GRADIENT BASED THRESHOLD FREE COLOR FILTER ARRAY INTERPOLATION

Ibrahim Pekkucuksen, Yucel Altunbasak

Center for Signal and Image Processing, Georgia Institute of Technology, Atlanta, GA, 30308, USA

ABSTRACT

Color Filter Array (CFA) interpolation is an integral part of image processing pipeline for single sensor digital cameras. Many CFA algorithms have been proposed over the years to improve resulting image quality. One such algorithm is the highly successful Directional Linear Minimum Mean-Square Error Estimation (DLMMSE) method. We make several observations on this algorithm and propose a new method to address those points. The proposed method yields visually pleasing results and outperforms all CFA interpolation algorithms that are included in a recent survey paper in terms of PSNR.

Index Terms— Demosaicing, color interpolation, Bayer color filter array

1. INTRODUCTION

Most digital cameras use single sensor arrays coupled with color filters to capture image data. The layout of color filter array determines which color channel is recorded at each pixel location. To obtain the complete color picture, missing color channel information needs to be interpolated from the single channel mosaicked image. This process is called Color Filter Array (CFA) interpolation or demosaicing. Although many different CFA patterns are available, the most prevalent one is the Bayer CFA [1] shown in Figure 1.

Simple interpolation techniques such as nearest neighbour or bilinear interpolation can be applied to the demosaicing problem. However, solutions ignoring edge structures and inter-channel correlation lead to color artifacts and blurry output. A notable early demosaicing method proposed using second order derivatives of red and blue pixels in initial green channel interpolation [2]. This idea became the basic building block of many subsequent algorithms. Authors of [3] proposed using variance of color differences to decide which interpolation direction to take. Directional filtering and a posteriori decision (DFPD) algorithm in [4] proposed performing interpolation in both horizontal and vertical directions and then making a hard direction decision based on sum of gradients along each direction.

DLMMSE algorithm [5] starts with calculating (G-R) and (G-B) color differences along horizontal and vertical directions. Then, these calculations are treated as noisy estimations of actual color differences and they are combined optimally using linear minimum mean-square error framework. After green channel interpolation is completed, missing red and blue channel pixels are filled using primary color difference signals and simple bilinear interpolation. Based on DLMMSE algorithm, authors of [6] proposed using directional anisotropic scale-adaptive filtering to improve demosaicing performance further.

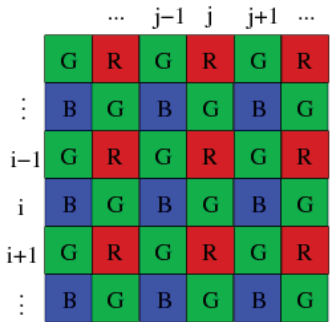


Fig. 1. Bayer CFA pattern.

In this paper, we present a new demosaicing method that addresses a few limitations we observed in DLMMSE algorithm. Firstly, as a result of its directional nature, DLMMSE algorithm uses only a subset of a target pixel's neighbours (pixels that share the same column or row with the target pixel) to find out how much each direction should contribute to color difference calculation. Although the solution is optimal in its own domain, unconsidered neighbour pixels might provide additional information that could improve color difference estimation. That is why, we want to include every neighbor pixel inside a given local window to the decision making process. However, since available color difference at every pixel is either (G-R) or (G-B), we cannot apply variance metric as DLMMSE method does. For this reason, we use gradients of color differences to come up with weights for each direction.

Secondly, DLMMSE method operates on horizontal and vertical directions like other directional methods. However, for a given direction, the conditions might be different for pixels falling to the opposite sides of the target pixel especially near edges or in texture regions. That is why, we decouple north-south and east-west directions from each other

and consider them seperately. Instead of making a hard direction decision, we combine estimations from every direction which eliminates the need for setting thresholds.

The rest of the paper is orginazed as follows. Section 2 describes proposed demosaicing method in detail, section 3 presents objective and subjective comparison results, and section 4 offers a brief discussion and conclusion.

2. ALGORITHM OVERVIEW

Bayer CFA pattern contains twice as many green channel samples as red and blue ones. That is why green channel suffers less from aliasing which makes it the natural choice for the starting point of demosaicing process. Proposed algorithm first interpolates the green channel in a single run, then uses its results to fill in red and blue channels.

2.1. Green Channel Interpolation

The first step of the algorithm is applying Hamilton and Adams' [2] interpolation formula to all pixels in both vertical and horizontal directions. For red pixel locations, horizontal and vertical green channel estimations are calculated as follows:

$$\tilde{G}^H_{i,j} = (G_{i,j-1}+G_{i,j+1})/2+(2*R_{i,j}-R_{i,j-2}-R_{i,j+2})/4$$

$$\tilde{G}^V_{i,j} = (G_{i-1,j}+G_{i+1,j})/2+(2*R_{i,j}-R_{i-2,j}-R_{i+2,j})/4. \quad (1)$$

Similarly, for green pixels with red vertical neighbours, vertical red channel estimation is:

$$\tilde{R}^V_{i,j} = (R_{i-1,j}+R_{i+1,j})/2+(2*G_{i,j}-G_{i-2,j}-G_{i+2,j})/4. \quad (2)$$

And for green pixels with horizontal red channel neigbours,

$$\tilde{R}^H_{i,j} = (R_{i,j-1}+R_{i,j+1})/2+(2*G_{i,j}-G_{i,j-2}-G_{i,j+2})/4. \quad (3)$$

Estimations with blue pixels are calculated in the same manner, simply by replacing R with B in the formulas above. The next step is to find horizontal and vertical color difference estimations using original and directionally estimated pixel values.

$$\tilde{\Delta}^V_{g,r}(i,j) = \begin{cases} \tilde{G}^V_{i,j} - R_{i,j}, & \text{G is interpolated} \\ G_{i,j} - \tilde{R}^V_{i,j}, & \text{R is interpolated} \end{cases}$$

$$\tilde{\Delta}^H_{g,r}(i,j) = \begin{cases} \tilde{G}^H_{i,j} - R_{i,j}, & \text{G is interpolated} \\ G_{i,j} - \tilde{R}^H_{i,j}, & \text{R is interpolated} \end{cases} \quad (4)$$

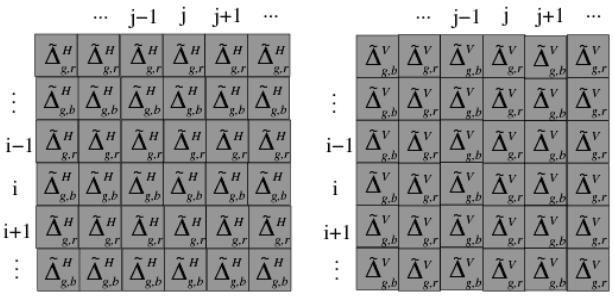


Fig. 2. Horizontal and vertical color difference maps.

Again, horizontal and vertical (G-B) difference estimations are calculated similarly. By the end of this step, we have two difference maps, one for horizontal and the other one for vertical estimations as shown in Figure 2.

Next, directional color differences are combined to form final difference estimation for the target pixel:

$$\begin{aligned} \tilde{\Delta}_{g,r}(i,j) = &[w_N * f * \tilde{\Delta}^V_{g,r}(i-4:i,j)+ \\ &w_S * f * \tilde{\Delta}^V_{g,r}(i:i+4,j)+ \\ &w_E * \tilde{\Delta}^H_{g,r}(i,j-4:j) * f'+ \\ &w_W * \tilde{\Delta}^H_{g,r}(i,j:j+4) * f']/w_T \end{aligned}$$

$$w_T = w_N + w_S + w_E + w_W$$

$$f = [1\ 1\ 1\ 1\ 1]/5. \quad (5)$$

The vector f could be modified to put more weight to color differences closer to the target pixel. The weight for each direction (w_N, w_S, w_E, w_W) is calculated by adding color difference gradients in that direction over a local window. For a window size of 5 by 5:

$$w_N = 1/(\sum_{a=i-4}^{i}\sum_{b=j-2}^{j+2} D^V_{a,b})^2$$

$$w_S = 1/(\sum_{a=i}^{i+4}\sum_{b=j-2}^{j+2} D^V_{a,b})^2$$

$$w_W = 1/(\sum_{a=i-2}^{i+2}\sum_{b=j-4}^{j} D^H_{a,b})^2$$

$$w_E = 1/(\sum_{a=i-2}^{i+2}\sum_{b=j}^{j+4} D^H_{a,b})^2 \quad (6)$$

where gradients are defined as:

$$D^V_{i,j} = |\tilde{\Delta}^V_{i-1,j} - \tilde{\Delta}^V_{i+1,j}|$$

$$D^H_{i,j} = |\tilde{\Delta}^H_{i,j-1} - \tilde{\Delta}^H_{i,j+1}| \quad (7)$$

Finally, target green pixel value is calculated by adding estimated color difference to the available (red or blue) target pixel:

$$\tilde{G}(i,j) = R(i,j) + \tilde{\Delta}_{g,r}(i,j)$$
$$\tilde{G}(i,j) = B(i,j) + \tilde{\Delta}_{g,b}(i,j) \quad (8)$$

2.2. Red and Blue Channel Interpolation

Red pixel values at blue locations and blue pixel values at red locations are interpolated using the following filter that was proposed in [6]:

$$p_{rb} = \begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 10 & 0 & 10 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 10 & 0 & 10 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 0 \end{bmatrix} * \frac{1}{32}$$

$$\tilde{R}_{i,j} = \tilde{G}_{i,j} - \tilde{\Delta}_{g,r}(i-3:i+3, j-3:j+3) \otimes p_{rb}$$
$$\tilde{B}_{i,j} = \tilde{G}_{i,j} - \tilde{\Delta}_{g,b}(i-3:i+3, j-3:j+3) \otimes p_{rb} \quad (9)$$

where $\otimes$ denotes element-wise matrix multiplication and then summation of elements.

For red and blue pixels at green locations, we use bilinear interpolation over the closest four neighbors. The immediate vertical neighbours of a green pixel are either red or blue pixels. For the red pixel case, red and blue pixel values at a green pixel location are interpolated as follows:

$$\begin{aligned} \tilde{R}(i,j) = G(i,j) &- (\tilde{G}_{i-1,j} - R_{i-1,j})/4 \\ &- (\tilde{G}_{i+1,j} - R_{i+1,j})/4 \\ &- (\tilde{G}_{i,j-1} - \tilde{R}_{i,j-1})/4 \\ &- (\tilde{G}_{i,j+1} - \tilde{R}_{i,j+1})/4 \end{aligned}$$

$$\begin{aligned} \tilde{B}(i,j) = G(i,j) &- (\tilde{G}_{i-1,j} - \tilde{B}_{i-1,j})/4 \\ &- (\tilde{G}_{i+1,j} - \tilde{B}_{i+1,j})/4 \\ &- (\tilde{G}_{i,j-1} - B_{i,j-1})/4 \\ &- (\tilde{G}_{i,j+1} - B_{i,j+1})/4 \end{aligned} \quad (10)$$

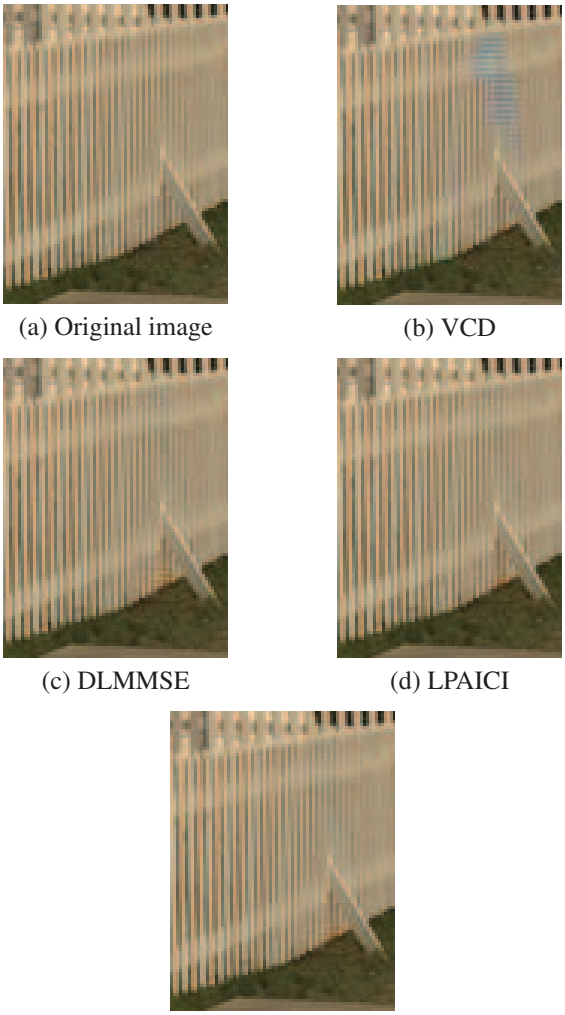
The interpolation formulas are similar for the blue vertical neighbour case. At this point, we interpolated missing pixels for every channel and reconstructed our color image.

Table 1. Comparison of PSNR values for different demosaicing methods.

No.	SA	VCD	DL	LPA	Prop
1	41.90	42.97	42.90	**43.86**	43.35
	46.32	46.74	47.56	**47.75**	47.66
	42.96	43.50	43.86	**44.46**	44.10
2	40.21	40.93	41.39	42.10	**42.11**
	43.48	43.83	43.69	44.81	**45.02**
	39.54	40.37	40.44	41.07	**41.10**
3	42.53	42.92	42.95	**43.77**	43.57
	44.52	45.58	46.26	46.53	**46.77**
	40.48	41.85	41.80	**42.65**	42.26
4	35.96	36.57	36.33	**37.42**	37.26
	40.37	40.25	39.63	**41.15**	41.12
	36.78	37.10	36.66	**37.85**	37.59
5	42.93	43.70	43.71	44.26	**44.29**
	46.12	46.71	46.68	47.01	**47.33**
	42.33	43.10	42.93	**43.42**	43.37
6	39.18	39.73	39.98	40.44	**40.66**
	43.29	43.33	43.19	43.85	**44.42**
	40.53	40.82	40.96	41.39	**41.65**
7	43.40	44.47	44.75	44.91	**45.33**
	46.42	47.03	46.82	47.15	**47.87**
	42.24	43.55	43.54	43.77	**43.98**
8	40.98	41.10	41.69	42.18	**42.35**
	44.33	44.09	44.14	44.72	**45.25**
	40.33	40.60	40.88	41.46	**41.57**
9	42.15	42.31	42.77	42.95	**43.27**
	45.07	44.89	44.88	45.14	**45.83**
	40.33	40.98	40.95	41.36	**41.48**
10	40.32	40.21	40.40	40.91	**40.94**
	43.45	42.89	42.43	43.13	**43.57**
	39.41	39.41	39.34	39.77	**39.83**
11	38.54	38.93	39.20	39.37	**39.60**
	41.56	41.97	42.26	42.32	**42.90**
	38.13	38.77	38.84	39.10	**39.25**
12	37.18	37.29	37.99	37.78	**38.09**
	39.88	39.86	39.87	40.05	**40.30**
	35.68	35.75	**36.22**	36.17	36.18
Avg	41.36	41.78	41.89	42.39	**42.53**

3. EXPERIMENTAL RESULTS

The proposed algorithm is tested on the first 12 images of Kodak test set that was used in a recent survey paper [7]. The survey paper compared eleven state-of-the-art methods from which we report the four highest performing ones. These methods are Local Polynomial Approximation (LPA) [6], Directional Linear Minimum Mean Square-Error Estimation (DLMMSE) [5], Variance of Color-Difference (VCD) [3], and Successive Approximation with edge-weighted improvement (SA) [8], respectively. The PSNR comparison results are summarized in Table 1. The proposed algorithm has the best overall PSNR average, outperforming the nearest method (LPA) by 0.14 dB and DLMMSE method by 0.64 dB. Figure 3 shows commonly used fence region from the lighthouse image for subjective quality comparison.



(a) Original image (b) VCD

(c) DLMMSE (d) LPAICI

(e) Proposed

Fig. 3. Fence region from the lighthouse image.

4. CONCLUSION

In this paper, we presented an easy to implement, noniterative, gradient based demosaicing algorithm. Experimental results show that proposed algorithm outperforms other available demosaicing solutions in terms of objective PSNR comparison. The performance of the proposed algorithm might be improved by using adaptive size filters in green channel interpolation stage instead of a fixed filter. Introducing a gentle post processing step might also improve its results. However, additional computational cost and quality tradeoff is always a concern with post processing steps.

5. REFERENCES

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

[2] J. F. Hamilton and J. E. Adams, "Adaptive Color Plane Interpolation in Single Sensor Color Electronic Camera," U.S. Patent 5 629 734, 1997.

[3] K.-H. Chung and Y.-H. Chan, "Color demosaicing using variance of color differences," IEEE Transactions on Image Processing, vol. 15, no. 10, pp. 2944-2955, Oct. 2006.

[4] D. Menon, S. Andriani, and G. Calvagno, "Demosaicing with directional filtering and a posteriori decision," IEEE Transactions on Image Processing, vol. 16, no. 2, pp. 32141, Jan. 2007.

[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, Dec. 2005.

[6] D. Paliy, V. Katkovnik, R. Bilcu, S. Alenius, and K. Egiazarian, "Spatially adaptive color filter array interpolation for noiseless and noisy data," International Journal of Imaging Systems and Technology, vol. 17, no. 3, pp. 105-122, 2007.

[7] X. Li, B. Gunturk, and L. Zhang, "Image Demosaicing: A Systematic Survey," Proceedings of the SPIE - The International Society for Optical Engineering, vol. 6822, no. 1, Jan. 2008.

[8] C. Y. Su, "Highly effective iterative demosaicing using weighted-edge and color-difference interpolations," IEEE Transactions on Consumer Electronics, vol. 52, no. 2, pp. 639-645, May 2006.