographer wants to take another picture with a different scene, it is difficult to move the camera to

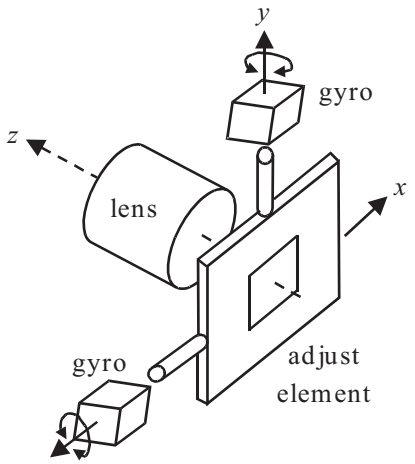


FIGURE 20.6

The detection of camera shaking by gyro sensors.

the new direction. The gyroscope cannot be stopped immediately so that the stabilizer with gyroscope is only applied on some special conditions.

20.2.2 Control Mechanisms

Most of current day active OIS systems rely on gyro sensors to measure the shift or rotation of the camera [22]. Usually two gyro sensors are equipped to measure the angular velocity of the camera in x- and y-directions as shown in Figure 20.6. The output signals from the sensors are processed with analog circuits to remove noise and converted to digital data through an analog-to-digital converter (ADC). The microprocessor embedded in the control system then analyzes the data to calculate the shift amount of the control lens or sensor to compensate for the camera vibration.

The in-lens driving mechanism is similar to the pick-up lens stabilization mechanism used in optical disc as shown in Figure 20.7. The frame holds the lens element and allows it to shift inside the space. There are two coils on the poles and the frame is made of magnetic material. As the coils are driven by electric current, the lens elements can move in the x- and y-directions. There are many alternative mechanical designs to achieve the objective [23], [24], [25], [26]. The driving actuator can be a traditional stepping motor, piezoelectric motor or magnets. If stepping motor or piezoelectric motor are adopted, the actuator stops at the default position with electric power turned off or when the stabilization function is turned off. The actuators with magnets provide a low-cost solution. But a more complicated mechanical design is needed to hold the lens element at the specified location when the power is turned off.

In practice, there is a delay between the microprocessor receiving the data of angular speed and finishing the calculation of adjustment offset. The camera might have already shifted to a new position in this time duration. A motion prediction algorithm should be adopted to estimate the instant camera position. This is especially true in still image capture, since there is a period of time between the user pressing the shutter button and when

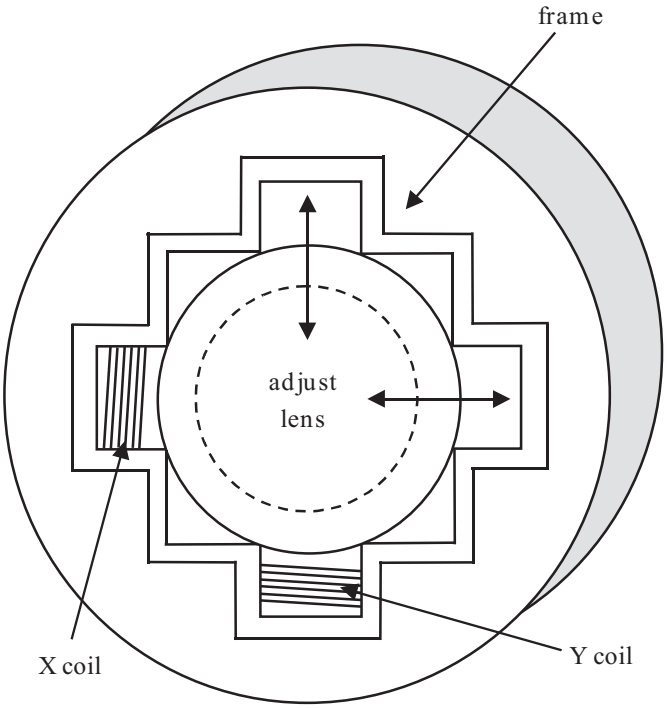


FIGURE 20.7

A simple lens adjustment mechanism.

the exposure starts. A camera should perform auto-exposure and auto-focus before starting image exposure in still image capture mode. During this period, the microprocessor must collect enough data to predict the locus of the camera movement.

The architecture of a basic in-camera driving mechanism is shown in Figure 20.8. The CCD is assembled on a plate that also integrates the driving mechanism. Electric coils can be formed on the plate at four sides of the CCD and provide both horizontal and vertical movement capability. There is usually another plate that supports the CCD plate and makes sure the CCD can be moved in a perfect plane perpendicular to the optical axis of the lens. The coils are typically equipped in pairs in order to balance the movement of the CCD. Since the coils are bigger, the power consumption for moving the CCD module board is also higher. In addition, the weight as well as the metal plate is also much higher than a simple lens.

There are two approaches to optical image stabilization. One is continuously compensating for the camera vibrations in the preview mode of the digital camera operation. The other one is to stay fixed with the control lens at center in the preview mode. The sensor location is adjusted only when the user presses the shutter button. In the first mode, the user can get a steady image displayed on the color LCD screen and be sure to obtain a good quality picture. Unfortunately, it is more difficult for the motor to drive the control lens or image sensor to a larger extent when the user presses the shutter button. This is due to the fact that the lens might lie at one position while the microprocessor needs it to move to the opposite position. With the second optical stabilization mode, both the control lens and

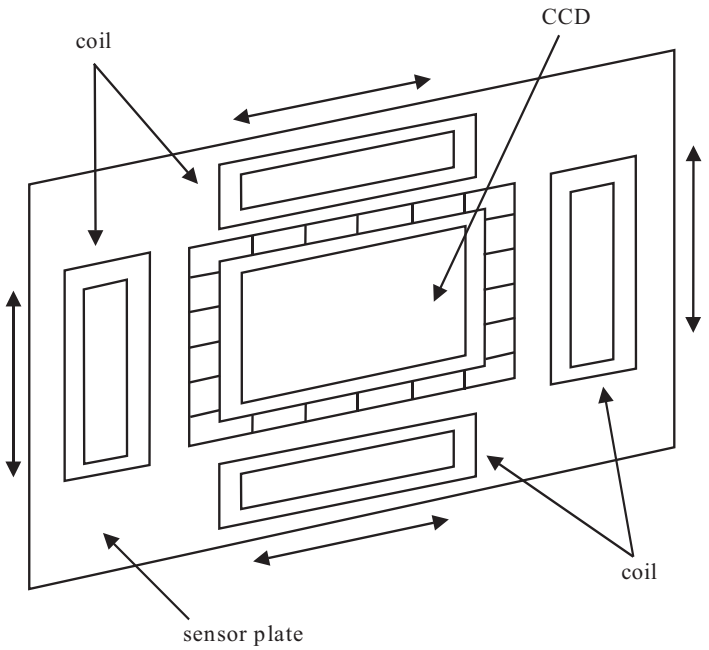


FIGURE 20.8

A simple sensor adjustment mechanism.

image sensor always lie at the center or their default positions in the preview mode. The lens or sensors are moved only after the user presses the shutter button. During the period between the shutter button being pressed and the image being captured, there is enough time for the microprocessor to determine the right amount of movement. Hence the control lens or image sensor can move and compensate for a larger extent of shift caused by vibration.

20.3 Digital and Electronic Image Stabilization

This section reviews the system design of electronic and digital image stabilization. These two approaches are quite similar except for the camera vibration detection as well as motion vector estimation method. The electronic stabilization uses a hardware sensor to detect the camera vibration, while the digital image stabilization analyzes successive images. After the motion vector has been determined, both electronic and digital image stabilization can use similar motion compensation methods. Since DIS gains the advantage of lower hardware cost, it has attracted more researchers to put their efforts on studying it. Hence the remaining text of this section will only focus on the description of DIS algorithms.

The most important steps in DIS are the determination and compensation of the camera motion vector. Popular DIS algorithms differ in one or more steps listed below:

- feature extraction to calculate local motion vectors,
- representative region selection to calculate local motion vectors,
- local motion vector filtering and the global motion vector calculation,
- determination of the final vectors for motion compensation,
- the way image warping is performed, and
- algorithm implementation on digital cameras.

20.3.1 Problem Formulation

The behavior of motion stabilization can be explained with Figure 20.9, where W and H denote the resolution in a final video frame. W_s and H_s represent the width and height of the effective pixel area for a CCD or CMOS image sensor, respectively. The resolution of a raw video frame should be bigger than a frame in an output video. The DIS algorithm dynamically decides the most suitable sensor regions in the effective area such that the vibration caused by photographers can be well compensated. For the example shown in Figure 20.9, there are three frames F_1 to F_3 which are taken in the same scene under different camera shifts. The object in the scene keeps the same locations, but its corresponding region in sensor area has been changed due to the camera being jiggled. The DIS algorithm should identify what the stationary objects are and determine the GMV $V_{1,2}$ and $V_{2,3}$. To keep the objects stabilized, a straightforward idea is to find the GMV and compensate for the relative location offsets among the successive frames. In this approach, the resolution of the stabilized images is smaller than the original effective sensor area and the adjustable range is quite limited.

Another comprehensive DIS system is integrating motion compensation and real-time digital zooming for providing higher flexible motion stabilization [27], [28]. As shown in

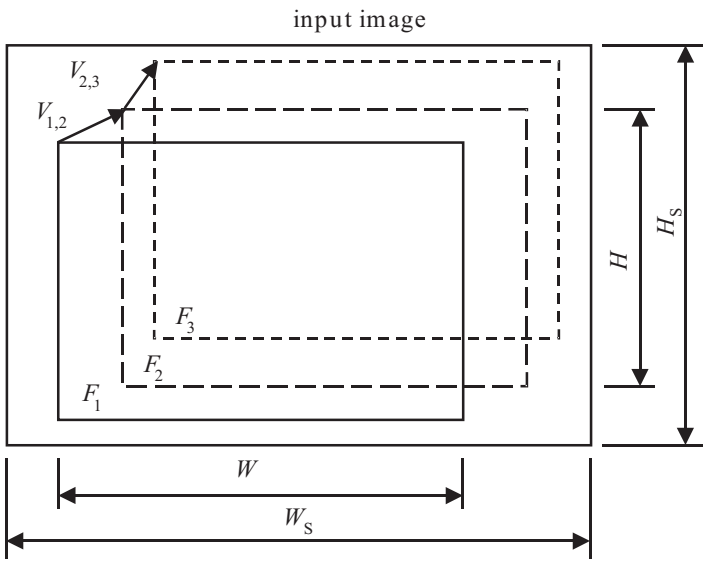


FIGURE 20.9

Camera vibration compensation by adaptively selecting sensor area.

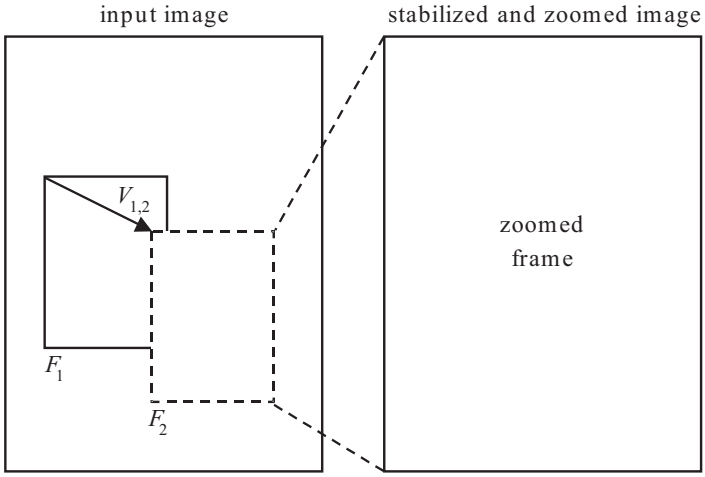


FIGURE 20.10

Incorporating image stabilization and digital zooming.

Figure 20.10, the window size of the cropped region is much smaller than the original raw image such that the adjustable range is larger than the one in Figure 20.10. The cropped region is zoomed to fit the resolution of the final video frame. But the video quality may become poor since the final video frames are processed by the digital zooming step.

Although the issues of motion estimation have been well studied in the field of video compression, estimating GMVs among successive video frames is not so trivial like video compression. A typical GMV estimation approach is first estimating motion vectors by several selected regions in the frame. The estimated vectors of these regions are called local motion vectors (LMVs) and are used to determine the GMVs. Unfortunately, the estimated motion vector for a region may not represent its exact motion vector caused by the camera vibration. Motion estimation in video compression searches the most similar regions between the two frames with minimal sum-of-absolute-difference (SAD). This approach is particularly bad for uniform color and low contrast regions which lack the textures or other obvious features. Any random noise in the region may dominate the motion estimation result and the offset between the exact and the estimated motion vector is unpredictable. As shown in Figure 20.11, another issue is that the movement of one or more objects is not caused by the camera's vibrations, but the estimated motion vectors for the regions includ-

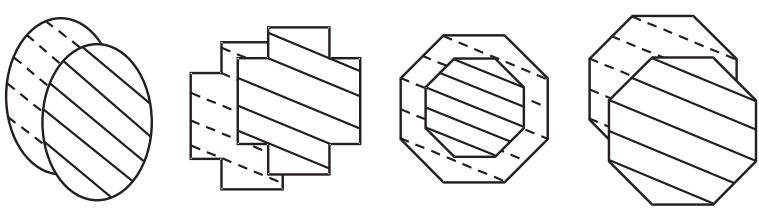


FIGURE 20.11

Some motion objects.

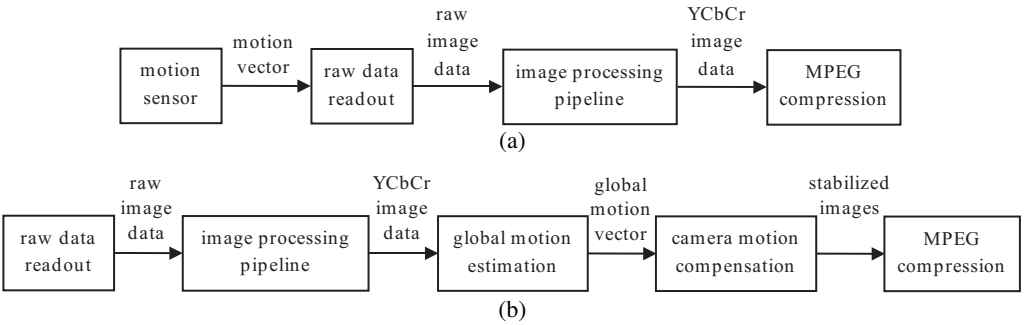


FIGURE 20.12

The system architecture of motion stabilization: (a) electronic image stabilization, (b) digital image stabilization.

ing these objects may affect the final judgment of GMVs. Finally, the processing algorithms should take into account that some pixels may not be exposed normally. For those overexposed or underexposed pixels, the image details are lost. The estimated motion vectors based on the regions with poor exposure are not suitable for GMV calculation.

In addition to the issues related to GMV estimation, directly applying the motion compensation based on the measured GMVs may not get acceptable stabilization results. This is because camera vibrations/fluctuations are composed of at least two different factors — the unwanted shaking and the intentional panning. Both shaking and panning are caused by the photographer, but panning is the desired motion to take a scene with wider view. Only camera shaking needs to be compensated in digital image stabilization. Several approaches to identifying shaking and panning have been proposed, and they will be reviewed in Sections 20.4 and 20.5.

20.3.2 System Architecture

The system architecture designs of electronic and digital image stabilization are shown in Figure 20.12a and Figure 20.12b, respectively. The EIS first uses a hardware component called gyro sensor to determine the camera motion vector or angular velocity. Then the corresponding registers in CCD/CMOS timing controller are set to control the starting readout position of raw image data in the sensor area. As shown in Figure 20.9, the effective area of an image sensor should be wider than the required area in the final image/video. The EIS system dynamically determines the start position in the effective area to compensate the camera vibration detected by the motion sensor. The captured raw data are dealt within the same image processing pipelines as the system without image stabilization. Using EIS, the embedded software design effort is much lower than DIS, because it only needs to set the parameters for the sensor readout timing based on the response of motion sensors. However, the extra cost as well as increased size may not be acceptable for miniature size digital cameras or camera modules built in mobile phones.

Unlike OIS and EIS systems that are hardware dependent, DIS system is a pure software solution which can be integrated into a camera's embedded software system with no extra component cost. As shown in Figure 20.12b, the raw image is directly captured from the

entire effective sensor area. Then, the data is processed by the image processing pipeline for color and tone reproduction. The processed data are typically converted to the form in *YCrCb* color space. To stabilize the captured images, the GMV representing the camera movement is estimated. Most GMV estimation approaches partition the entire sensor area into several local regions and estimate the local motion vectors individually.

Since several factors may affect the accuracy of the estimated local motion vectors (LMVs), directly integrating the estimated motion vectors does not necessarily reflect the exact camera vibrations. On the other hand, camera movement is composed of the factors of shaking and panning. It is necessary to filter out the motion estimation error and identify the component of camera shaking. The type of camera movement cannot be simply identified based on only two consecutive image frames. Instead of shifting the start position of the sensor readout area based on a measured GMV between two frames, the final compensating motion vector (CMV) is further refined by analyzing the GMVs among many successive images. After CMV is determined, the next step is to crop the sensor area according to CMV.

20.4 Global Motion Estimation of Digital Image Stabilization

The global motion estimation is to calculate the camera vibration or the active sensor area shift between two consecutive frames in a video sequence. The performance of a stabilization algorithm highly depends on the accuracy of the estimated GMVs. Thus, finding stable features for estimating GMVs is the most important task in a DIS system. In a practical image sequence, estimating GMV usually suffers from a few irregular factors: large moving objects, low contrast area, and repeated patterns. Such conditions degrade the overall performance of a DIS system. Finding a way to remove their effects is crucial for accurate GMV estimation.

A good DIS system should be able to remove both the translational and the rotational camera motions [29]. The problem of GMV estimation can be formulated as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \gamma \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - x_c \\ y - y_c \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} \tag{20.1}$$

where (x,y) and (x',y') are the positions of a pixel in the previous and the current frames, respectively. γ is the zooming factor, (x_c,y_c) is the estimated rotational center, θ is the rotational angle. a and b are the translational offsets in horizontal and vertical directions, respectively.

For reducing computational complexity, most of real-time DIS systems only perform two-dimensional image stabilization without considering the zooming condition. The zooming factor γ is simply set as 1 in this case. On the other hand, dealing with rotational camera motion is also a time consuming process. Only a few real-time DIS systems consider rotational camera vibrations. Under such a constraint, the corresponding parameters in Equation 20.1 are simply set as $\theta = 0$ and $(x_c,y_c) = (0,0)$, and only the translational offset vector $[a,b]^T$ is required to be calculated.

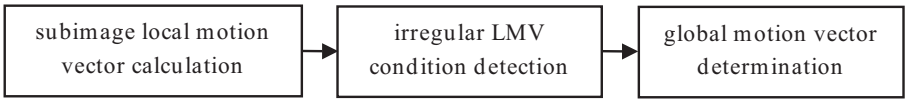


FIGURE 20.13

The processing flow of GMV estimation.

A typical processing flow of GMV estimation is shown in Figure 20.13. It first individually estimates the motion vectors in the selected regions. Depending on the types of DIS systems as well as the consideration of time complexity, there are several heuristic methods proposed to select the local regions for the evaluation of motion vectors. Then the irregular LMVs are detected by evaluating the reliability of the measured LMVs. The measured LMVs may not represent the exact motion vector of a region if this region contains large moving objects. The measured LMV in a region may also be dominated by random noise if the region lacks high contrast patterns or it is not exposed normally. These irregular LMVs should be bypassed in the final GMV estimation.

20.4.1 Local Motion Estimation

The objective of GMV estimation can be recognized as finding the shifts of the image background among a sequence of frames, since the background area should be kept in the same absolute locations while taking a video. When the camera shakes, image locations of the steady objects in the background are changed with a consistent motion vector. This problem can be dealt with by finding the motion vector of the background area in image frames. For solving the problem, GMV estimation is usually done based on the estimated LMVs of several regions that are selected heuristically. However, the accuracy of LMVs is always affected by moving objects, repeated patterns, and uniform color area in the background. The estimated LMV of a region may represent the exact GMV. But in many cases, a LMV may represent the motion vector of a moving object or even an incorrect vector in an irregular region.

The selection of the regions affects the accuracy of final GMV estimation and only the regions that belong to the image background should be used for GMV estimation. Several local region selection schemes have been proposed in the literature as shown in Figure 20.14. To compromise the time complexity such that the DIS can be accommodated with the computational capability in a practical camera system, the four corners or the top of the image are typically selected. These regions are recognized as the most stable ones for representing the background area. This heuristic comes from the observation that most photographers take video with aligning the objects-of-interest at the center of the scene, whereas the corner region usually belongs to the background of the scene. Based on the selected regions, the remaining issue is how to estimate the motion vectors for these selected regions [30], [31].

20.4.1.1 Sum-of-Absolute-Differences

To be compatible with standard video encoding process, several available DIS systems use macro block (MB), which typically contains 16×16 pixels, as the basic unit for the

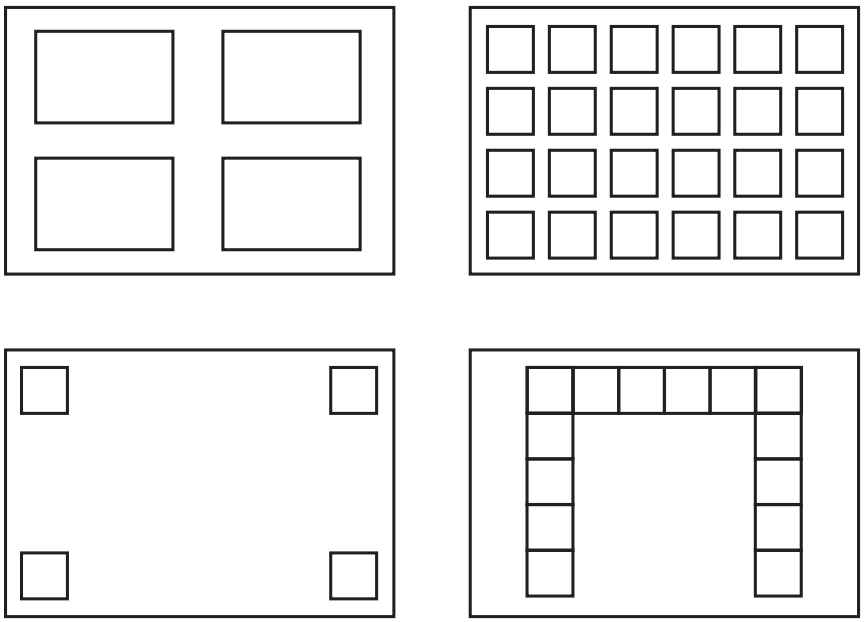


FIGURE 20.14

The selected regions for local motion vector estimation.

motion estimation [32], [33]. For each region shown in Figure 20.14, the sensor data in a region are further divided into several MBs. The estimated motion vector for a macro block is based on the evaluation of the sum of absolute difference (SAD):

$$SAD(i,j) = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} \mid C(x,y) - P(x+i, y+j) \mid \tag{20.2}$$

where M denotes the macro block size which is typically set as sixteen, C and P represent the current MB and the reference block, respectively. The candidate motion vector with smallest SAD value is chosen as the motion vector of the current MB. The measured motion vector is not necessary to be the exact location offset of a MB, and the local motion vector should be further screened by irregular region detection.

20.4.1.2 Representative Points

Directly applying SAD calculation for each pixel may suffer from the trouble of high computation complexity. If the camera hardware platform lacks an efficient SAD evaluation engine, the performance can hardly satisfy the real-time requirement. Instead of calculating all pixels in a region, a simplified version, which only selects a few representative points in a region, is popularly applied in consumer digital cameras [8], [27]. The system may adopt the SAD calculation from Equation 20.2 for only those representative points. The number of processed pixels is significantly reduced so that this approach is particularly useful for real-time DIS systems. Of course, the accuracy of the motion estimation process is also degraded a little.

20.4.1.3 Gray-Coded Bit-Plane Matching

Another idea to reduce the computation load is using the gray-coded bit-plane of image sequences [34], [35]. The approach can be realized by using only binary Boolean functions, which significantly reduce computational complexity. This approach first represents the input image with 2^K gray levels as follows:

$$F(x,y) = a_{K-1}2^{K-1} + a_{K-2}2^{K-2} + \cdots + a_1 2^1 + a_0 2^0 \tag{20.3}$$

where $a_k, 0 \leq k \leq K-1$, is either 0 or 1.

The next step is representing a video frame by K-bit gray code. The K-bit gray-code $g_{K-1} \cdots g_2 g_1 g_0$ can be computed as

$$g_{K-1} = a_{K-1} \tag{20.4}$$

$$g_k = a_k \oplus a_{k+1}, 0 \leq k \leq K-2 \tag{20.5}$$

where the symbol $\oplus$ represents exclusive OR (XOR) operation.

The LMVs are estimated based on the gray-coded bit planes of two successive frames. The correlation measure $Cor(i,j)$ is obtained as follows:

$$Cor(i,j) = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} g_k^t(x,y) \oplus g_k^{t-1}(x+i,\ y+j) \tag{20.6}$$

where $g_k^t(x,y)$ represents the t-th frame with the first k-th order gray-coded bit-plane.

The number of mismatch bits between the corresponding image regions in the previous and the current frame is used for the correlation measure. The complexity reduction is due to the fact that it only uses XOR operation for measuring the correlation instead of SAD operation.

Standing on a different point of view, this approach may not allow significant processing speed improvements. Most recent camera platforms have already included dedicated hardware engines for SAD evaluation. Calculating SAD is no longer a very time-consuming step from a system design point of view. However, the approach with bit-plane matching still gains the advantage of complexity reduction if the DIS is implemented in embedded software or calculations are realized by a specific hardware engine for correlation measurement.

20.4.1.4 Phase Correlations

The concept of LMV estimation based on phase correlation requires measuring the correlation in frequency domain in place of spatial domain [36]. The popular image data transform tools such as discrete Fourier transform (DFT) and discrete cosine transform (DCT) can be applied. Given a region $z(x,y)$ in the current image and a reference region $z'(x',y')$ in the previous image, the problem of motion estimation is formulated as finding a best correlation in their transform domain $Z(u,v)$ and $Z'(u',v')$, respectively. The approach may have good performance by adaptively selecting the useful features in the transform domain, but it also suffers from the problem of higher computational complexity. If the hardware platform of a digital camera already includes several hardware accelerators for DFT or DCT, this approach would be a good choice for estimating LMVs.

20.4.2 Irregular Local Motion Vector Detection

20.4.2.1 Deviation Analysis of Sum-of-Absolute-Differences

As described in the previous section, LMV estimation based on the evaluation of SAD values is a very efficient approach. For a MB, which has more textures or image details inside it, the distribution of SAD values in all search steps should be diversified. On the contrary, the calculated SAD values would be very close while estimating the LMV for a low contrast region. The random noise may dominate the estimated motion vector and the resulting offset is unpredictable. Hence the estimated LMV in such a region is not reliable. To identify whether the region is a low contrast one or not, a straightforward way is to simply analyze the deviation of SAD data passing from all search steps. This approach gains the benefit of directly utilizing the intermediate data in the motion estimation process without extra time-consuming operations.

Based on the concept described above, the evaluation of the motion estimation reliability for a MB can be formulated as follows:

$$s = \frac{1}{N-1} \sum_{\substack{all\ (i,j)\ in\ the \\ search\ window}} (SAD(i,j) - \hat{a})^2 \tag{20.7}$$

where s denotes the motion estimation reliability, N is the total number of search steps in motion estimation for a MB, and

$$\hat{a} = \sum_{\substack{all\ (i,j)\ in\ the \\ search\ window}} SAD(i,j)\ /\ N \tag{20.8}$$

denotes the average SAD values in all search steps for the macro block [32].

A region which has lower motion estimation reliability is recognized as a low contrast region. The estimated LMVs corresponding to these regions are called irregular LMVs which should be bypassed in GMV estimation. In this way, the problem of unstable GMV caused by uniform color regions will be reduced drastically.

20.4.2.2 Low-Pass Filtering

The deviation analysis of SAD values can identify some types of irregular conditions without referring to the information from other MBs or regions. But it is still possible to get wrong judgment. The accuracy of motion estimation can be further improved by applying a low-pass filter on the estimated LMVs of neighboring regions or selecting the median vector among the motion vectors of the MBs inside a region. The improvement comes from the reason that the LMV of an individual MB may be affected by random noise so that a few LMVs can be recognized as noise vectors. Popular noise filters would be helpful for coping with this kind of problem. This approach can also partially remedy the problem of some moving objects included in the scene. Note that the horizontal and vertical motion offsets in each region can be handled separately. It is not necessary to map the horizontal and vertical motion offsets onto the same LMV for a region.

20.4.2.3 Histogram Analysis

Another comprehensive approach for irregular LMV detection is based on histogram analysis of LMVs [37]. It first constructs the histogram H of the LMVs. Then a 5×5

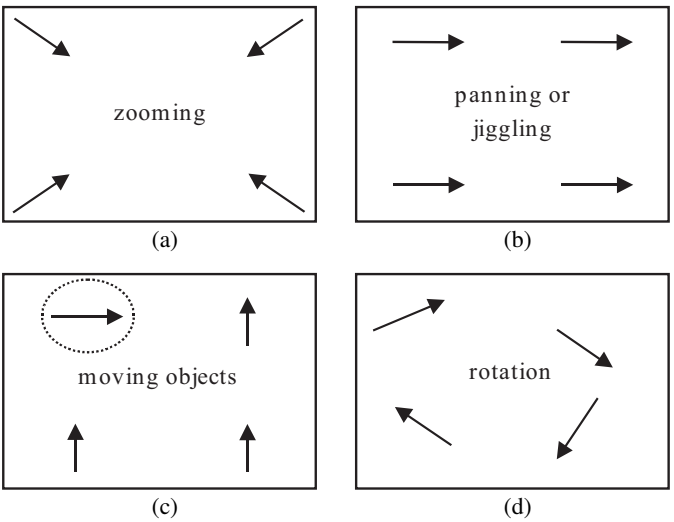


FIGURE 20.15

Possible types of LMV combinations.

lowpass filter is applied to H to enhance the noise immunity as follows:

$$H'(x,y) = \sum_{i=-2}^{2} \sum_{j=-2}^{2} H(x+i, y+j) \tag{20.9}$$

The vector which has the maximum statistic value p of the filtered histogram H' is selected as the corresponding LMV for a region as in the following equation:

$$p = arg \quad max_{(x,y)}\{H'(x,y)\} \tag{20.10}$$

20.4.3 Global Motion Vector Determination

Global motion vector (GMV) represents the camera vibration offset between two successive frames. This GMV is determined based on the estimated LMVs as well as their reliability. The unreliable LMVs are corresponding to irregular regions so that they should be bypassed or given lower weights in GMV determination [38]. A simple heuristic approach to it is just selecting and averaging a few LMVs whose motion vector stability is good enough or selecting the median vector among them as the GMV of an image frame.

Figure 20.15 shows several possible combinations of the LMVs. In Figure 20.15a, the LMVs do not have a consistent direction; however, they direct to the center point. In practical camera design, this situation can be easily detected without complicated GMV estimation. This is because the zoom lens motor is controlled by embedded software. The video recording module can always get the correct status of the zoom motor. Hence detecting this situation is not a big problem for a DIS system. In Figure 20.15b, all regions have similar LMVs. Thus the GMV can be simply determined by averaging the LMVs.

Determining the GMV in the situation of Figure 20.15c is not so trivial like the previous two cases. Some heuristic rules should be applied to identify which regions include some

moving objects. In this case, most regions have similar directions but some others do not. The GMV is estimated according to two rules. i) If only a few regions have different LMVs from the others, these regions may be corresponding to moving objects. Hence the GMV should be determined without including the LMVs into the calculation. ii) The GMV estimation should refer to the reliability analysis passing from irregular LMV detection. If the reliability of a LMV is low, it means the corresponding region may be located in low contrast area or repeated patterns. The estimated LMVs should be bypassed or given a lower weight in GMV calculation. The problem of calculating GMV can be formulated as follows:

$$\mathbf{M_G} = \chi \sum_{i=0}^{v} \mathbf{M_L}(i) \times w_i \tag{20.11}$$

$$\chi = \frac{1}{\sum_{i=0}^{v} w_i} \tag{20.12}$$

where $\mathbf{M_G}$ represents the estimated GMV, $\mathbf{M_L}(i)$ is the LMV of the i-th region, and w_i is the weighting factor for region i which is inversely proportional to the LMV reliability of the region i. Note that w_i is set to zero if the region i includes some moving objects.

Handling the rotational type of camera fluctuation is much more complex than handing the translational motion [39], [40]. As shown in Figure 20.15d the motion vectors in different regions are inconsistent. As stated in Equation 20.1, we assume the zoom factor γ is set as 1 for simplifying the discussion. Given m estimated motion vectors as well as their original representative points, the problem of GMV estimation for dealing with both translational and rotational motions can be formulated as follows:

Let $\mathbf{p}_i = [\, x_i,\ y_i \,]^T$ and $\mathbf{p}'_i = [\, x'_i,\ y'_i \,]^T$ be the estimated coordinates of the representative point of the region i in the previous and current frames, respectively. We assume the entire image frame is partitioned into m regions. The problem of GMV estimation becomes an optimization problem. The goal of the optimization is to find the parameters θ, x_c, y_x, a, and b in

$$\hat{\mathbf{p}}_i = \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i - x_c \\ y_i - y_c \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}, \quad 1 \le i \le m \tag{20.13}$$

such that the error function

$$\Delta = \sum_{i=1}^{m} \|\mathbf{p}'_i - \hat{\mathbf{p}}_i\|^2 = \sum_{i=1}^{m} [(x'_i - \hat{x}_i)^2 + (y'_i - \hat{y}_i)^2] \tag{20.14}$$

is minimized.

20.5 Motion Compensation

Directly compensating the camera vibration according to the estimated GMVs may not get acceptable stabilization results in most image sequences. A typical DIS system incorporates the irregular GMV detection and motion vector stability analysis to prevent the unreliable estimation results from affecting the final GMV estimation. However, a GMV

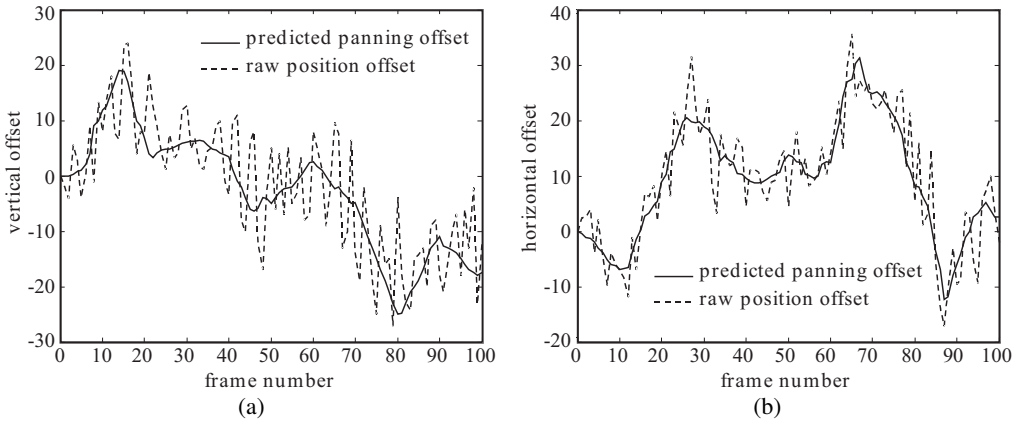


FIGURE 20.16

The estimated global motion vectors (GMVs) and the predicted panning offsets in vertical and horizontal directions.

is typically estimated using two frames. It lacks a complete view of camera fluctuations among multiple video frames. The GMV estimation based on two successive frames cannot efficiently differentiate panning and shaking (jiggling), the two types of camera vibration. Besides, the estimated GMV is still possibly an error vector. Hence GMV should not be directly applied to final motion compensation for a video frame.

20.5.1 Compensation Motion Vector Determination

As shown in Figure 20.16, the estimated GMVs can be plotted on two motion trajectory charts in horizontal and vertical directions. Amongst panning, shaking and estimation errors, only the shaking factor in camera vibration needs to be compensated. According to the amplitude as well as the frequency in the trajectories, the motion compensation algorithm should determine the trajectory of the desired camera panning component and then it compensates only for the unwanted shaking component.

To calculate panning trajectory, many available approaches try to eliminate unwanted higher frequency vibrations from the detected GMV trajectories. The low frequency component is usually recognized as the camera panning movement. However, the highest frequency component may not always correspond to the camera shaking, but it is possibly an error factor while estimating GMV. Depending on the accuracy of the estimated GMVs, DIS systems may adopt different strategies to smooth the trajectory for calculating camera panning and determine final motion compensation vectors.

20.5.1.1 Moving Average

A straightforward approach to determining the trajectory of camera panning motion is using moving average. For real-time applications, most DIS systems only refer to the GMVs of the previous frames and the current frame. A good motion correction should first generate a panning vector $\mathbf{M_P}$ that resembles the intentional camera movement and removes the shaking components. A typical panning vector calculation based on moving

average is given by

$$\mathbf{M_P}(t) = k\mathbf{M_P}(t-1) + \alpha\mathbf{M_G}(t), \;\; 0 < k < 1 \;\; \textit{and} \;\; 0 \leq \alpha \leq 1 \tag{20.15}$$

The parameters k and α can be regarded as the smoothing and tracking factors, respectively. The increase in k causes smoother panning trajectory, while the increase in α results in the increase in tracking capabilities. After panning vector has been calculated, the final compensation vector $\mathbf{M_C}$ can be determined as follows:

$$\mathbf{M_C}(t) = \mathbf{M_G}(t) - \mathbf{M_P}(t) \tag{20.16}$$

20.5.1.2 Kalman Filter

Another systematic approach to predicting camera panning is based on Kalman filtering [41], [42]. The Kalman filter provides an estimate to the state $\mathbf{x}_t$ of a discrete-time process that is governed by the linear stochastic difference equations:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}_{t-1} \tag{20.17}$$

$$\mathbf{m}_t = \mathbf{H}\mathbf{x}_t + \mathbf{n}_t \tag{20.18}$$

where $\mathbf{m}_t$ is a measurement of the frame t, the variables $\mathbf{w}_t$ and $\mathbf{n}_t$ represent the process and measurement noise, respectively.

The state equation 20.17, and observation equation 20.18, are typically constructed with constant velocity motion model (CVMM) [43], [44] when applying Kalman filter on DIS systems. The state equations of the CVMM are formulated as follows:

$$\mathbf{x}_t = \begin{bmatrix} p_t^x \\ p_t^y \\ v_t^x \\ v_t^y \end{bmatrix} = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix} \begin{bmatrix} p_{t-1}^x \\ p_{t-1}^y \\ v_{t-1}^x \\ v_{t-1}^y \end{bmatrix} + \mathbf{w} \tag{20.19}$$

$$\mathbf{m}_t = \begin{bmatrix} m_t^x \\ m_t^y \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \end{bmatrix} \begin{bmatrix} p_t^x \\ p_t^y \\ v_t^x \\ v_t^y \end{bmatrix} + \mathbf{n} \tag{20.20}$$

where four state variables p_t^x, p_t^y, v_t^x, and v_t^y are used to represent the horizontal position, vertical position, horizontal velocity, and vertical velocity of the frame t, respectively. The measurement frame position is denoted as $\mathbf{m}_t = [m_x^t, m_y^t]^T$, which constitutes the predicted panning offset.

The remaining task is using Kalman filter to estimate the state variables by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of measurements. The equations for this filter then fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback that incorporates a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate. The detailed theory as well as equations for Kalman filter can be found in Reference [42].

FIGURE 20.17

Cropping images. © 2006 IEEE

20.5.2 Image Warping

The final stage of an image stabilization system is the image warping. With the determined compensation vector, the process of image warping stabilizes the image frame by adjusting the location offset of the sensor active area, filling up the missing image area, and deblurring the images.

As shown in Figure 20.17, the four original frames are not kept still due to camera fluctuations. In these frames, a girl is passing through the front of the camera. If the motion compensation only considers the translational type of camera vibrations, the DIS just simply changes the location of the cropped window. The relative positions of the objects belonging to the image background are kept unchanged in the final frames. This compensation method is the most popular one in consumer digital cameras due to the consideration of computational complexity. Its major drawback is the resolution of the resulting frames has been reduced since parts of image areas in the boundaries are trimmed. In addition to the drawback of limited adjustable margins of sensor area, such a simple approach cannot handle severe camera shake as well as rotational type of camera vibration.

A sophisticated image stabilization system may consider both translational and rotational motion models, and apply image inpainting as well as deblurring algorithm to repair the corrupted frames. The unstabilized images are first translated and rotated to let the locations of the objects in the scene keep still after stabilization as shown in Figure 20.18. The stabilized frames suffer from the trouble of some areas being trimmed. Filling up those

FIGURE 20.18

Rotational motions.

missing image areas is called image or video inpainting. To generate a full-frame video with good visual quality, several robust video inpainting approaches have been proposed in the literature [45], [46], [47], [48], [49]. The video inpainting propagates local motion data in the known areas to predict the stimulus values for those missing pixels. The approach first determines the optical flow field in the missing areas. The motion value of a pixel in a missing area is calculated based on the known motion values of its neighborhoods. Then, the motion values locally guide warping to generate a smooth stitched video. If some missing pixels still exist in the stitched frames based on the previous frame, the algorithm can also refer to other frames to fill up the missing pixels. Or those missing pixels are simply interpolated by their neighboring pixels in the same frame.

In addition to filling up the missing pixels with image inpainting, another issue for compensating camera motion is how to reduce image blurriness, which is also called motion deblur. Motion blur is caused by a moving scene point that spreads out several pixel locations during the exposure period of the sensor. Image deconvolution based on an estimated point spread function (PSF) is the most popular approach to image deblur, but an accurate PSF is hard to obtain in real situations. Another good way to solve the problem is first evaluating the relative blurriness and then transferring sharper image pixels from neighboring frames to the corresponding blurry image pixels. This approach relies on accurate blurriness measure. Detailed descriptions of related algorithms can be found in References [45], [50], and [51].

20.6 Conclusion

This chapter surveyed popular image stabilization techniques. As the presented overview indicates, the image stabilization techniques constitute an indispensable tool in portable

imaging devices such as consumer digital cameras. To remove undesired hand shakes or jiggles and produce stable visual data, camera manufacturers rely on optical, electronic, and digital image stabilization. Optical image stabilization gains the advantage of stabilizing both still image and video sequences without embedded software efforts. But the production cost of this approach is definitely higher than that of the other two stabilization approaches. Digital image stabilization is particularly useful for low cost and miniature size cameras. Typically, digital image stabilization is completed by performing local motion estimation, global motion estimation and motion compensation using estimated motion vectors. With advances in imaging software, this technique has been extensively studied and promising results have been reported. Electronic image stabilization is quite similar to digital image stabilization except it uses motion sensors to detect camera vibration instead of using software approaches.

Acknowledgment

Figure 20.6 is reprinted from Reference [32] with the permission of IEEE.

References

[1] A. Engelsberg and G. Schmidt, "A comparative review of digital image stabilising algorithms for mobile video communications," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 591–597, August 1999.

[2] F. Vella, A. Castorina, M. Mancuso, and G. Messina, "Digital image stabilization by adaptive block motion vectors filtering," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 3, pp. 796–801, August 2002.

[3] H. Okuda, M. Hashimoto, K. Sumi, and S.I. Kaneko, "Optimum motion estimation algorithm for fast and robust digital image stabilization," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 1, pp. 276–280, February 2006.

[4] A.A. Yeni and S. Erturk, "Fast digital image stabilization using one bit transform based sub-image motion estimation," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 917–921, August 2005.

[5] O. Urhan and S. Erturk, "Single sub-image matching based low complexity motion estimation for digital image stabilization using constrained one-bit transform," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 4, pp. 1275–1279, November 2006.

[6] J.S. Jin, Z. Zhu, and G. Xu, "A stable vision system for moving vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 1, pp. 32–39, March 2000.

[7] S.C. Hsu, S.F. Liang, and C.T. Lin, "A robust digital image stabilization technique based on inverse triangle method and background detection," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 335–345, May 2005.

[8] S.C. Hsu, S.F. Liang, K.W. Fan, and C.T. Lin, "A robust in-car digital image stabilization

technique," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 2, pp. 234–247, March 2007.

[9] C.W. Chiu, C.P. Chao, and D.Y. Wu, "Optimal design of magnetically actuated optical image stabilizer mechanism for cameras in mobile phones via genetic algorithm," *IEEE Transactions on Magnetics*, vol. 43, no. 6, pp. 2582–2584, June 2007.

[10] K. Sato, S. Ishizuka, A. Nikami, and M. Sato, "Control techniques for optical image stabilizing system," *IEEE Transactions on Consumer Electronics*, vol. 39, no. 3, pp. 461–466, August 1993.

[11] M. Oshima, T. Hayash, S. Fujioka, T. lnaji, H. Mitani, J. Kajino, K. Ikeda, and K. Komoda, "VHS camcorder with electronic image stabilizer," *IEEE Transactions on Consumer Electronics*, vol. 35, no. 4, pp. 749–758, November 1989.

[12] T. Kinugasa, N. Yamamoto, H. Komatsu, S. Takase, and T. Imaide, "Electronic image stabilizer for video camera use," *IEEE Transactions on Consumer Electronics*, vol. 36, no. 4, pp. 520–525, August 1990.

[13] T. Morofuji, "Vibration correction apparatus and optical device," U.S. Patent 6 704 502, March 2004.

[14] Canon, "What is vari-angle prism?" Available online: www.canon.com/bctv/faq/vari.html.

[15] Canon, "What is optical image stabilizer?" Available online: http://www.canon.com/bctv/faq/optis.html.

[16] Nikon, "Vibration reduction (VR) technology." Available online: http://nikonimaging.com/global/technology/scene/16/.

[17] Panasonic, "Mega optical image stabilizer." Available online: http://www2.panasonic.com/webapp/wcs/stores/servlet/MegaOISExplained?storeId=15001.

[18] Konica-Minolta, "Anti-shake technology." Available online: http://ca.konicaminolta.com/products/consumer/digital_camera/slr/dynax-7d/02.html.

[19] Olympus, "Image stabilizer for enhanced shooting confidence." Available online: http://www.olympus-europa.com/consumer/dslr_16742.htm.

[20] Pentax, "Shake reduction technology." Available online: http://www.pentaximaging.com/files/scms_docs/shake_reduction_fact_sheet.pdf.

[21] WebSiteOptimization.com, "Multimedia: Use image stabilization." Available online: http://www.websiteoptimization.com/speed/tweak/stabilizer/.

[22] B. Golik and D. Wueller, "Measurement method for image stabilizing system," in *Proceedings of the IS&T / SPIE Symposium on Electronic Imaging*, San Jose, CA, USA, January 2007, pp. 65020O:1–10.

[23] K. Noguchi, "Vibration correcting device, lens barrel, and optical device," U.S. Patent 6 985 176, January 2006.

[24] T. Kitazawa, T. Kitaguchi, H. Shimizu, M. Katoh, Y. Sata, S. Sasaki, and A. Etoh, "Imaging apparatus, and method and device for shake correction in imaging apparatus," U.S. Patent 6 940 542, September 2005.

[25] S. Wada, A. Kosaka, Y. Hara, and J. Tanii, "Lens barrel with built-in blur correction mechanism," U.S. Patent 6 701 071, March 2004.

[26] H. Kawahara and T. Kudo, "Image sensing apparatus featured with vibration prevention function," U.S. Patent 6 388 705, March 2002.

[27] K. Uomori, A. Morimura, H. Ishii, T. Sakaguchi, and Y. Kitamura, "Automatic image stabilizing system by full-digital signal processor," *IEEE Transactions on Consumer Electronics*, vol. 36, no. 3, pp. 510–519, August 1990.

[28] A. Amanatiadis and I. Andreadis, "An integrated dynamic image stabilizer applied to zooming systems," in *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, Ottawa, ON, Canada, May 2007, pp. 1–5.

[29] J.Y. Chang, W.F. Hu, M.H. Cheng, and B.S. Chang, "Digital image translational and rotational motion stabilization using optical flow technique," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 1, pp. 108–115, February 2002.

[30] J.K. Paik, Y.C. Park, and D.W. Kim, "An adaptive motion decision system for digital image stabilizer based on edge pattern matching," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 3, pp. 607–616, August 1992.

[31] J.K. Paik, Y.C. Park, and S.W. Park, "An edge detection approach to digital image stabilization based on tri-state," *IEEE Transactions on Consumer Electronics*, vol. 37, no. 3, pp. 521–530, August 1991.

[32] W.C. Kao, S.H. Chen, and P.Y. Hsiao, "Real-time image stabilization for digital video cameras," in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems*, Singapore, December 2006, pp. 1651–1654.

[33] M.K. Kim, E. Kim, D. Shim, S.I. Jang, G. Kim, and W. Kim, "An efficient global motion characterization method for image processing applications," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 4, pp. 1010–1018, November 1997.

[34] S.J. Ko, S.H. Lee, and K.H. Lee, "Digital image stabilizing algorithms based on bit-plane matching," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 617–622, August 1998.

[35] S.J. Ko, S.H. Lee, S.W. Jeon, and E.S. Kang, "Fast digital image stabilizer based on gray-coded bit-plane matching," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 598–603, August 1999.

[36] S. Erturk, "Digital image stabilization with sub-image phase correlation based global motion estimation," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1320–1325, November 2003.

[37] H.H. Chen, C.K. Liang, Y. C. Peng, and H.A. Chang, "Integration of digital stabilizer with video codec for digital video cameras," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 7, pp. 801–813, July 2007.

[38] Y. Equsa, H. Akahori, A. Morimura, and N. Wakami, "An application of fuzzy set theory for an electronic video camera image stabilizer," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 3, pp. 351–356, August 1995.

[39] L. Xu and X. Lin, "Digital image stabilization based on circular block matching," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 566–574, May 2006.

[40] H.K. Seok, H.K. Kwak, and J. Lyou, "A rotational motion estimation method for digital image stabilization," in *Proceedings of the IEEE TENCON*, Melbourne, Australia, November 2005, pp. 1–6.

[41] R.E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the (ASME) — Journal of Basic Engineering*, vol. 82, no. D, pp. 35–45, December 1960.

[42] G. Welch and G. Bishop, "An introduction to the Kalman filter," in *SIGGARPH2001 Course*, 2001.

[43] A. Litvin, J. Konrad, and W.C. Karl, "Probabilistic video stabilization using Kalman filtering and mosaicking," in *Proceedings of the IS&T / SPIE Symposium on Electronic Imaging*, San Jose, CA, USA, January 2003, pp. 663–674.

[44] S. Erturk, "Real-time digital images stabilization using Kalman filters," *Real-Time Imaging*, vol. 8, no. 4, pp. 317–328, August 2002.

[45] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Transactions on Pattern Anaysis and Machine Intelligence*, vol. 28, no. 7, pp. 1150–1163, July 2006.

[46] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, New Orleans, LA, USA, July 2000, pp. 417–424.

[47] Y. Wexler, E. Shechtman, and M. Irani, "Space-time video completion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington, D.C., USA, June / July 2004, vol. I, pp. 120–127.

[48] J. Jia, T.P. Wu, Y.W. Tai, and C.K. Tang, "Video repairing: Inference of foreground and background under severe occlusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, June / July 2004, vol. I, pp. 364–371.

[49] Q. Luo and T.M. Khoshgoftaar, "An empirical study on estimating motions in video stabilization," in *Proceedings of the IEEE International Conference on Information Reuse and Integration*, Las Vegas, NV, USA, August 2007, pp. 360–366.

[50] M.B. Ezra and S.K. Nayar, "Motion-based motion deblurring," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 689–698, June 2004.

[51] A. Rav-Acha and S. Peleg, "Restoration of multiple images with motion blur in different directions," in *Proceedings of the IEEE Workshop on Applications of Computer Vision*, Palm Springs, CA, USA, December 2000, pp. 22–28.

Index

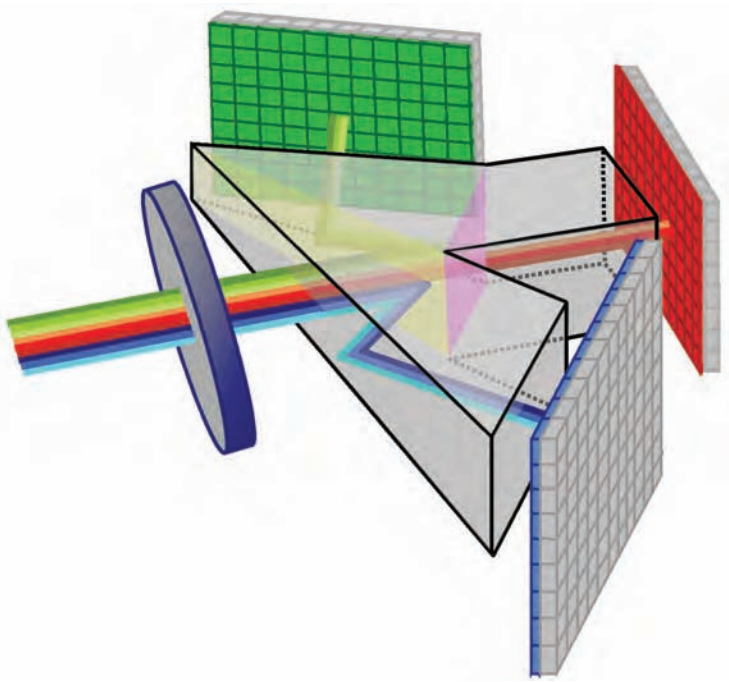


FIGURE 1.1

Three-sensor digital camera architecture.

(a) (b)

FIGURE 1.2

Three-sensor imaging. (a) Registration of three grayscale images on the left, top and bottom of the figure which were acquired using sensors with red, green and blue color filters, respectively. (b) Final full-color image achieved by registering three sensor images.

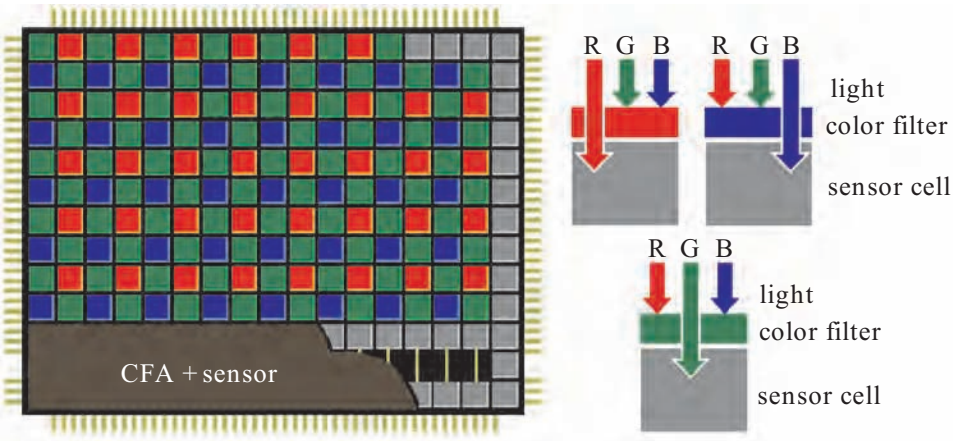


FIGURE 1.3

(left) Image sensor covered by a Bayer color filter array. (right) The concept of acquiring the visual information using color filters.

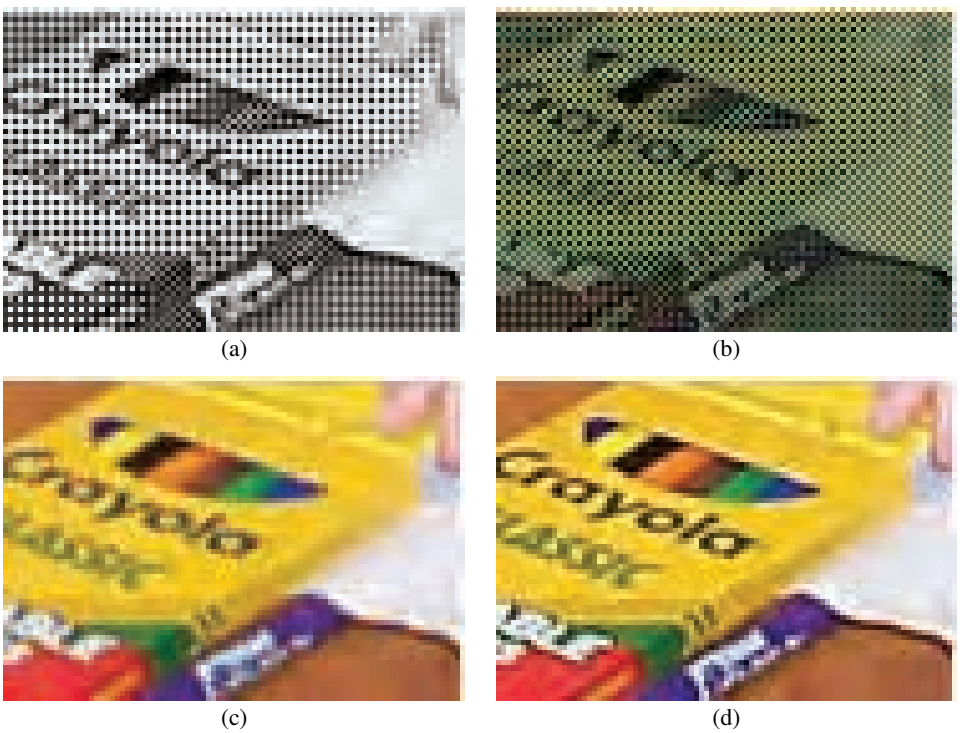
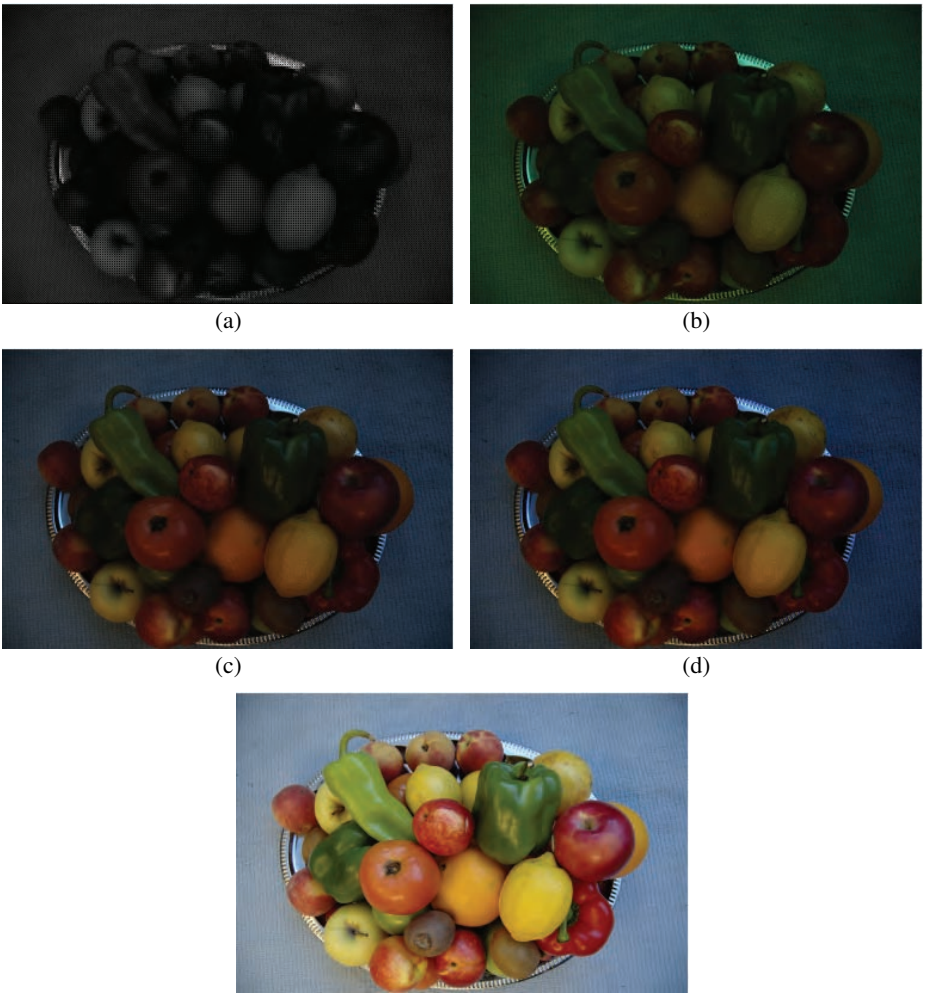


FIGURE 1.4

Single-sensor imaging: (a) grayscale mosaic image, (b) color version of the mosaic image, (c) demosaicked full-color image, and (d) postprocessed demosaicked image with improved visual quality.



(a) (b) (c) (d) (e)

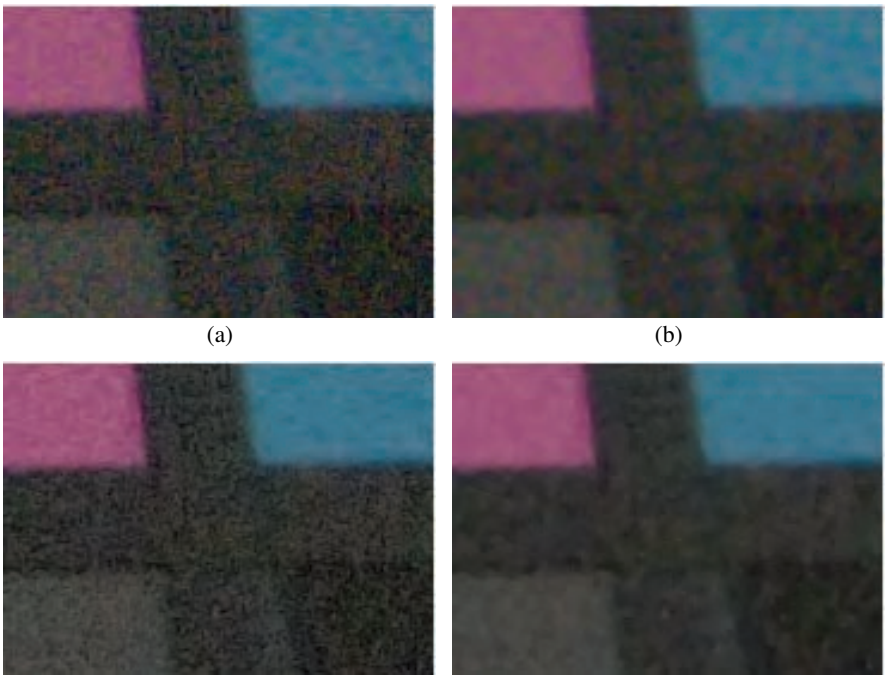
FIGURE 1.5

Images stored at different stages of the single-sensor color imaging pipeline: (a) mosaic CFA image, (b) demosaicked image, (c) white-balanced image, (d) color-corrected image, and (e) tone / scale-rendered image.

(a) (b) (c)

FIGURE 1.6

Image sharpening. Finished images generated by the pipeline depicted in Figure 1.5: (a) original pipeline, i.e., *no sharpening*, (b) original pipeline with added *image sharpening after demosaicking*, and (c) original pipeline with added *image sharpening before demosaicking*.

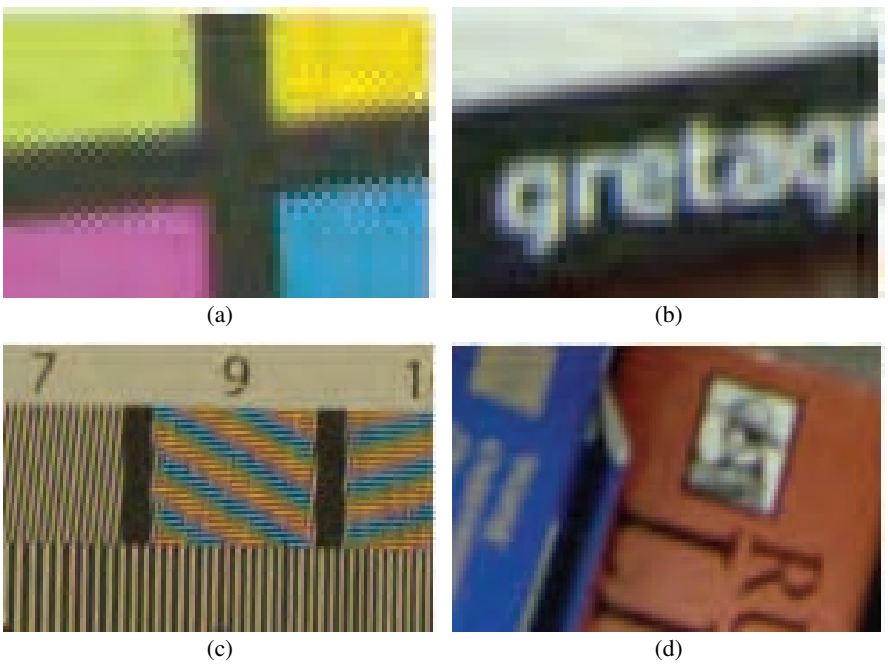


(a) (b)

(c) (d)

FIGURE 1.7

Cropped parts of a color checker image captured with ISO 1600 setting: (a) captured noisy image, (b) luminance noise suppression, (c) color noise suppression, and (d) both luminance and color noise suppression.



(a) (b)

(c) (d)

FIGURE 1.8

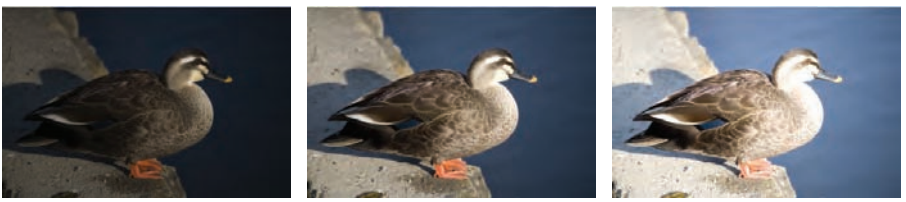
Typical demosaicking defects: (a) zipper effects, (b) color shifts, (c) aliasing artifacts, and (d) blurring effects.

(a) (b)

(c) (d)

FIGURE 1.9

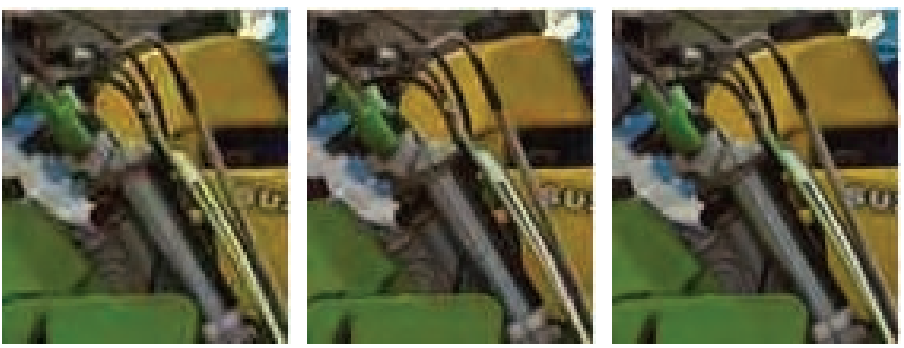
Coloration shifts due to incorrect white balance settings: (a) cool appearance, (b) warm appearance, (c) grayish appearance, (d) saturation effects. See also Figure 1.5e corresponding to an *as-shot* white balance setting.



(a) (b) (c)

FIGURE 1.10

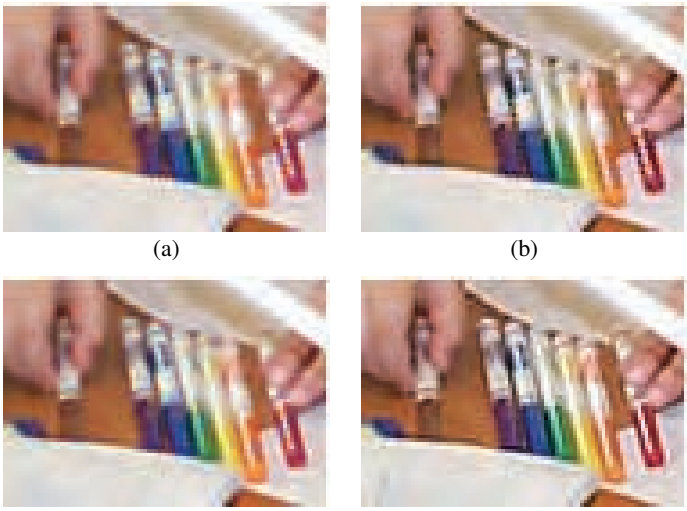
Influence of exposure settings on image quality: (a) underexposure, (b) normal exposure, and (c) overexposure.



(a) (b) (c)

FIGURE 1.11

Influence of data compression on image quality for the same demosaicking method: (a) lossy compression of a full-color image, (b) lossy compression of a CFA image, and (c) lossless compression.

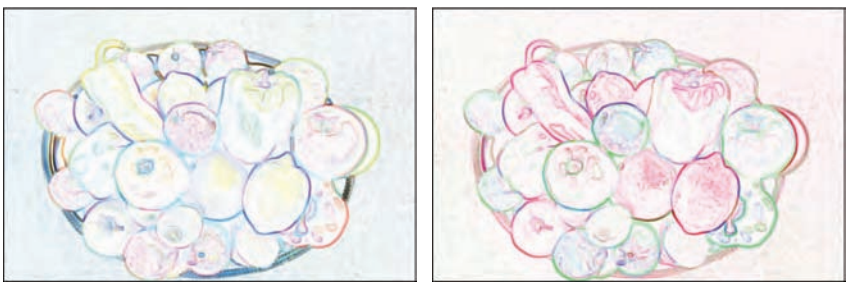


(a) (b)

(c) (d)

FIGURE 1.12

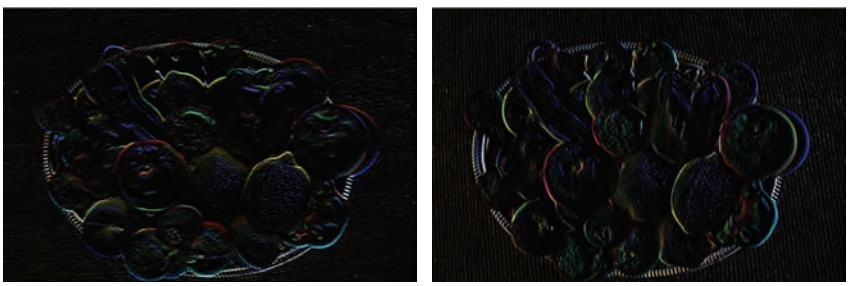
Different visual quality of images demosaicked using: (a) spatial characteristics, (b) spatial and spectral characteristics, (c) spatial and structural characteristics, and (d) spatial, structural, and spectral characteristics.



(a) (b)

FIGURE 1.13

Correlation characteristics of the image shown in Figure 1.5e. Brighter values correspond to higher correlations. (a) RGB values denote *spatial correlations* in the red, green and blue channel of the original image, respectively. (b) RGB values denote *spectral correlations* between red and green, green and blue, and red and blue channels of the original image, respectively.



(a) (b)

FIGURE 1.14

Structural content of the image shown in Figure 1.5e: (a) horizontal edges, and (b) vertical edges.

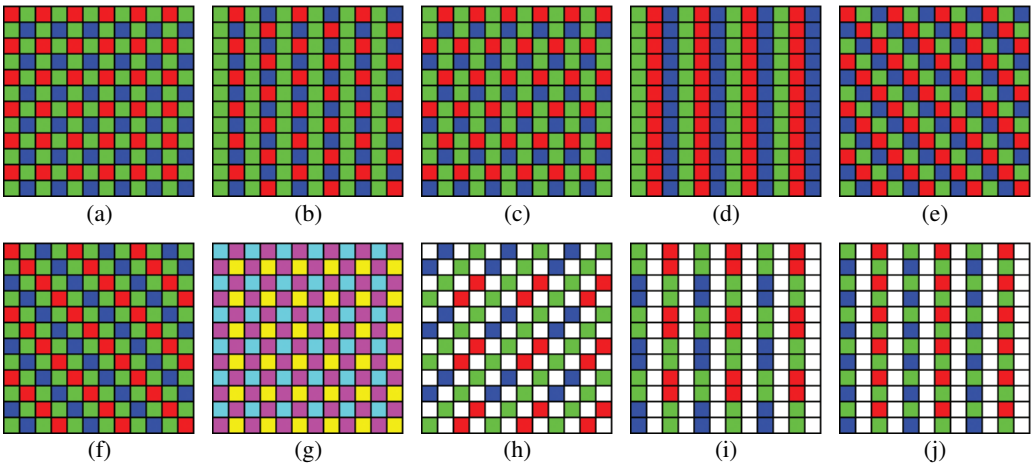


FIGURE 5.2

Examples of existing CFAs: (a) Bayer [4], (b) Yamanaka [5], (c) Lukac [7], (d) vertical stripes [7], (e) diagonal stripes [7], (f) modified Bayer [7], (g) cyan-magenta-yellow, (h) Kodak I [16], (i) Kodak II [16], (j) Kodak III [16].

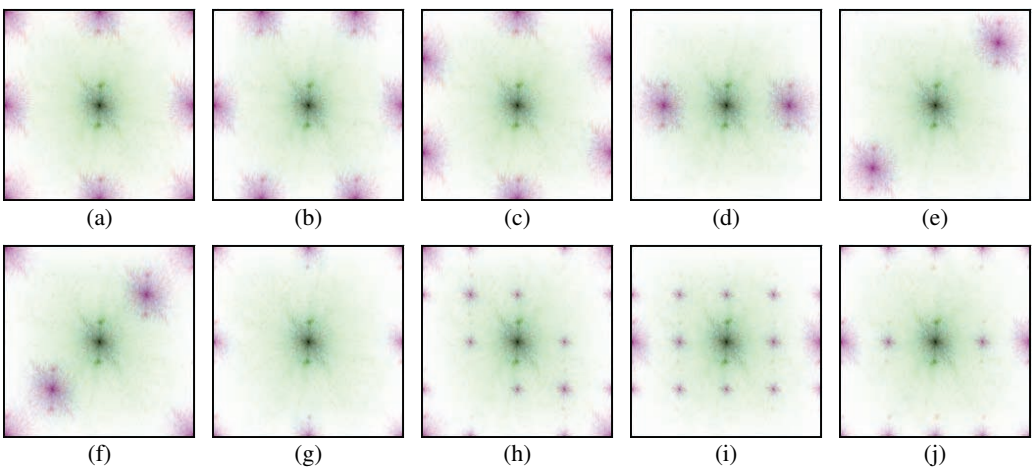
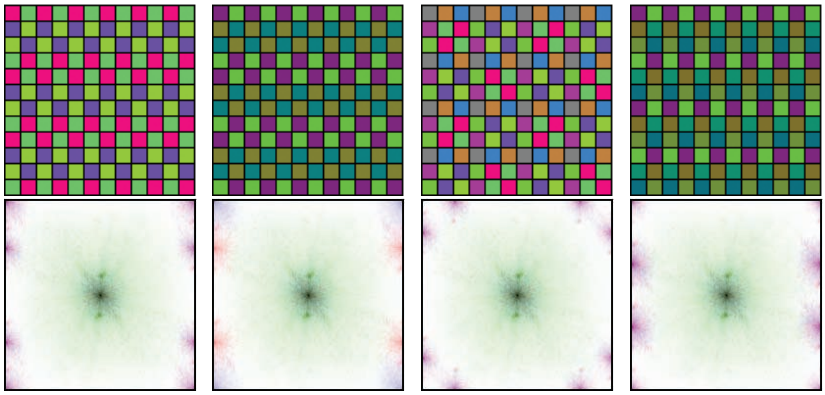


FIGURE 5.3

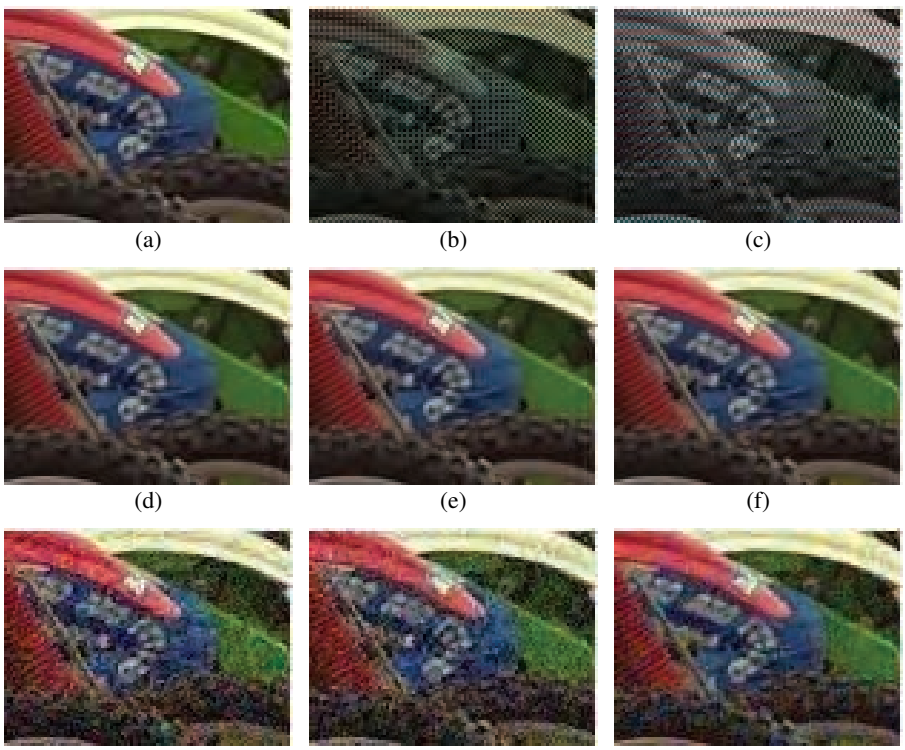
Log-magnitude spectra of a typical color image (i.e., image *flower* here) sampled with CFAs corresponding to Figure 5.2. Color coding is used to distinguish different components, with the $x_g(\boldsymbol{n})$ component shown in green, $x_\alpha(\boldsymbol{n}) = x_r(\boldsymbol{n}) - x_g(\boldsymbol{n})$ in red, and $x_\beta(\boldsymbol{n}) = x_b(\boldsymbol{n}) - x_g(\boldsymbol{n})$ in blue. Individual spectra correspond to: (a) Bayer [4], (b) Yamanaka [5], (c) Lukac [7], (d) vertical stripes [7], (e) diagonal stripes [7], (f) modified Bayer [7], (g) cyan-magenta-yellow, (h) Kodak I [16], (i) Kodak II [16], (j) Kodak III [16].



(a) (b) (c) (d)

FIGURE 5.5

Proposed CFAs (top) and resultant log-magnitude spectra (bottom) of a typical color image (i.e., image *flower* here). Color coding is used as in Figure 5.3 to distinguish components X_α, X_g, and X_β. Subfigures correspond to: (a) *pattern A*, (b) *pattern B*, (c) *pattern C*, and (d) *pattern D*.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

FIGURE 5.7

Bike image sensor data (top row), with nonlinear and linear reconstruction methods shown for the case of clean (middle row) and noisy (bottom row) sensor data. Individual images correspond to: (a) original image, (b) Bayer CFA sampling, (c) *pattern A* sampling, (d) nonlinear Bayer reconstruction, (e) linear Bayer reconstruction, (f) linear *pattern A* reconstruction, (g) noisy nonlinear Bayer reconstruction, (h) noisy linear Bayer reconstruction, and (i) noisy *pattern A* linear reconstruction.

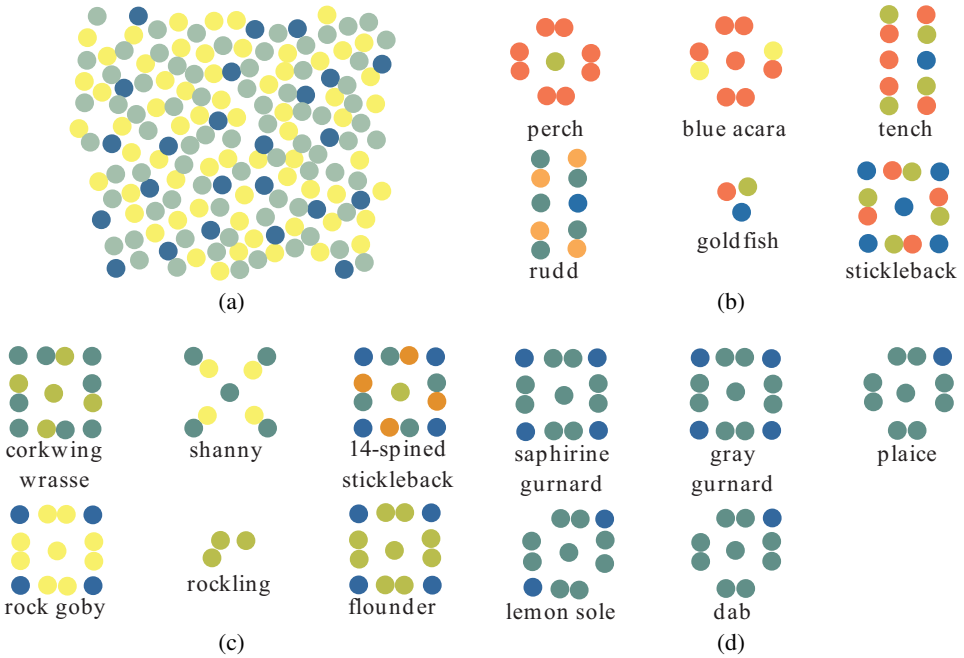


FIGURE 6.4

Cone mosaic of human and fish retina (pictures taken from Reference [37]): (a) human cone mosaic, (b) freshwater fish, (c) litoral coastal fish, and (d) deep coastal fish.

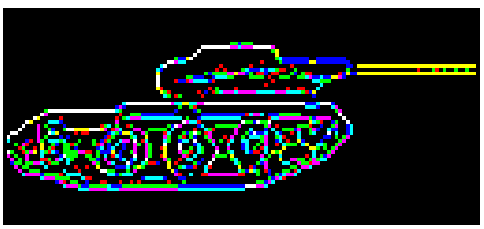


FIGURE 6.7

Summation of edge images of seven spectral bands. Different colors represent different intensity values (1: red, 2: green, 3: blue, 4: cyan, 5: magenta, 6: yellow, 7: white). © 2006 IEEE

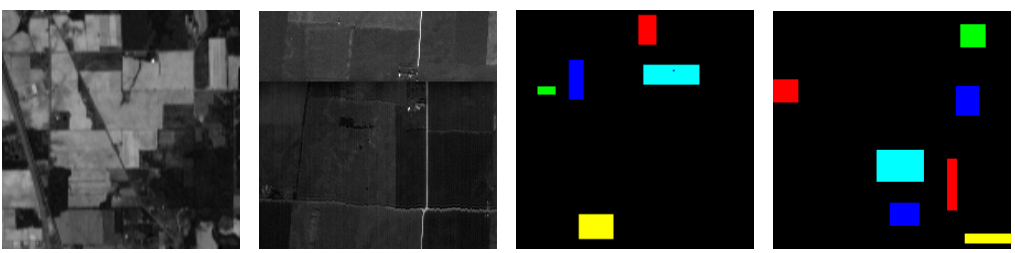


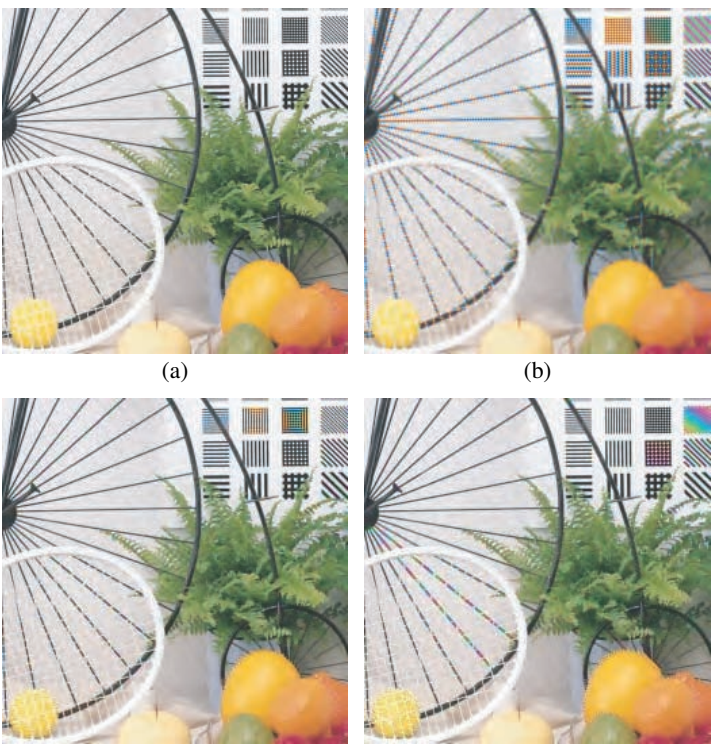
FIGURE 6.13

The visualization of the two real multispectral data sets and the corresponding class labels: (a) 92AV3C9 - band 1, (b) FLC1 - band 3, (c) class label of 92AV3C9, red-grass, green-tower, blue-corn, cyan-soil, yellow-hay, (d) class label of FLC1, red-oats, green-corn, blue-red clover, cyan-bare soil, and yellow-wheat. © 2006 IEEE

(a) (b)

FIGURE 7.12

Reconstruction of lighthouse image (a) using only $Q_2(u-0.5,v)$ and (b) using only $Q_2(u,v-0.5)$.



(a) (b)

(c) (d)

FIGURE 7.16

Portion of JPEG2000 test image bike, downsampled by four in each direction: (a) original color image, (b) image reconstructed from Bayer CFA using bilinear interpolation, (c) image reconstructed from Bayer CFA using the adaptive frequency demultiplexing algorithm, and (d) image reconstructed from the stripe CFA using the least-squares frequency demultiplexing.

(a) (b)

(c) (d)

FIGURE 7.17

Portion of Spincalendar: (a) original color image, (b) image reconstructed from Bayer CFA using bilinear interpolation, (c) image reconstructed from Bayer CFA using the adaptive frequency demultiplexing algorithm, and (d) image reconstructed from the stripe CFA using the least-squares frequency demultiplexing.

(a) (b) (c)

FIGURE 8.2

Example of color image decomposition into its luminance and chrominance components. (a) Color image with three chromatic values R, G and B at each spatial location. (b) The luminance component, a scalar value per spatial position corresponding to the mean of RGB values. (c) The chrominance values having three components per spatial position corresponding to the difference of each RGB component with the luminance.

(a) (b)

FIGURE 8.6

Examples of demosaicking by (a) pixel copy and (b) bilinear interpolation.

(a) (b)

FIGURE 8.7

Examples of demosaicking by (a) constant hue and (b) predetermined coefficients.

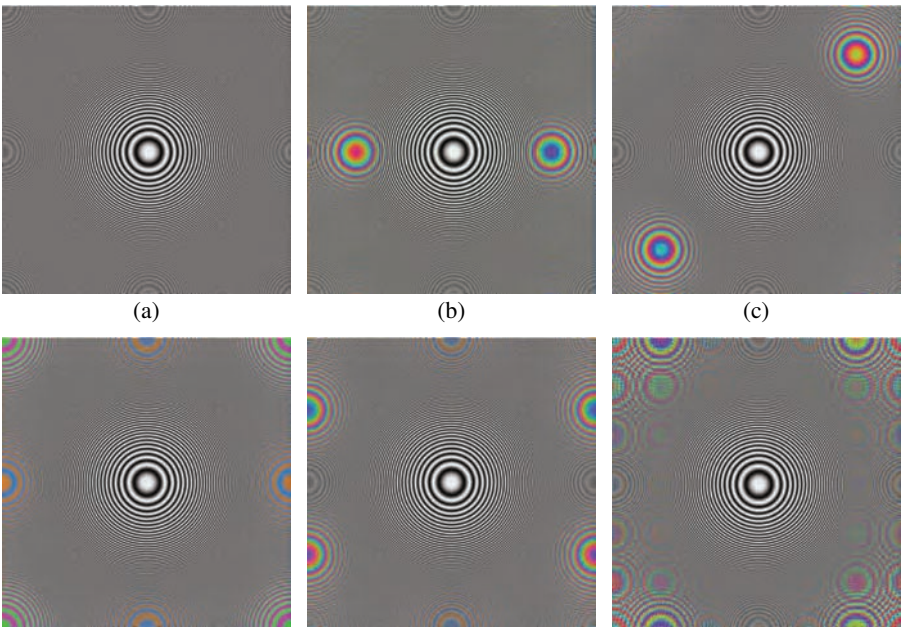


FIGURE 8.11

Results obtained using the 'CZP' image: (a) original image, (b-f) demosaicked images corresponding to (b) vertical stripe pattern, (c) diagonal stripe pattern, (d) Bayer pattern, (e) Lukac pattern, and (f) pseudorandom pattern.

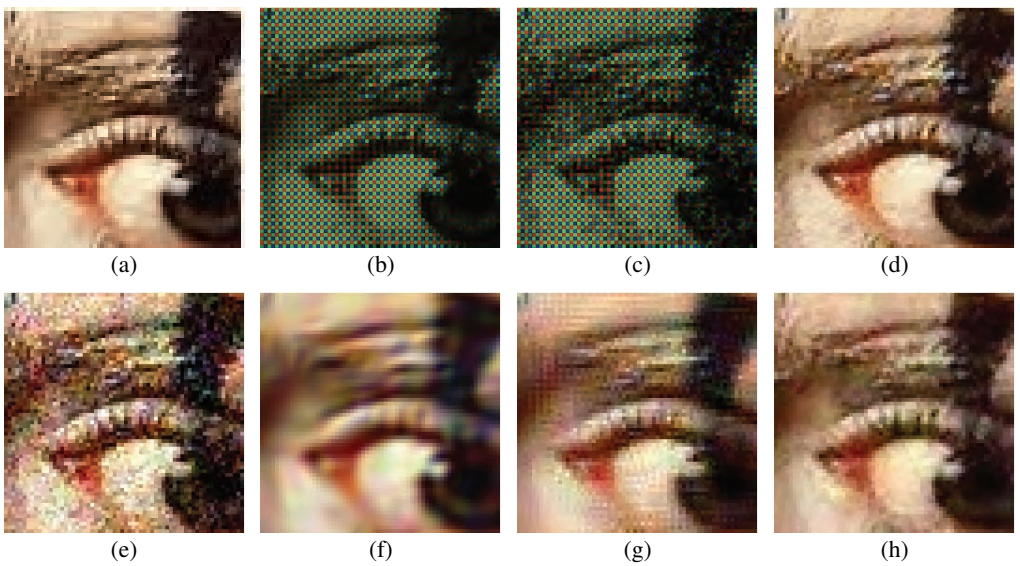


FIGURE 9.1

Zoomed portion of the *Clown* image: (a) original color image, (b) color version of ideal CFA image, (c) color version of noisy CFA image, (d) demosaicking the ideal CFA image, (e) demosaicking the noisy CFA image, (f) demosaicking the noisy CFA image followed by denoising, (g) denoising the noisy CFA image followed by demosaicking, and (h) joint denoising and demosaicking of the noisy CFA image.

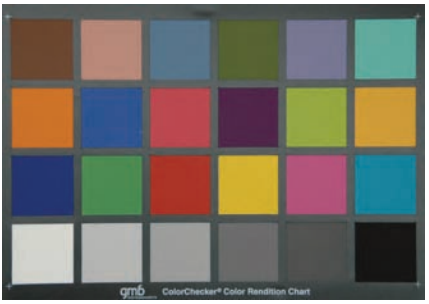
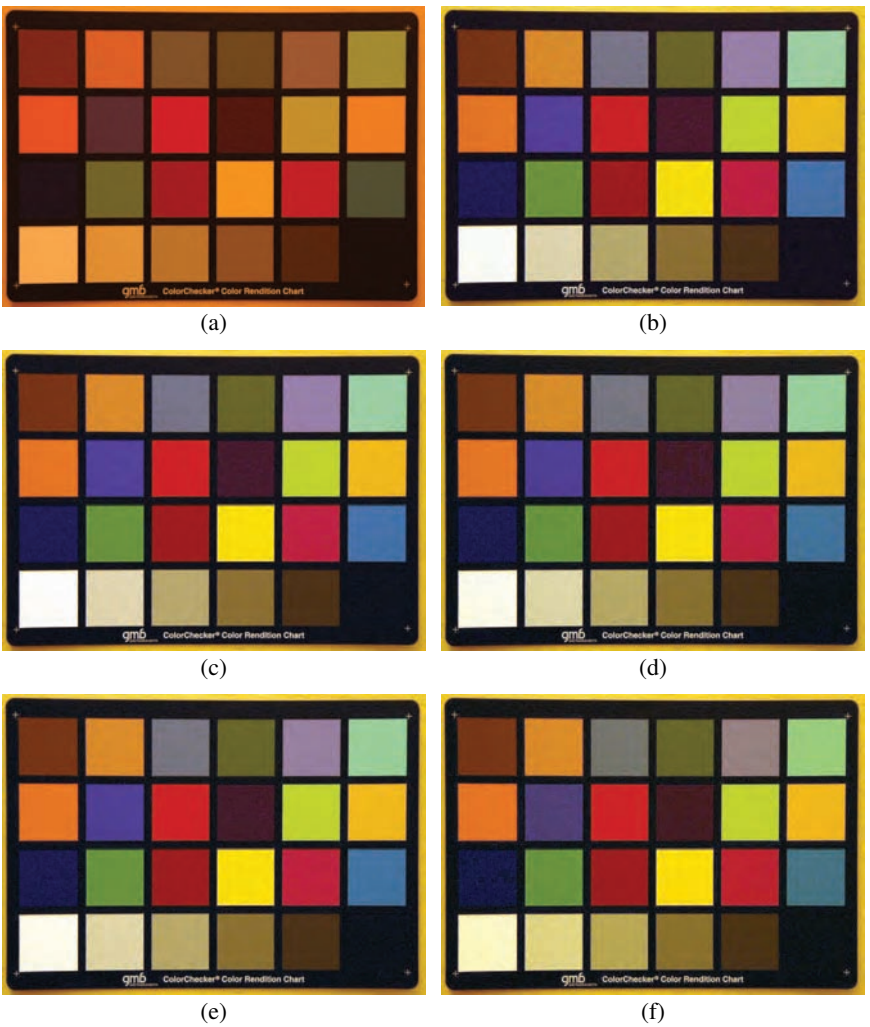


FIGURE 10.2

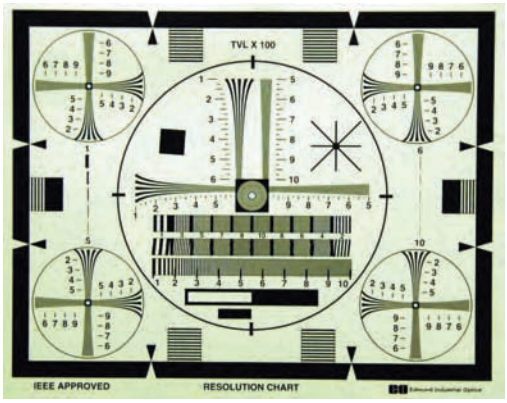
GretagMacbeth color rendition chart.



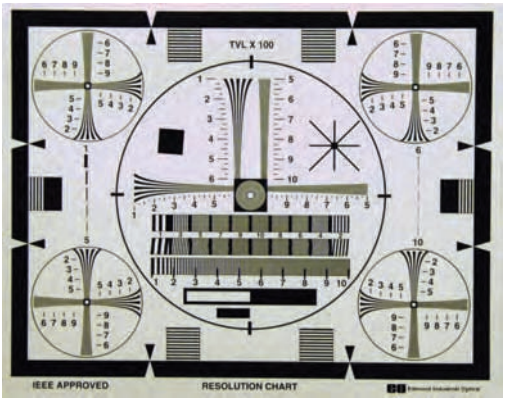
(a) (b) (c) (d) (e) (f)

FIGURE 10.9

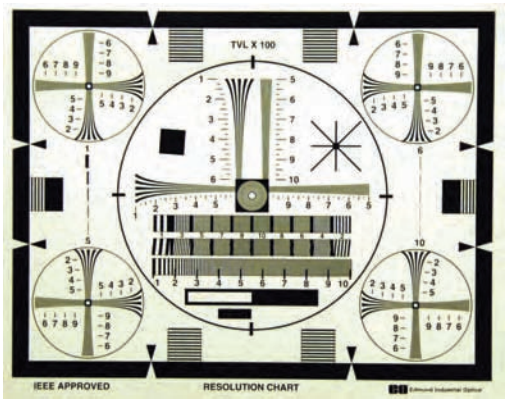
AWB methods for the Macbeth color chart: (a) original image, (b) gray world, (c) white patch, (d) iterative white balancing, (d) illuminant voting, and (f) color by correlation.



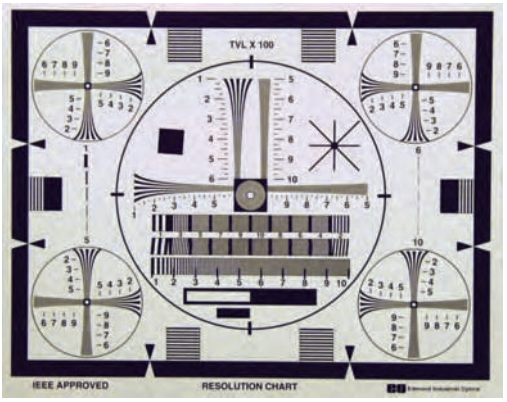
(a)



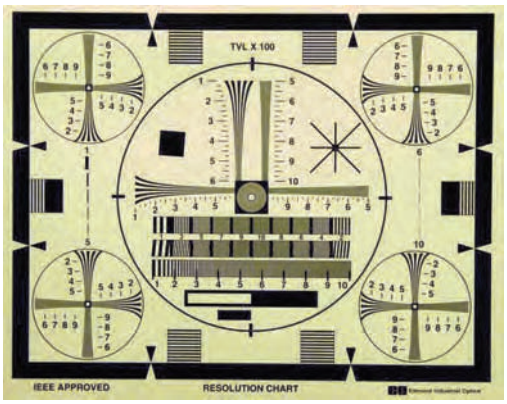
(b)



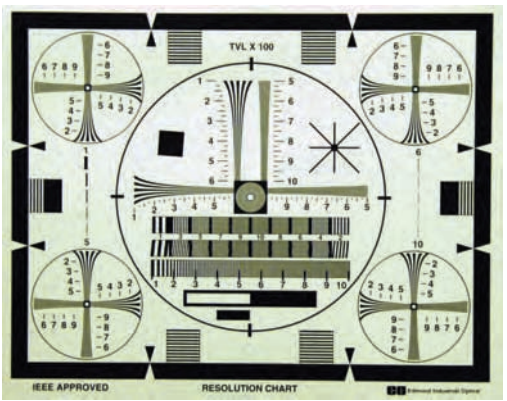
(c)



(d)



(e)



(f)

FIGURE 10.10

AWB methods for the resolution chart: (a) original image, (b) gray world, (c) white patch, (d) iterative white balancing, (d) illuminant voting, and (f) color by correlation.

FIGURE 10.11

AWB methods for the *bookshelf* image: (a) original image, (b) gray world, (c) white patch, (d) iterative white balancing, (d) illuminant voting, and (f) color by correlation.

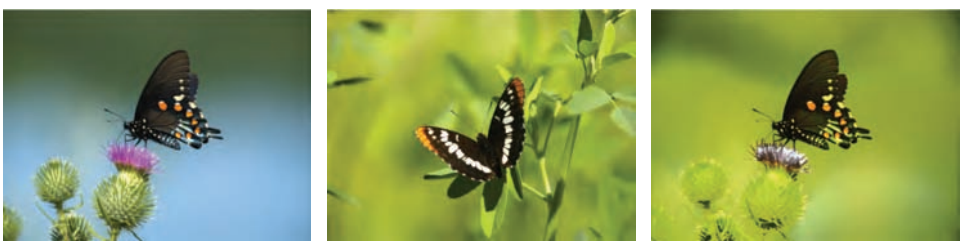


FIGURE 11.1

Color transfer example: (a) original image, (b) image with the target palette, and (c) output image. A color mapping is applied on the original picture to match the palette of an example provided by the user.

(a) (b) (c)

(d) (e) (f)

FIGURE 11.4

Results for linear techniques: (a) original image, (b) target palette, (c) separable linear transfer, (d) Cholesky based transfer, (e) principal axes transfer, and (f) linear Monge-Kantorovitch transfer. All transfers are done in the RGB color space.

(a) (b) (c)

(d) (e) (f)

FIGURE 11.5

Results for different scenarios: (a) original image, (b) target palette, (c) separable transfer, (d) composition transfer, (e) discrete Kantorovitch, and (f) IDT.

(a) (b) (c)

(d) (e) (f)

FIGURE 11.8

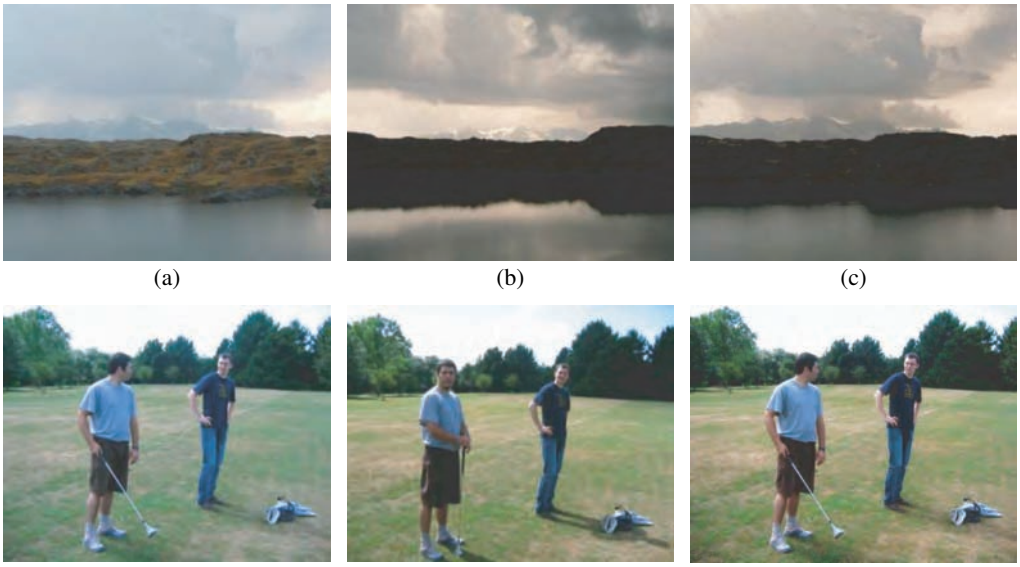
Results of the linear Monge-Kantorovitch transfer for different color spaces: (a) target color palette, (b) RGB, (c) YUV, (d) XYZ, (e) CIELAB, and (f) CIELUV.

(a) (b) (c)

(d) (e) (f)

FIGURE 11.9

Results of the IDT transfer for different color spaces: (a) target color palette, (b) RGB, (c) YUV, (d) XYZ, (e) CIELAB, and (f) CIELUV.



(a) (b) (c)

(d) (e) (f)

FIGURE 11.12

Examples of color grading for matching lighting conditions: (a-c) the color properties of the sunset are used to synthesize the evening scene depicted at sunset, (d-f) the color grading corrects the change of lighting conditions induced by clouds, and (c, f) the color transfer achieved by employing IDT followed by the regraining process.

(a) (b) (c)

(d) (e)

FIGURE 11.13

Example of color grading for image and video restoration used to recreate different atmospheres: (a) original frame, (b) 1970s atmosphere, (c) pub atmosphere, (d) linear MK result using 1970s atmosphere, and (e) IDT followed by regraining using pub atmosphere.

FIGURE 11.14

Color grading results: (a) original image, (b) target palette, (c) linear MK result, and (d) IDT result.

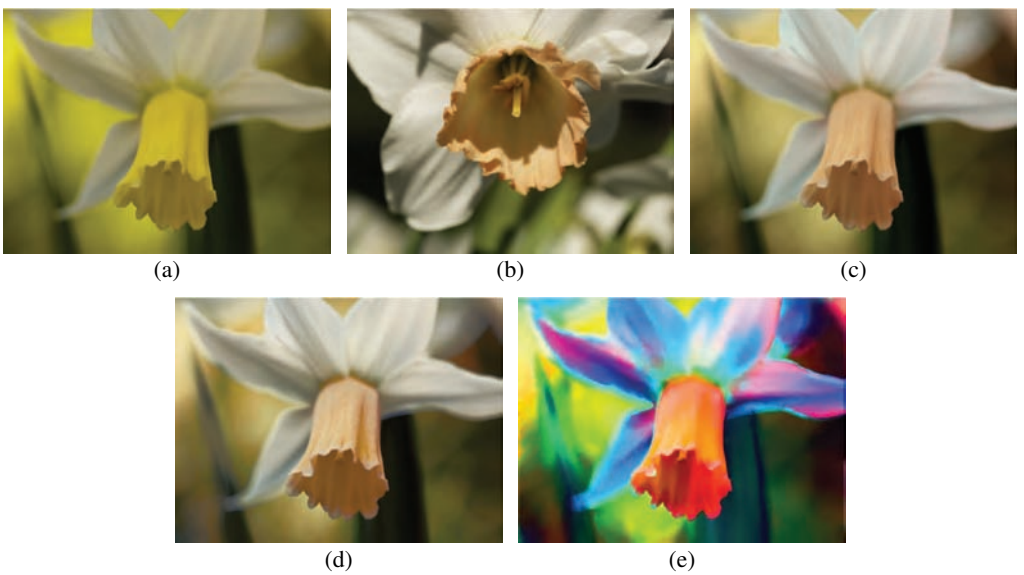


FIGURE 11.15

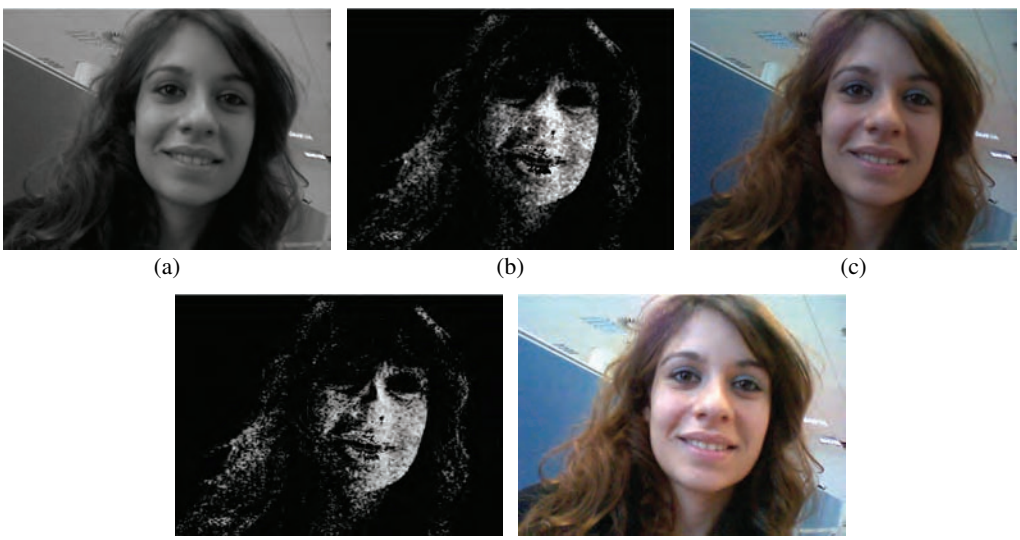
Color grading results: (a) original image, (b) target palette, (c) linear MK result, (d) IDT followed by the regraining process resulting in better color contrast, (e) original image recolored with the whole color spectrum using the IDT algorithm followed by the regraining process.

(a) (b) (c)

(d) (e) (f)

FIGURE 12.5

Skin detection examples on RGB images: (a,d) original images compressed in JPEG format, (b,e) simplest threshold method output, and (c,f) probabilistic threshold output.

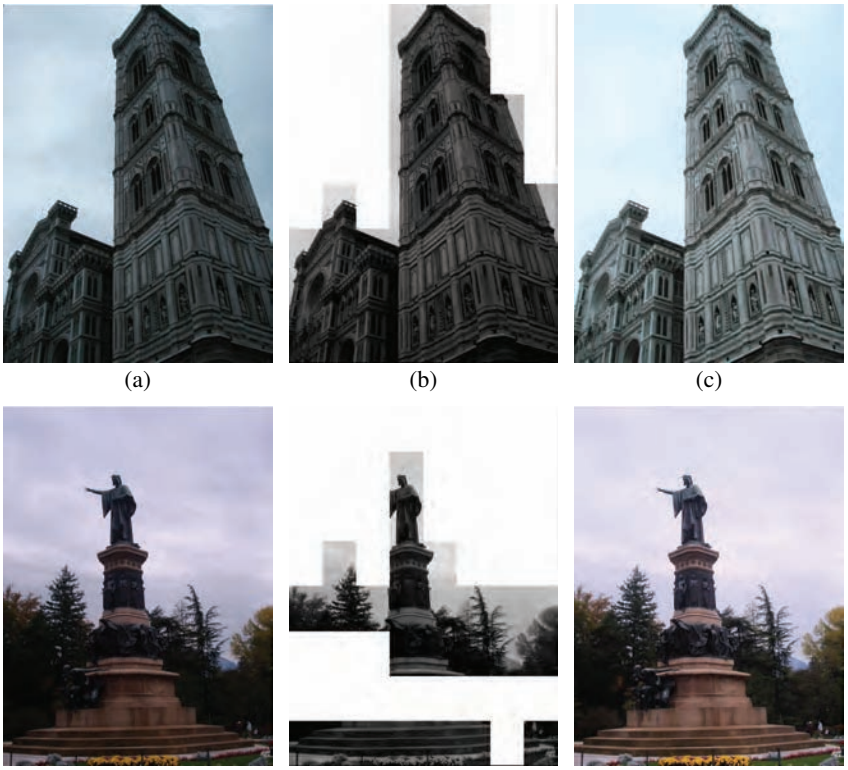


(a) (b) (c)

(d) (e)

FIGURE 12.7

Exposure correction results by real-time and post processing: (a) Bayer image, (b) skin detected in the Bayer image in real-time, (c) RGB color-interpolated image from Bayer data, (d) skin detected in RGB data using postprocessing, and (e) exposure-corrected image obtained from RGB image.

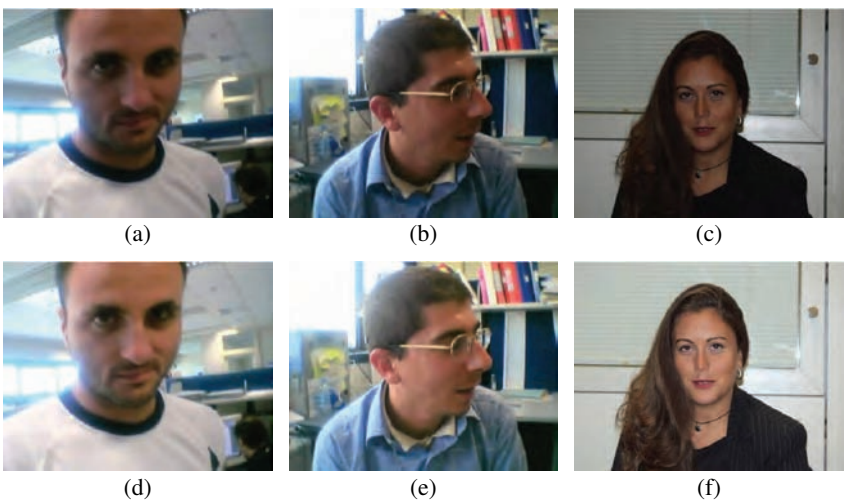


(a) (b) (c)

(d) (e) (f)

FIGURE 12.8

Exposure correction results by postprocessing: (a,d) original color input images, (b,e) contrast and focus visually significant blocks detected, and (c,f) exposure-corrected images obtained from RGB images.



(a) (b) (c)

(d) (e) (f)

FIGURE 12.9

Exposure correction results: (a-c) original images, (d-f) corrected images. Images in (a) and (b) were acquired by Nokia 7650 VGA sensor and compressed in JPEG format whereas the image in (c) was acquired with Olympus E-10 camera equipped with a 4.1 Mega-pixel CCD sensor.

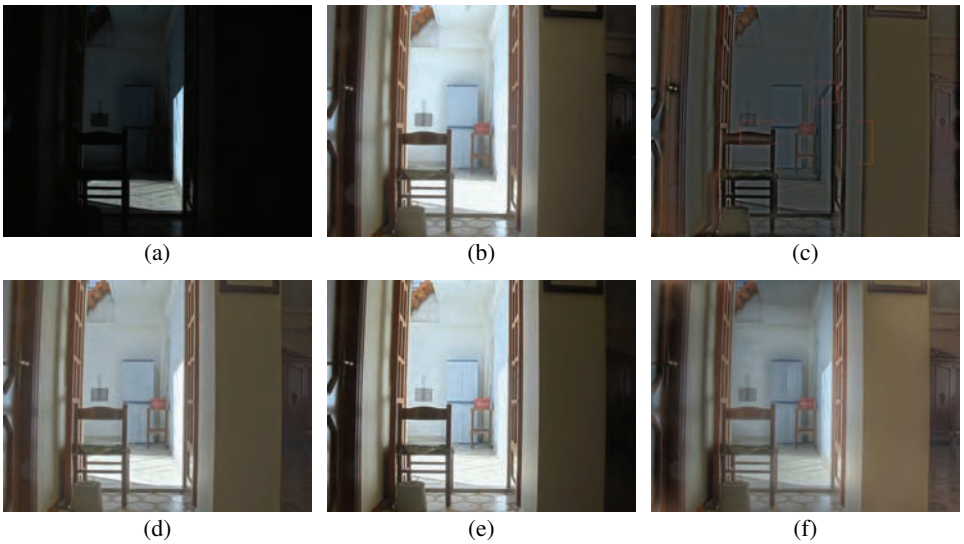


FIGURE 12.15

(a) HDR image built from the sequences of images shown in Figure 12.13 using linear scaling in the $[0, ..., 1]$ range and quantization to 8 bits, (b) image obtained using histogram adjustment mapping, (c) image mapped using Chiu's algorithm with some halo artifacts highlighted, (d) image mapped using bilateral filtering, (e) photographic tone reproduction mapping, and (f) gradient compression mapping.

FIGURE 14.4

Cropped region of Window image: (a) after bilinear demosaicking; (b) after bilinear demosaicking and JPEG (10:1) compression.

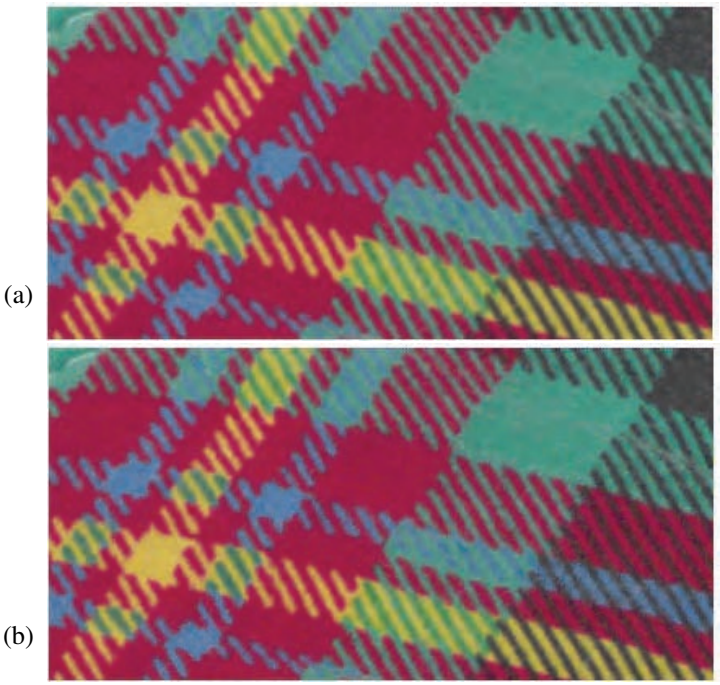


FIGURE 14.12

Comparison of the images reconstructed from a real CFA image by using bilinear demosaicking and JPEG2000 coding at 10:1 compression ratio: (a) conventional pipeline and (b) alternative pipeline.

FIGURE 16.1

Examples of the variability of the red-eye artifacts.

FIGURE 16.3

Examples of red eyes of different subjects, that shows the spread color distribution of the red-eye defects.

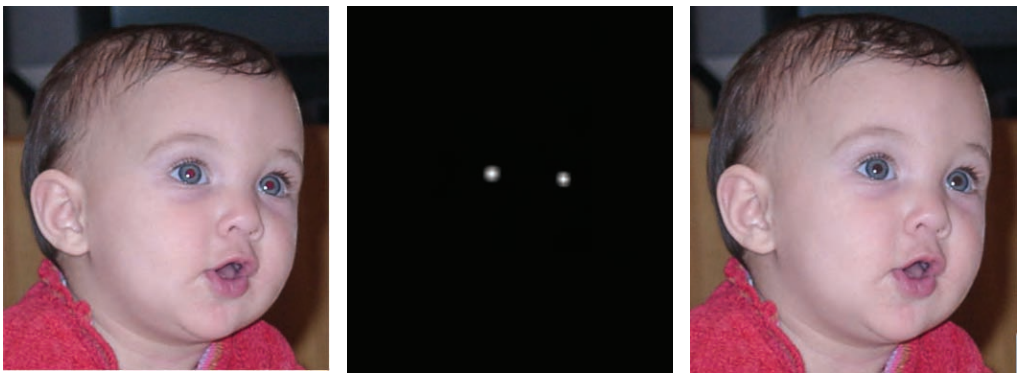
FIGURE 16.5

Further examples of the great variability of the red-eye artifacts. In the image on the left the subject has only one red eye, while in the image on the right the colors of the two eyes of the same subject are so different that probably most of the existing detection algorithms will miss one of them.

(a) (b) (c)

FIGURE 16.9

Red-eye correction process: (a) the original red-eye artifact, (b) the final correction, and (c) the corresponding smoothed pupil mask.



(a) (b) (c)

FIGURE 16.14

Correction process: (a) original image, (b) the smoothing mask of the red-eye areas, and (c) the corrected image.

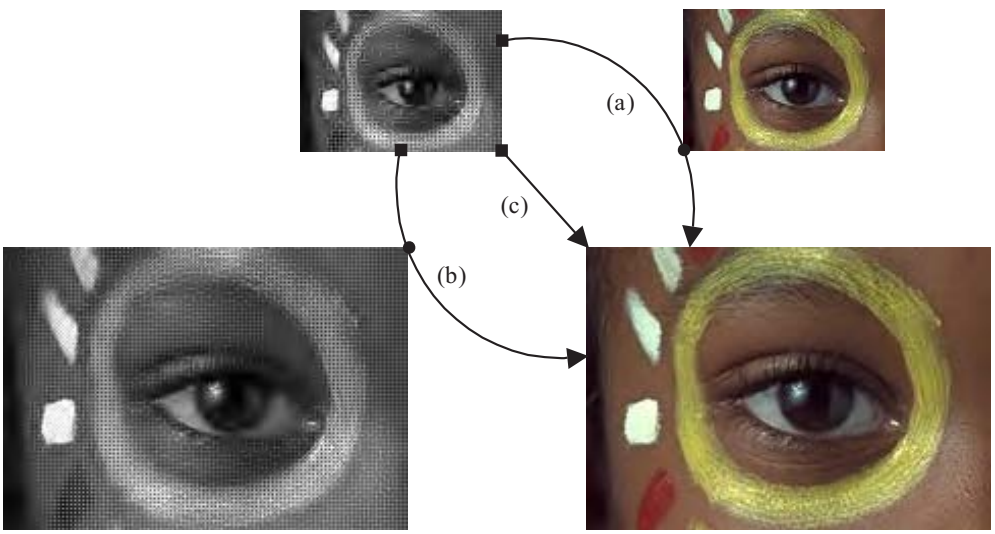
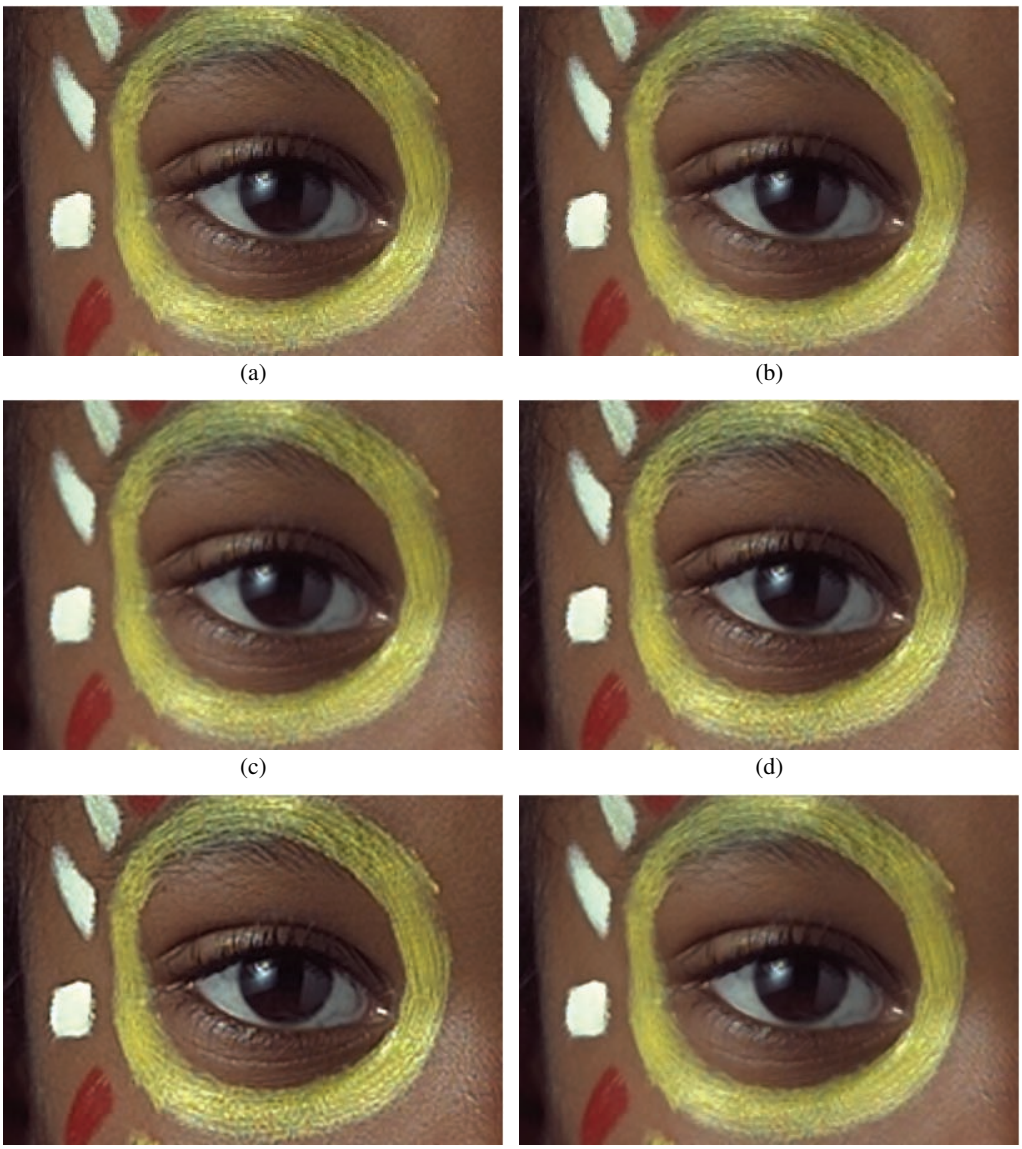


FIGURE 17.1

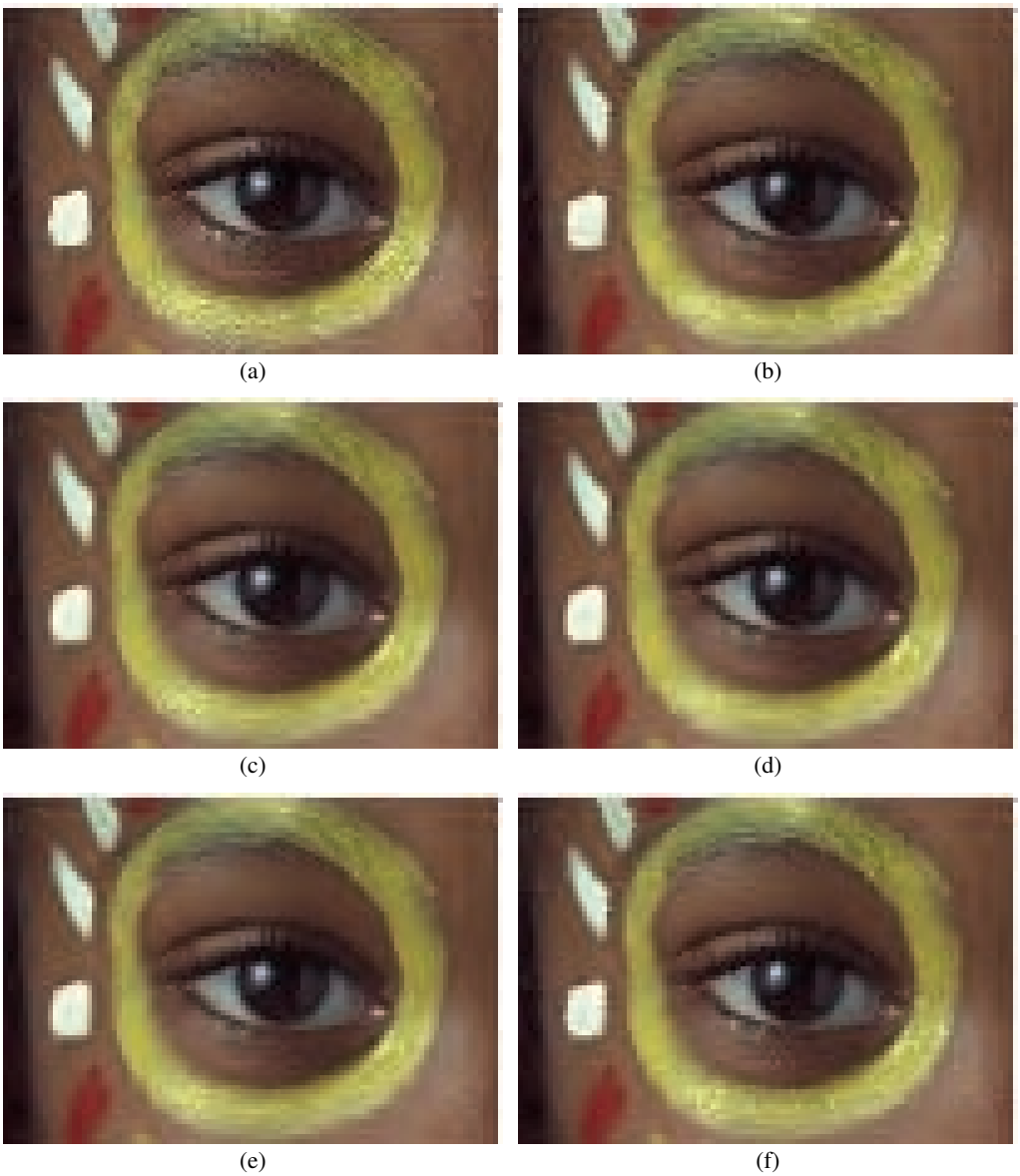
Image resizing approaches for single-sensor digital cameras: (a) demosaicked image resizing, (b) CFA image resizing, (c) joint demosaicking / image resizing. Displayed pictures represent: (top left) CFA image, (top right) demosaicked image, (bottom left) resized CFA image, and (right bottom) resized demosaicked image.



(a) (b) (c) (d) (e) (f)

FIGURE 17.3

Camera image upsampling with $\tau = 2$. Cropped 280×400 areas from demosaicked images upsampled using (a) nearest neighbor, (b) median, (c) bilinear, (d) bicubic, (e) joint interpolation / edge enhancement, and (f) edge-adaptive solutions.



(a) (b) (c) (d) (e) (f)

FIGURE 17.4

Camera image downsampling with $\tau = 1/2$. Cropped 70×100 areas from demosaicked images downsampled using (a) nearest neighbor, (b) median, (c) bilinear, (d) bicubic, (e) edge-adaptive, and (f) vector median solutions.

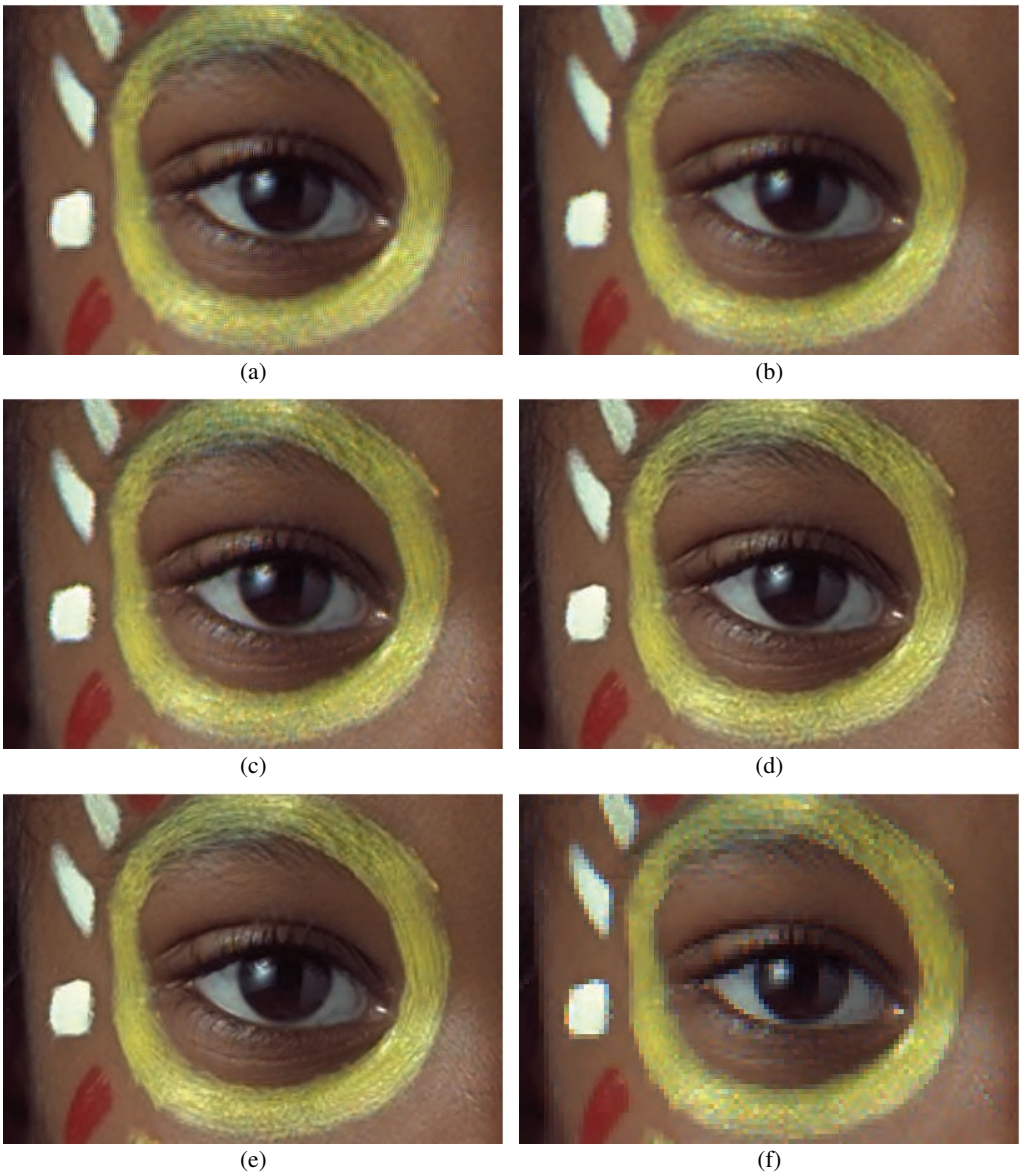
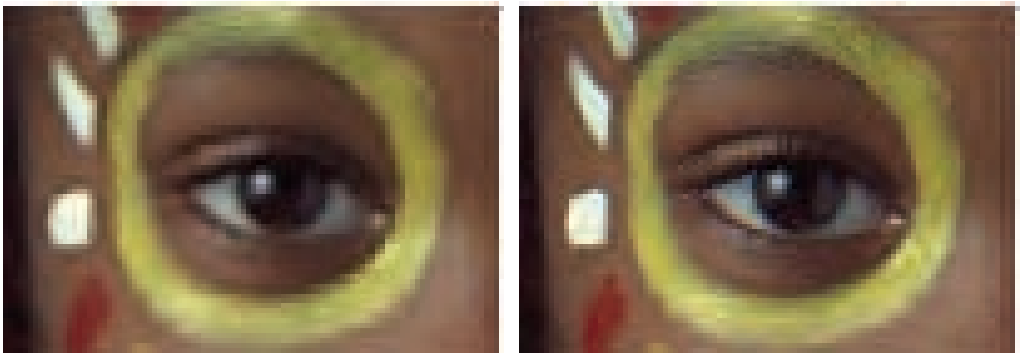


FIGURE 17.7

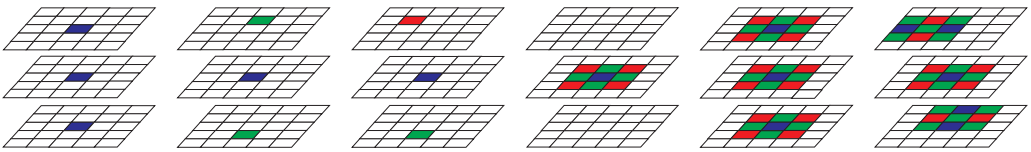
Camera image upsampling with $\tau = 2$. Cropped 280×400 areas from images obtained by (a) structure conversion-based CFA image resizing, (b) pixel mapping-based CFA image resizing using spatial correlations, (c) pixel mapping-based CFA image resizing using spatial and spectral correlations, (d) partially integrated demosaicking / resizing via mixing the upsampled demosaicked green channel and two upsampled color difference images, (e) partially integrated demosaicking / resizing by sharing edge orientation estimates, and (f) joint demosaicking / resizing.



(a) (b)

FIGURE 17.8

Camera image downsampling with $\tau = 1/2$. Cropped 70×100 areas from images obtained by (a) structure conversion-based CFA image resizing, and (f) joint demosaicking / resizing.



(a) (b) (c) (d) (e) (f)

FIGURE 18.2

Examples of basic windows used in video-demosaicking: (a) non-motion compensated temporal window, (b, c) motion compensated temporal windows, (d) spatial window, (e) non-motion compensated spatiotemporal window, and (f) motion compensated spatiotemporal window.

(a) (b) (c)

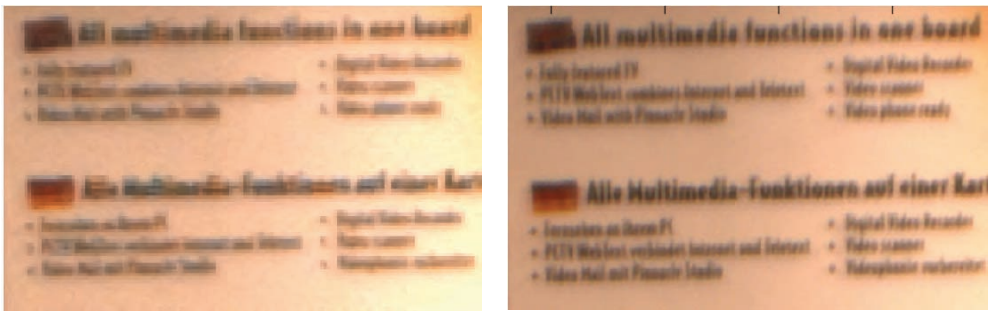
FIGURE 18.7

Experimentation using test image sequence from Figure 18.6a: (a) original image, (b) image demosaicked using the spatial method described in Section 18.3, and (c) image demosaicked using the spatiotemporal method presented in Section 18.4.

(a) (b) (c)

FIGURE 18.8

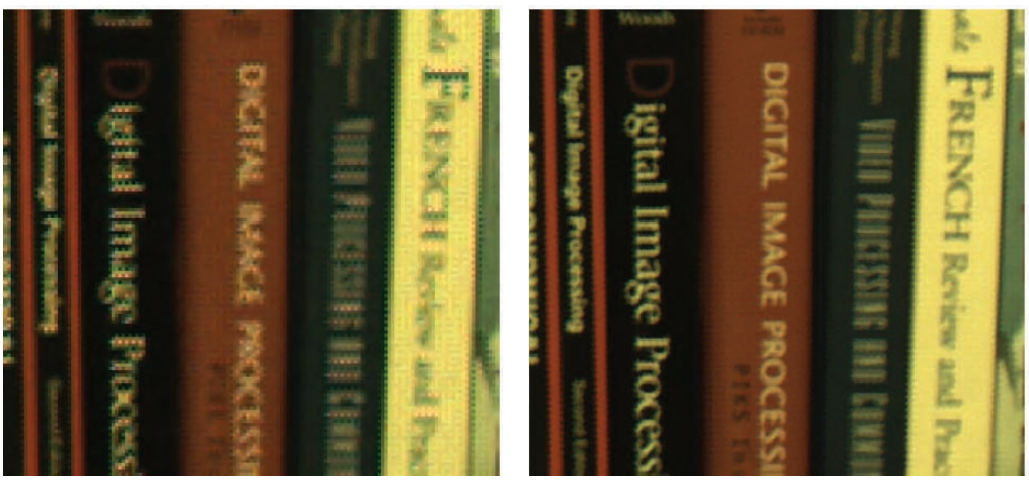
Experimentation using real single-sensor data: (a) original mosaic image, (b) image demosaicked using the spatial method described in Section 18.3, and (c) image demosaicked using the spatiotemporal method presented in Section 18.4.



(a) (b)

FIGURE 19.1

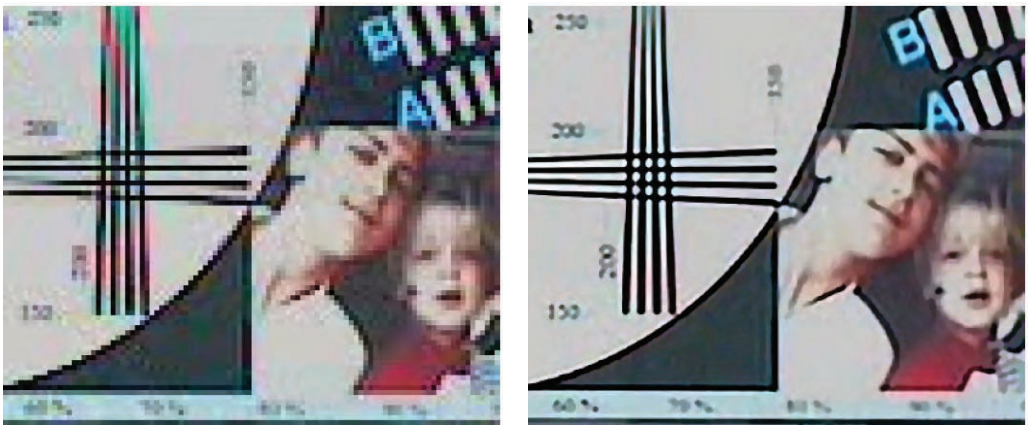
(a) Example showing the typical image quality problems associated with inexpensive single-sensor imaging devices. The image is noisy with evident color artifacts and has limited overall resolution as evidenced by the difficulty in reading the text numbers. (b) Image enhanced using a multiframe color supperresolution approach.



(a) (b)

FIGURE 19.10

Multiframe color superresolution applied to a real data sequence. (a) One of thirty-one low-resolution input images of size $141 \times 147 \times 3$ demosaicked by the single-frame method of Reference [7]. (b) Resulting image after multiframe demosaicking using the robust data penalty term of Equation 19.14.



(a) (b)

FIGURE 19.11

(a) One of forty captured images which are demosaicked and compressed in an unknown fashion by the imaging system. These images were used to reconstruct the higher resolution image in (b) by exploiting the multiframe demosaicking algorithm. © 2007 IEEE

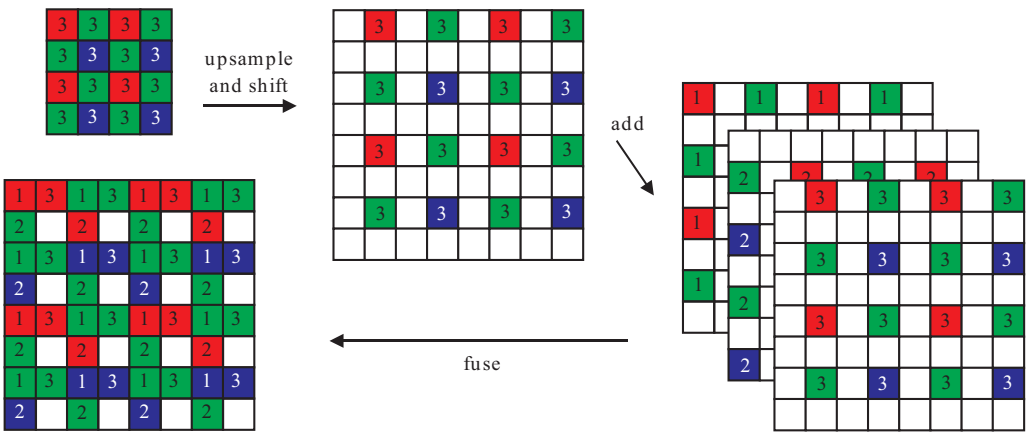


FIGURE 19.12

Diagram showing an example of the shift-and-add process. The input color channel image is upsampled by the resolution enhancement factor $r = 2$, and shifted according to the inverse of the image translation. This shifted image is added and averaged with the other $N - 1$ (in this case 2) upsampled and shifted low-resolution images.