

Research Article

Demosaicking Based on Optimization and Projection in Different Frequency Bands

Osama A. Omer and Toshihisa Tanaka

Department of Electrical and Electronic Engineering, Tokyo University of Agriculture and Technology, Tokyo 184-8588, Japan

Correspondence should be addressed to Osama A. Omer, osama@sip.tuat.ac.jp

Received 30 July 2007; Revised 10 November 2007; Accepted 23 November 2007

Recommended by Alain Tremeau

A fast and effective iterative demosaicking algorithm is described for reconstructing a full-color image from single-color filter array data. The missing color values are interpolated on the basis of optimization and projection in different frequency bands. A filter bank is used to decompose an initially interpolated image into low-frequency and high-frequency bands. In the low-frequency band, a quadratic cost function is minimized in accordance with the observation that the low-frequency components of chrominance slowly vary within an object region. In the high-frequency bands, the high-frequency components of the unknown values are projected onto the high-frequency components of the known values. Comparison of the proposed algorithm with seven state-of-the-art demosaicking algorithms showed that it outperforms all of them for 20 images on average in terms of objective quality and that it is competitive with them from the subjective quality and complexity points of view.

Copyright © 2008 O. A. Omer and T. Tanaka. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Usage of digital cameras is spreading rapidly as they are easy-to-use image input devices. The increasing popularity of digital cameras has provided motivation to improve all elements of the digital photography signal. Digital color cameras are typically designed to use a single image sensor. Each individual sensor element is able to capture a single color. The arrangement of the color filters is called a color filter array (CFA). In the Bayer pattern [1], a popular CFA pattern, the sensor produces a two-dimensional array in which each spatial location contains only a red (R), green (G), or blue (B) component. Green pixels are sampled at a higher rate than blue and red pixels. The recovery of full-color images from a CFA-based detector requires a method for calculating the values of the missing colors at each pixel. Such methods are commonly referred to as color-interpolation or color-demosaicking algorithms.

A number of demosaicking algorithms [2–22] with an exploiting structure between channels have been proposed. These algorithms vary from fast with lower quality to more complex with higher quality and can be classified into two categories: noniterative [2–10] and iterative [12–16].

In general, noniterative algorithms require less computational time but have a worse image quality. Among the noniterative algorithms, bilinear interpolation is the simplest and fastest, but it has the lowest quality. It works well in smooth regions and fails in regions with high-frequency components such as edges. To avoid interpolation across edges, a method has been proposed that interpolates along an object boundary with edge sensing [2]. The edges are sensed by finding the outlier pixel in a square of four pixels and interpolating the missing values using the neighboring pixels excluding the outlier. Some algorithms, such as that of Lukac et al. [7], interpolate color assuming that the quotient of two color channels varies slowly. This follows from the fact that, if two colors occupy the same coordinate in the chromaticity plane, the ratios between their components are equal. Instead of using the quotient, some algorithms [3, 4] use the color differences on the basis of the assumption that differences between green and red (or blue) vary slowly within the same image object. The algorithms in this category make no use of the obtained estimate of one color to get further improvements in the other colors. The main drawback of these demosaicking algorithms is that the simple assumption made about smoothness or about the slowly

varying quotient is not enough to overcome the error around edges.

In addition, some algorithms [8] are very complex due to the need for matrix inversion and nonlinear operations. With others [9, 10], the frequent switching between horizontal and vertical directions may break thin, low-contrast lines into pieces. One way to overcome this problem is to use an averaging filter (as suggested by the authors), but this leads to a smoothness problem, as will be shown in the simulation results.

The iterative algorithms update the initially interpolated image on the basis of the assumption that an improvement in one channel will lead to improvements in other channels. In Kimmel's algorithm [12], the demosaicking is performed in two steps. The first step is reconstruction: the green component is first reconstructed using the red and blue gradients, and then the red and blue ones are reconstructed using the green values, edge approximations, and a simple color ratio rule that says that, within a given "object," the red/green ratio is locally constant (the same is true for the blue/green one). In the second step, the reconstructed full-color image is enhanced using an inverse diffusion filter. This algorithm is very complex due to the calculation of the color ratios in each iteration and the use of nonlinear operations for image enhancement. Moreover, convergence is not guaranteed. Gunturk et al. proposed an algorithm based on projections onto convex sets to refine the red and blue planes that alternatively enforce the two convex-set constraints [13]. While this algorithm efficiently uses the spectral correlation, the spatial correlation is not incorporated effectively. An extension of this algorithm incorporates spatial correlation [14]. It is used in a simultaneous demosaicking and super-resolution framework. It forces the full-color image to obey the color difference rule by inserting a color difference constraint in the alternative projection process. The main disadvantage of this algorithm is its complexity. It requires filtering in each iteration. Moreover, the incorporation of the spatial correlation property without avoiding smoothness across edges leads to color artifacts in the reconstructed image. The algorithm proposed by Su [16] effectively incorporates the spatial correlation in the initial step by using weighted-edge interpolation. Both the refinement and iterative steps are based on a color difference rule, which states that (green-blue) and (green-red) color differences are constant within a region. The iteration is based on thresholding the variance of the change for each channel. If the variance is larger than a certain value, the color difference rule is applied to that channel. The main disadvantage of this algorithm is that there is no guarantee of convergence during the iteration since the iterative step is not convex, so the resulting full-color image depends on the initial estimation. In a way similar to Su's algorithm [16], the idea of iteratively applying the color difference rule in an algorithm has been proposed [15]. However, this algorithm is more complex than Su's and convergence is not guaranteed. In the algorithms of Farsiu et al. [23, 24], the assumption of smooth luminance and chrominance is used in a simultaneous demosaicking and super-resolution framework. The main drawbacks of this algorithm are its complexity and the over-smoothness of chrominance

G_{11}	R_{12}	G_{13}	R_{14}	G_{15}	R_{16}
B_{21}	G_{22}	B_{23}	G_{24}	B_{25}	G_{26}
G_{31}	R_{32}	G_{33}	R_{34}	G_{35}	R_{36}
B_{41}	G_{42}	B_{43}	G_{44}	B_{45}	G_{46}
G_{51}	R_{52}	G_{53}	R_{54}	G_{55}	R_{56}
B_{61}	G_{62}	B_{63}	G_{64}	B_{65}	G_{66}

FIGURE 1: Bayer pattern.

because avoiding smoothness across chrominance edges is not considered.







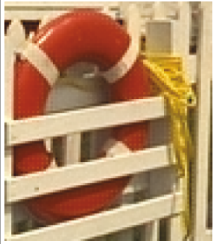



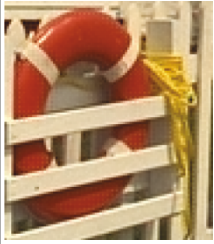

Although these iterative algorithms partially reduce the errors around edges, some of them produce errors in the smooth regions, as shown in Table 1, which presents examples of the effect of an increasing number of iterations on the edgy and smooth regions. While the successive iterations reduce the artifacts around the edges, the smooth regions are deformed with new artifacts.

We addressed three outstanding problems with demosaicking algorithms: the deformation of smooth regions by successive iterations, the lack of convergence, and algorithm complexity. These problems can be overcome by iteratively enhancing only the edgy regions in the low-frequency band rather than the entire initially interpolated image because the chrominance is smoother in the low-frequency band than in the whole image. Moreover, a significant improvement in the quality of demosaicked images is obtained by combining enhancement of the low-frequency band with projection of high-frequency bands from known channels onto unknown channels as proposed by Gunturk et al. [13]. A dyadic filter bank can be used to obtain the low-frequency and high-frequency bands. The enhancement is achieved by viewing the demosaicking as an optimization problem in which a cost function is minimized. The cost function is based on the observation that the chrominance varies slowly in an object region. Unlike the one used by Farsiu [24], the proposed cost function is defined as the weighted L_2 -norm of the chrominance in the low-frequency band, where edge indicators are used as weights to ensure slowly varying chrominance in each object region while high-frequency bands are reconstructed by projection. Using edge indicators helps to avoid smoothness in the chrominance across edges. Since the proposed cost function is positive definite quadratic by definition, it is guaranteed to converge to a global minimum. Comparison of the proposed algorithm with seven demosaicking algorithms (both noniterative and iterative) showed that the proposed algorithm works well in producing full-color images with fewer color artifacts in both the edgy and smooth regions.

The rest of this paper is organized as follows. Section 2 describes our iterative demosaicking algorithm and suggests an initial interpolation for fast convergence. Section 3 presents and discusses the simulation results. Section 4 concludes the paper with a brief summary.

We use the following notation. Let R , G , and B pixel values correspond to the red, green, and blue color channels, respectively. When necessary, we specify the location of a pixel

TABLE 1: Effect of iterations on edgy and smooth regions.

Iterations	Edgy region		Smooth region	
	Li [15]	Su [16]	Li [15]	Su [16]
0				
5				
10				

by using $R_{i,j}$, $G_{i,j}$, or $B_{i,j}$. For matrix A of size $M \times N$, \underline{A} is defined as a lexicographically ordered vector of size $MN \times 1$.

2. PROPOSED ITERATIVE DEMOSAICKING ALGORITHM

We assume that the given color channels are sampled using the Bayer pattern [1] (see Figure 1). Therefore, only one out of the R , G , and B values is known at each pixel. Our goal is to reconstruct the missing values. To achieve this goal, we developed a fast and efficient demosaicking algorithm consisting of simple interpolation, projection of high-frequency bands of unknown values onto high-frequency bands of known values, and chrominance enhancement in the low-frequency band. An illustrative example for the proposed algorithm is shown in Figure 2. A row-crossing edgy part is used to illustrate the main steps. The dashed lines in the graphs indicate the original values for the blue channel, and the solid lines indicate the estimated values for the blue channel. There are four main steps:

- (i) initial interpolation: each of the three channels is interpolated;
- (ii) high-frequency bands projection: each initially interpolated channel is subsampled into four subimages, then the high-frequency components of the unknown values are projected onto the high-frequency components of the corresponding known values;

- (iii) low-frequency band optimization: the low-frequency band components are enhanced by optimizing the weighted L_2 -norm of the chrominance, and high-frequency bands of red and blue channels are forced to equal the high-frequency bands of green channel;
- (iv) postprocessing: the estimated color values at the locations of the known color values are replaced by the observed color values, and the estimated color values at locations of the unknown color values are projected onto the range $[0, 255]$.

Note that the smooth regions in the low-frequency band of the initially interpolated image are not updated because an iterative update of a smooth region deforms it, as shown in Table 1. As the iteration number increases, degradation in the smooth regions increases. Also note that the low-frequency band of the green channel after it is interpolated in the initial stage is not updated in order to reduce complexity. Besides, in our framework, updating the green values leads to negligible improvements. The initial interpolation, moreover, helps speed up convergence in the optimization step. The main steps of the proposed algorithm are described in more detail in the following subsections.

2.1. Initial interpolation

As stated above, interpolation of the initial green values is an essential step. We use a modified edge-sensitive interpolation for the green values. While edge-sensitive algorithms

have proven to be effective in demosaicking [13, 15, 16], they have two drawbacks. First, they test whether each pixel belongs to a horizontal or vertical edge, and this test is not always accurate because it depends on the values in a single row or column. Second, because the difference between the vertical and horizontal colors is used to detect an edgy region (directional area), a small variation in colors can lead to a wrong decision, that is, nondirectional regions are likely to be considered directional regions. We overcome these two drawbacks by using an interpolation method with two modifications. The first is to use robust differentiation to determine whether the current pixel is in a nondirectional or directional (horizontal or vertical) region. This differentiation is done using a 3×5 (horizontal) or a 5×3 (vertical) mask. The second is to use a certain threshold, denoted by θ , that is used to determine the nondirectional regions. The algorithm for this more efficient interpolation method is as follows.

- (1) *Interpolate missing green values:* missing green values are interpolated using modified edge-sensitive interpolation. Each pixel is checked if it belongs to a pure horizontal edge, a pure vertical edge, or a nondirectional region using the following test:

- (a) at the blue positions (such as B_{43} in Figure 1),

$$G_{43} = \begin{cases} \frac{1}{2}[G_{33} + G_{53}] + \frac{1}{4}[2B_{43} - B_{23} - B_{63}] & \text{if } \Delta_H > \Delta_V + \theta, \\ \frac{1}{2}[G_{42} + G_{44}] + \frac{1}{4}[2B_{43} - B_{41} - B_{45}] & \text{else if } \Delta_V > \Delta_H + \theta, \\ \frac{1}{4}[G_{33} + G_{53} + G_{42} + G_{44}] & \\ \quad + \frac{1}{8}[4B_{43} - B_{23} - B_{63} - B_{41} - B_{45}] & \text{otherwise,} \end{cases} \quad (1)$$

where

$$\begin{aligned} \Delta_H &= \frac{1}{4}|2G_{33} - G_{31} - G_{35}| + \frac{1}{4}|2G_{53} - G_{51} - G_{55}| \\ &\quad + \frac{1}{2}|G_{42} - G_{44}| + \frac{1}{4}|2B_{43} - B_{41} - B_{45}| \\ &\quad + \frac{1}{2}|R_{32} - R_{34}| + \frac{1}{2}|R_{52} - R_{54}|, \\ \Delta_V &= \frac{1}{4}|2G_{42} - G_{22} - G_{62}| + \frac{1}{4}|2G_{44} - G_{24} - G_{64}| \\ &\quad + \frac{1}{2}|G_{33} - G_{53}| + \frac{1}{4}|2B_{43} - B_{23} - B_{63}| \\ &\quad + \frac{1}{2}|R_{32} - R_{52}| + \frac{1}{2}|R_{34} - R_{54}|. \end{aligned} \quad (2)$$

Testing in the diagonal direction is omitted because preliminary experiments showed that including a diagonal direction step in the test does not significantly improve the results;

- (b) the same procedure is used at the red positions, but blue pixels are replaced by red ones.

- (2) *Interpolate missing blue values:*

- (a) at the known red positions (such as R_{34} in Figure 1),

$$B_{34} = G_{34} + \frac{1}{4}[(B_{23} - G_{23}) + (B_{45} - G_{45}) + (B_{43} - G_{43}) + (B_{25} - G_{25})]; \quad (3)$$

- (b) at the known green positions (such as G_{33} and G_{44} in Figure 1),

$$\begin{aligned} B_{33} &= G_{33} + \frac{1}{2}[(B_{23} - G_{23}) + (B_{43} - G_{43})], \\ B_{44} &= G_{44} + \frac{1}{2}[(B_{43} - G_{43}) + (B_{45} - G_{45})]. \end{aligned} \quad (4)$$

- (3) *Interpolate missing red values:* follow the same steps as for the blue values.

2.2. High-frequency bands projection

Since there is high correlation between the high-frequency components [13], high-frequency bands projection is performed by replacing the high-frequency components of the unknown colors with those of the known colors. This is done by obtaining four subimages for each of the three channels. For example, the green channel is regarded to have two subimages corresponding to the known green values and two corresponding to the interpolated green values. These subimages are obtained by downsampling each channel (as shown in Figure 3). The high-frequency bands of the subimages corresponding to unknown values are replaced with the ones corresponding to known values. An example for the green channel that illustrates this step is shown in Figure 3. The high-frequency bands of the unknown green values are replaced with the high-frequency bands of the corresponding known red or blue values. The R' and B' indicate the interpolated red and blue values, respectively, and G'_R and G'_B indicate the interpolated green values at known red and blue positions, respectively. The estimated green values after high-frequency bands projection at the known red and blue positions are, respectively, denoted by \tilde{G}_R and \tilde{G}_B . Once the subimages for each channel are reconstructed, they are recombined to reconstruct the full channel.

2.3. Low-frequency band optimization

After projection of the high-frequency bands of unknown values for all three channels onto the high-frequency bands of known values, each channel is decomposed into low-frequency and high-frequency bands using filter banks. However, high-frequency bands are not changed; they are forced to equal the high-frequency bands of the green channel.

In the low-frequency band, the main goal is to smooth the low-frequency components of the chrominance in each region. In this aim, we classify regions into edgy or smooth regions so that the edgy regions are updated while the

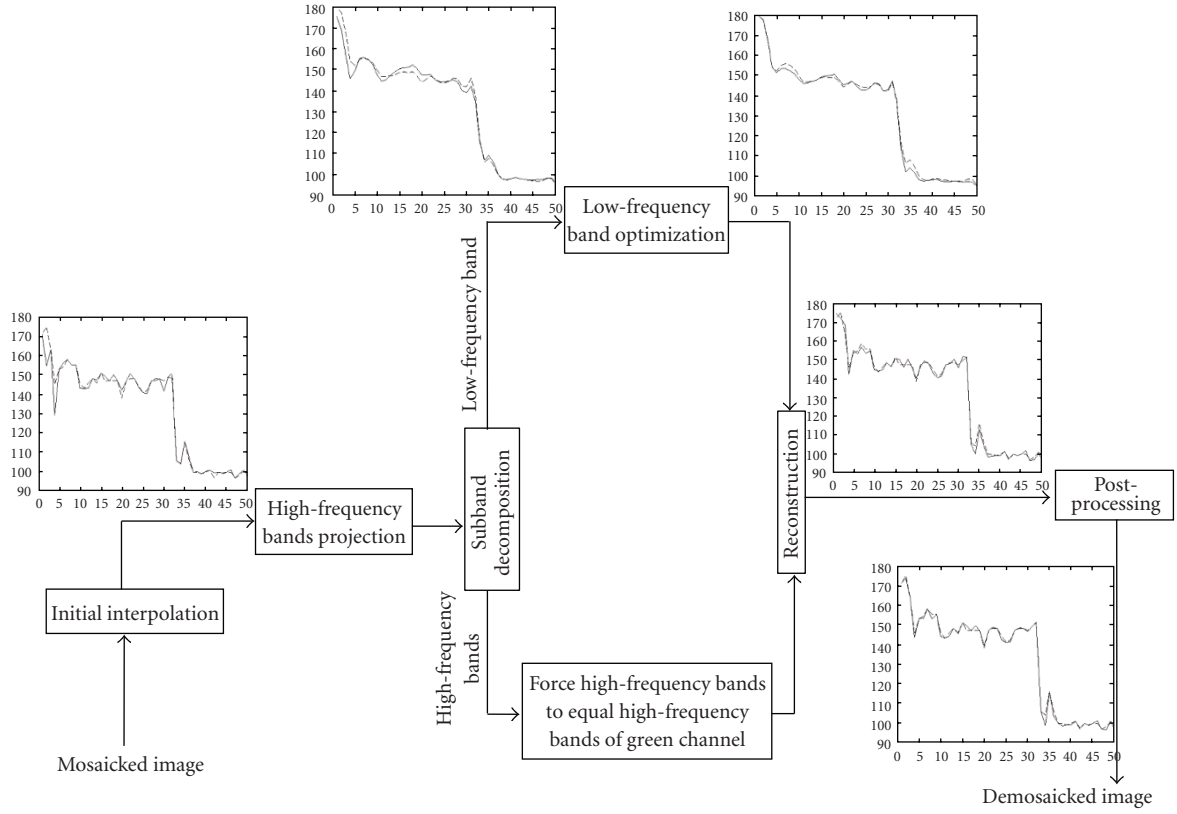


FIGURE 2: Illustrative example of the proposed algorithm.

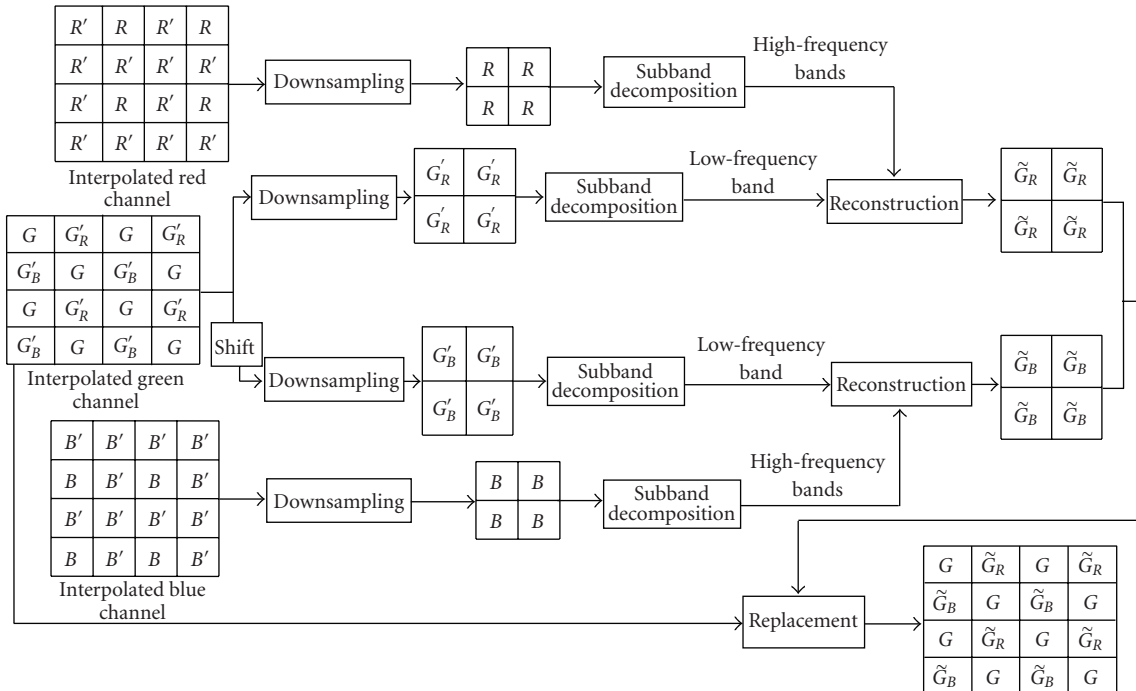


FIGURE 3: High-frequency bands projection.

TABLE 2: CMSE for test images.

Image	Proposed	Su	Li	POCS	Hirakawa	Zhang	Lu	Pei
1	11.5223	9.0701	9.5491	9.8627	20.1915	10.0229	48.4652	35.4014
2	6.6941	8.1973	7.1488	9.2067	8.2553	26.7206	14.0848	11.5806
3	4.1522	5.2607	5.1317	5.5116	4.9136	4.8436	8.7821	7.3658
4	8.6130	7.5209	8.1469	7.8791	14.2556	10.5272	29.0382	22.6715
5	10.9657	15.9342	14.8761	13.8602	18.6441	11.7424	42.4376	32.5822
6	8.6494	7.6680	8.1018	8.1252	11.4361	8.4658	34.7384	26.9961
7	4.0838	5.4185	4.6469	5.7639	5.8584	6.4146	9.7627	8.2171
8	17.6564	16.3399	16.9477	17.3349	27.2797	17.9737	93.4438	70.8990
9	3.5502	5.1127	4.2510	5.3213	4.3785	10.2184	8.0024	6.1832
10	5.1811	5.4941	6.1356	5.7750	8.1553	6.3844	14.4991	11.8602
11	7.3630	7.1778	8.0242	7.7097	11.4476	7.1311	25.4871	19.6494
12	22.2441	20.4843	24.6354	21.4891	32.9665	20.5274	52.7753	39.9807
13	24.1887	16.5642	21.3501	19.8854	47.3841	22.7666	84.4676	63.3374
14	14.8775	25.5082	22.8913	21.5165	19.0774	12.7567	32.0745	25.8959
15	8.1653	8.7452	8.7311	9.3909	10.9574	11.1007	14.1436	12.6439
16	3.9119	3.5109	4.2093	3.6718	4.6243	3.1003	15.6243	12.2054
17	4.5278	4.2685	5.1082	4.5449	7.5514	4.3734	13.4949	10.6898
18	3.7819	4.1147	4.0167	4.6106	4.9912	3.2537	12.2795	9.4206
19	6.3387	5.9186	6.5390	6.4372	9.5870	7.0495	35.0623	24.1324
20	3.9205	4.1727	4.3868	4.5715	5.7961	4.5516	12.1711	9.1130
Average	9.0194	9.3241	9.7414	9.6234	13.8875	10.4962	29.9683	22.1695

TABLE 3: S-CIELAB metric (ΔE_{ab}) for test images.

Image	Proposed	Su	Li	POCS	Hirakawa	Zhang	Lu	Pei
1	1.2580	1.1994	1.2554	1.2675	1.2655	1.1808	1.8778	1.9168
2	0.7083	0.8288	0.7450	0.8854	0.8313	1.2427	1.0470	0.8619
3	0.5374	0.5989	0.5818	0.6075	0.5677	0.5422	0.7498	0.6107
4	0.9426	0.9227	0.9702	0.9635	1.0072	0.9778	1.3082	1.2325
5	1.0957	1.4254	1.3662	1.2927	1.3019	1.1403	1.8354	1.4773
6	0.9117	0.9093	0.9428	0.9532	0.8705	0.8735	1.4045	1.3525
7	0.6378	0.7329	0.6922	0.7569	0.6900	0.6872	0.8841	0.7282
8	1.4321	1.5043	1.5116	1.5459	1.4312	1.3867	2.5748	2.3764
9	0.5247	0.6356	0.5834	0.6365	0.6035	0.6377	0.7522	0.5644
10	0.5716	0.5940	0.6019	0.6146	0.6407	0.6129	0.8247	0.6970
11	0.8083	0.8204	0.8466	0.8576	0.8478	0.7641	1.2469	1.1156
12	1.1313	1.2049	1.2314	1.2045	1.3084	1.1308	1.5831	1.2792
13	1.7115	1.5818	1.7135	1.6925	1.9018	1.6234	2.4384	2.1515
14	1.0650	1.3373	1.2697	1.2550	1.1750	0.9907	1.5685	1.2791
15	0.6859	0.7571	0.7444	0.7872	0.8015	0.7427	0.9122	0.7302
16	0.6289	0.6309	0.6593	0.6548	0.5812	0.5611	0.9866	0.9360
17	0.5605	0.5566	0.5742	0.5806	0.6017	0.5265	0.7950	0.6928
18	0.6188	0.6497	0.6429	0.6904	0.6255	0.5654	0.8905	0.7698
19	0.7996	0.8211	0.8219	0.8675	0.8637	0.8159	1.4199	1.2509
20	0.5880	0.6076	0.6135	0.6427	0.6224	0.5875	0.8548	0.7108
Average	0.8609	0.9159	0.9184	0.9378	0.9269	0.8795	1.2977	1.1367



FIGURE 4: Original images (numbered from left-to-right and top-to-bottom).

smooth regions are not. The classification is based on “edge indicators,” which are coefficients that indicate existence of edges at certain pixel positions as will be discussed later. If the average number of edge indicators within a certain window size $[(2w + 1) \times (2w + 1)]$ centered at location (i, j) is less than a certain threshold (θ_1) , this pixel location belongs to an edgy region; otherwise it belongs to a smooth one. The classification is represented by

$$\begin{aligned} C_L &\in \mathfrak{R}_E & \text{if } e_{av} < \theta_1, \\ C_L &\in \mathfrak{R}_S & \text{otherwise,} \end{aligned} \quad (5)$$

where

$$\begin{aligned} e_{av}(i, j) &= \frac{\sum_{l=-w}^w \sum_{m=-w}^w e_{i,j}^{l,m}}{(2w + 1)^2}, \\ e_{i,j}^{0,0} &= \frac{1}{1 + |HL_{i,j}| + |LH_{i,j}| + |HH_{i,j}|}. \end{aligned} \quad (6)$$

The C_L denotes either the R_L , G_L , or B_L , which is the low-frequency band component of the red, green, or blue channel, respectively; w is a parameter that determines window

size; \mathfrak{R}_E and \mathfrak{R}_S represent edgy and smooth regions, respectively; $HL_{i,j}$, $LH_{i,j}$, and $HH_{i,j}$ are coefficients of the high-frequency bands at position (i, j) ; and $e_{i,j}^{l,m}$ is a weight representing the edge indicator at position $(i + l, j + m)$.

The main goal is to smooth the low-frequency components of the chrominance in the edgy regions. To do this, we propose to consider only pixel locations which belong to an edgy region (\mathfrak{R}_E) and minimize the following cost function which is based on region-adaptive weights to avoid smoothness across edges:

$$\begin{aligned} J[\underline{R}_L, \underline{B}_L] &= \sum_{l=-P}^P \sum_{m=-P}^P [(\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb})^T W^{l,m} (\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb}) \\ &\quad + (\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr})^T W^{l,m} (\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr})] \\ &\quad \forall \underline{R}_L, \underline{B}_L \in \mathfrak{R}_E, \end{aligned} \quad (7)$$

where

$$W^{l,m} = \text{diag}(\tilde{\mathcal{E}}^{l,m}), \quad (8)$$

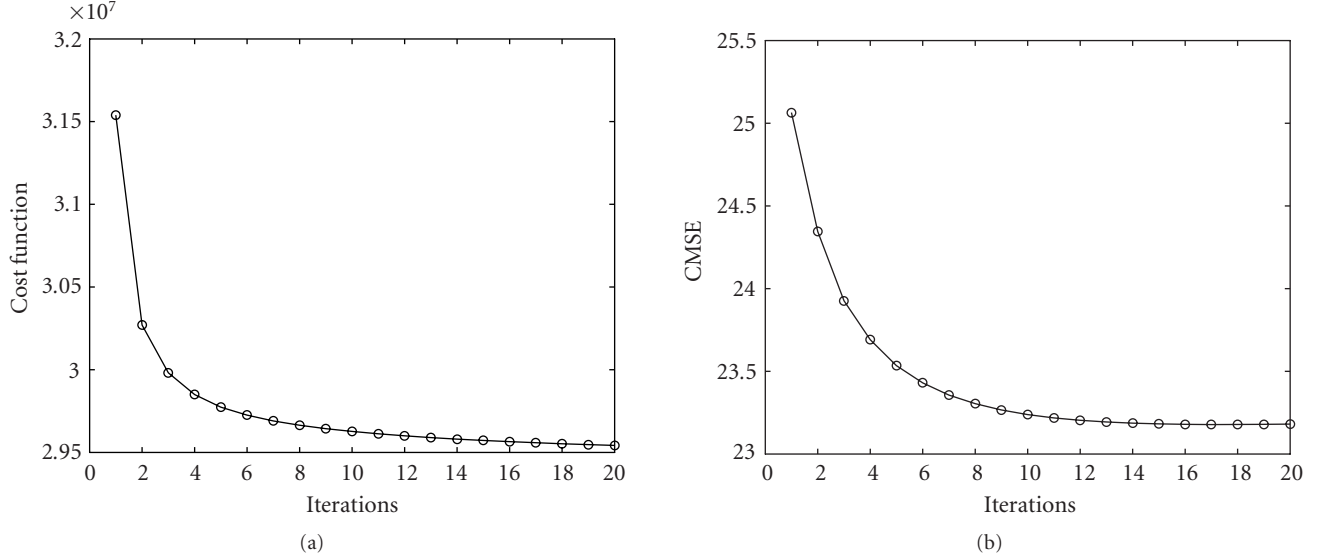


FIGURE 5: (a) Convergence of cost function; (b) corresponding convergence of CMSE.

and S_x^l and S_y^m are shifting operators in directions x and y by l and m , respectively. $W^{l,m}$ is the normalized edge indicator matrix, which is a diagonal matrix consisting of elements

$$\tilde{e}_{i,j}^{l,m} = \frac{e_{i,j}^{l,m}}{\sum_{l=-1}^1 \sum_{m=-1}^1 e_{i,j}^{l,m}} \quad (9)$$

in lexicographical order; \underline{X}_{cb} and \underline{X}_{cr} are the chrominance rearranged in lexicographical order:

$$\begin{aligned} \underline{X}_{cb} &= -0.169\underline{R}_L - 0.331\underline{G}_L + 0.5\underline{B}_L, \\ \underline{X}_{cr} &= 0.5\underline{R}_L - 0.419\underline{G}_L - 0.081\underline{B}_L. \end{aligned} \quad (10)$$

In the low-frequency band, full-color image enhancement is performed by optimizing J with respect to \underline{R}_L and \underline{B}_L . Specifically, the recursion is given by

$$\underline{C}_L^{k+1} = \underline{C}_L^k - \beta_C^k \nabla_{\underline{C}_L}^k J, \quad (11)$$

where $\nabla_{\underline{C}_L} J$ is the gradient of J with respect to \underline{C}_L , C represents a color channel (R or B), β_C is a scalar representing the step size in the direction of the gradient of \underline{C}_L , and superscript k represents the k th iteration. The gradient with respect to channel \underline{C}_L is

$$\begin{aligned} \nabla_{\underline{C}_L} J &= 2 \sum_{l=-P}^P \sum_{m=-P}^P (I - S_x^{-l} S_y^{-m}) W^{l,m} \\ &\times \left[k_{cr}(n) (\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr}) + k_{cb}(n) (\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb}) \right], \end{aligned} \quad (12)$$

where I is the identity matrix, β_C^k is determined by minimizing the function $J(\underline{C}_L^{k+1}) = J(\underline{C}_L^k - \beta_C^k \nabla_{\underline{C}_L}^k J)$ [25] as follows:

$$\begin{aligned} J(\underline{C}_L^{k+1}) &= \sum_{l=-P}^P \sum_{m=-P}^P \left((\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb}) \right. \\ &\quad \left. - k_{cb}(n) \beta_C^k (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J) \right)^T W^{l,m} \\ &\times \left((\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb}) \right. \\ &\quad \left. - k_{cb}(n) \beta_C^k (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J) \right) \\ &+ \left((\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr}) \right. \\ &\quad \left. - k_{cr}(n) \beta_C^k (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J) \right)^T W^{l,m} \\ &\times \left((\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr}) \right. \\ &\quad \left. - k_{cr}(n) \beta_C^k (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J) \right) \\ &= \sum_{l=-P}^P \sum_{m=-P}^P \left(\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb} \right)^T W^{l,m} (\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb}) \\ &+ \left(\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr} \right)^T W^{l,m} (\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr}) \\ &+ (\beta_C^k k_{cb}(n))^2 (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J)^T \\ &\times W^{l,m} (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J) \\ &+ (\beta_C^k k_{cr}(n))^2 (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J)^T \\ &\times W^{l,m} (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J) \\ &- 2\beta_C^k k_{cb}(n) (\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb})^T \\ &\times W^{l,m} (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J) \\ &- 2\beta_C^k k_{cr}(n) (\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr})^T \\ &\times W^{l,m} (\nabla_{\underline{C}_L}^k J - S_x^l S_y^m \nabla_{\underline{C}_L}^k J). \end{aligned} \quad (13)$$

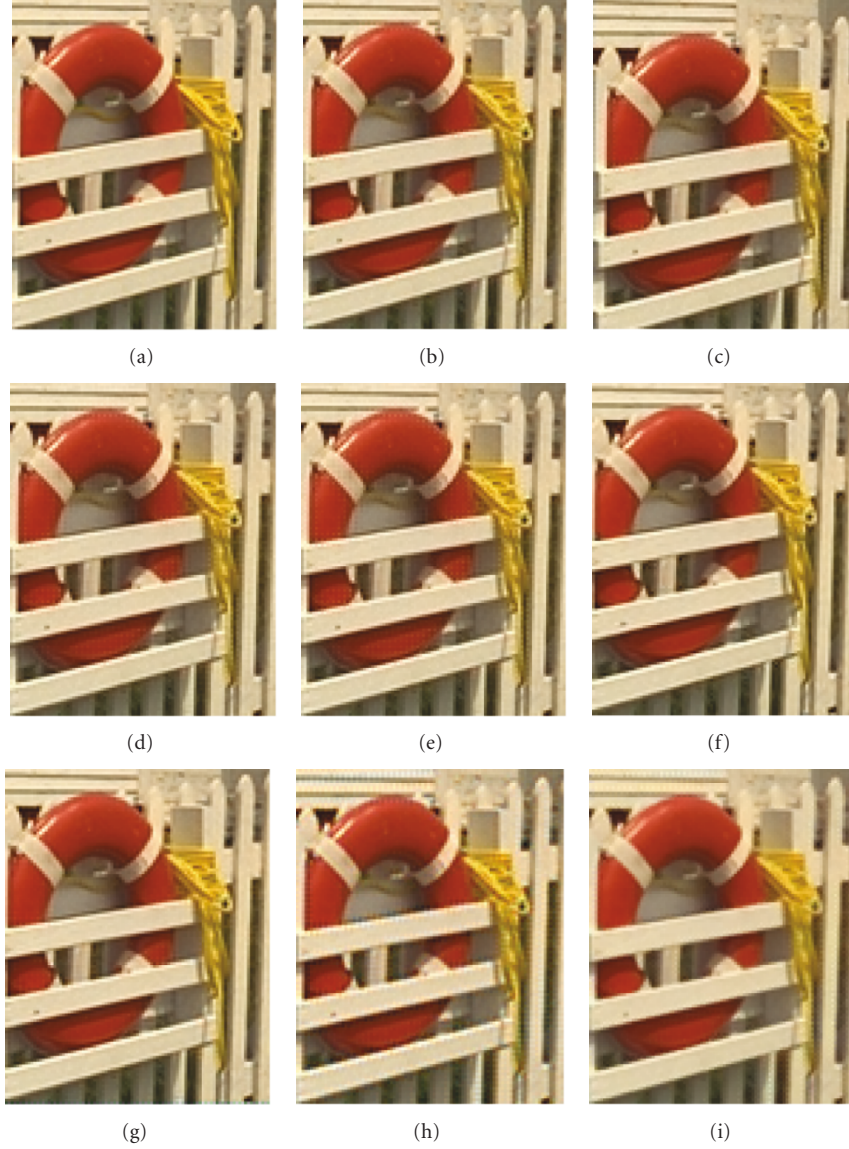


FIGURE 6: Part of image 19 containing smooth region: (a) original, (b) proposed, (c) POCS [14], (d) Su [16], (e) Li [15], (f) Hirakawa [9], (g) Zhang [10], (h) Pei [4], and (i) Lu [3] algorithms.

By differentiating this function with respect to β_C^k and then letting this differentiation equal zero, we can obtain β_C^k as follows:

$$\begin{aligned} \frac{\partial J(\underline{C}_L^{k+1})}{\partial \beta_C^k} = & -2 \sum_{l=-P}^P \sum_{m=-P}^P k_{cb}(n) (\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb})^T \\ & \times W^{l,m} (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J) \\ & - 2k_{cr}(n) (\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr})^T \\ & \times W^{l,m} (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J) \\ & + 2(k_{cb}(n))^2 \beta_C^k (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J)^T \\ & \times W^{l,m} (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J) \end{aligned}$$

$$\begin{aligned} & + 2(k_{cr}(n))^2 \beta_C^k (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J)^T \\ & \times W^{l,m} (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J) \\ = & 2 \sum_{l=-P}^P \sum_{m=-P}^P \beta_C^k ((k_{cb})^2 + (k_{cr})^2) \\ & \times (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J)^T W^{l,m} (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J) \\ & - 2k_{cb}(n) (\underline{X}_{cb} - S_x^l S_y^m \underline{X}_{cb})^T \\ & \times W^{l,m} (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J) \\ & - 2k_{cr}(n) (\underline{X}_{cr} - S_x^l S_y^m \underline{X}_{cr})^T \\ & \times W^{l,m} (\underline{\nabla}_{C_L}^k J - S_x^l S_y^m \underline{\nabla}_{C_L}^k J) = 0. \end{aligned} \tag{14}$$

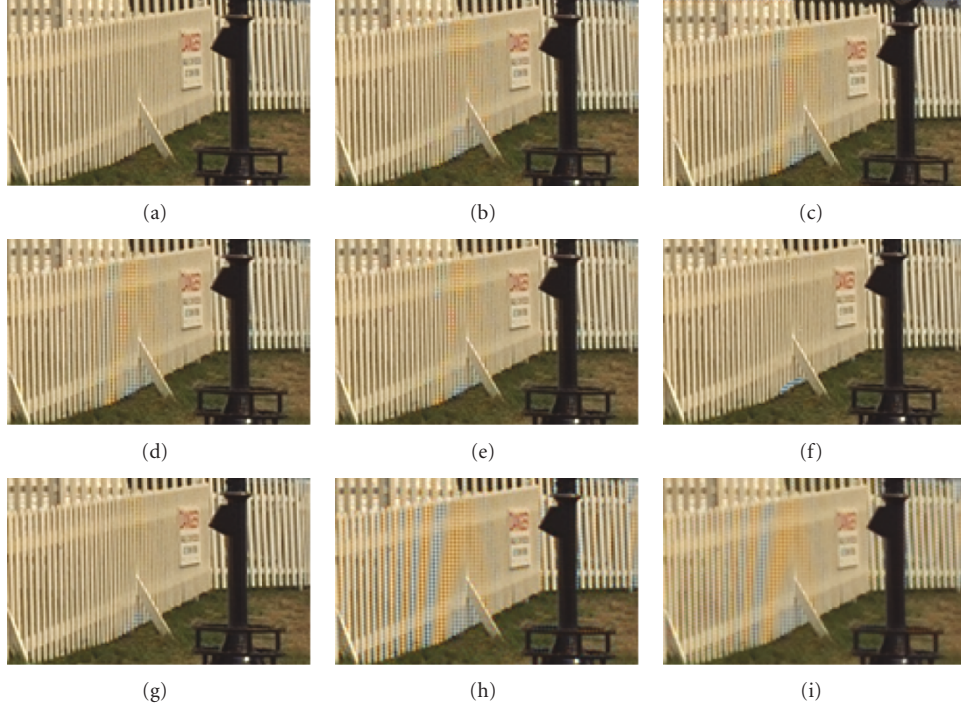


FIGURE 7: Part of image 19 containing edgy region: (a) original, (b) proposed, (c) POCS [14], (d) Su [16], (e) Li [15], (f) Hirakawa [9], (g) Zhang [10], (h) Pei [4], and (i) Lu [3] algorithms.

Therefore,

$$\beta_C^k = \frac{Q_1}{Q_2}, \quad (15)$$

where $Q_1 = \sum_{l=-P}^P \sum_{m=-P}^P (\nabla_{C_L}^k J)^T (I - S_x^{-l} S_y^{-m}) W^{l,m} (I - S_x^l S_y^m) (k_{cb}(n) \underline{X}_{cb} + k_{cr}(n) \underline{X}_{cr})$, $Q_2 = \sum_{l=-P}^P \sum_{m=-P}^P (k_{cb}^2(n) + k_{cr}^2(n)) (\nabla_{C_L}^k J)^T (I - S_x^{-l} S_y^{-m}) W^{l,m} (I - S_x^l S_y^m) \nabla_{C_L}^k J$, $k_{cb}(n)$ and $k_{cr}(n)$ are the coefficients in the n th term used to obtain \underline{X}_{cb} and \underline{X}_{cr} , respectively, as in (10); n equals 1 or 3 when C equals R or B , respectively.

2.4. Postprocessing

After K iterations of the optimization step, the cost function converges. The full-color channels are then reconstructed using the optimized low-frequency band and the projected high-frequency bands. After these two steps, the estimated values at the locations of the observed values are replaced by the observed ones. Also, due to the assumption that the color values are sampled using eight bits, the fully reconstructed image has to be projected onto the range $[0, 255]$:

$$\underline{C} = \mathcal{P}_{c0}(\mathcal{P}_{c1}(\underline{C}^K)), \quad (16)$$

where

$$\begin{aligned} \mathcal{P}_{c0}(\underline{C}) &= (I - D_{i,j}^* D_{i,j}) \underline{C} + D_{i,j}^* D_{i,j} \underline{C}, \\ (\mathcal{P}_{c1}(\underline{C}))_i &= \begin{cases} 0 & \text{if } \underline{C}_i < 0, \\ \underline{C}_i & \text{if } 0 \leq \underline{C}_i \leq 255, \\ 255 & \text{if } \underline{C}_i > 255. \end{cases} \end{aligned} \quad (17)$$




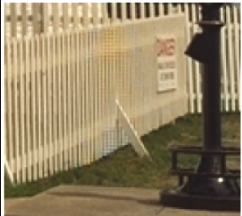
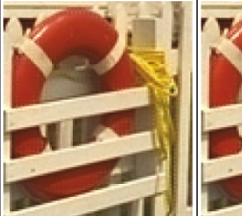
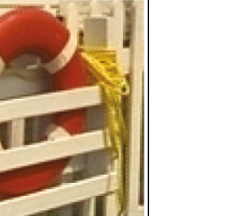
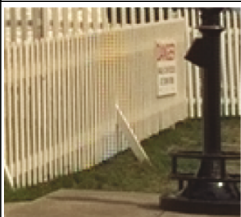


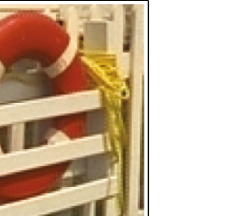
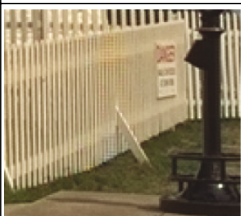
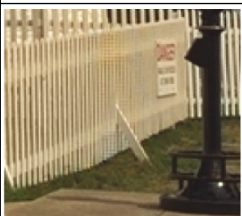

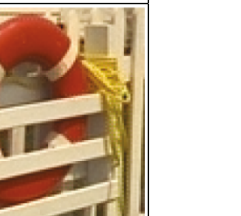
\underline{C} refers to color channels \underline{R} and \underline{B} , $D_{i,j}$ is the downsampling operator used to sample the pixels at locations $(2m+i, 2n+j)$, where $m = 0, \dots, (M/2) - 1$ and $n = 0, \dots, (N/2) - 1$; M and N are assumed to be even numbers without loss of generality, and $D_{i,j}^*$ is the adjoint of $D_{i,j}$, that is, the upsampling operator. The projection $\mathcal{P}_{c1}(\underline{C})$ is performed by replacing values in \underline{C} greater than 255 by 255 and values less than 0 by 0.

3. SIMULATION RESULTS

We tested our algorithm using 20 photographic images (test images are obtained from <http://r0k.us/graphics/kodak>); see Figure 4. We compared the results with those of seven state-of-the-art demosaicing algorithms: the Su [16], Li [15], POCS [14], Hirakawa [9], Zhang [10], Lu [3], and Pei [4] algorithms. We compared the performance of these algorithms from three aspects. First, we compared their demosaicked images using two objective quality measures: the color mean square error (CMSE) metric and the S-CIELAB metric (ΔE_{ab}^*) [26]. We then compared their demosaicked images subjectively. Finally, we compared the computational complexity of the proposed algorithm with that of the other iterative algorithms [14–16] and with that of the optimal demosaicking solutions [9, 10].

For all simulation runs, we used $\theta = 15$ and $\theta_1 = 0.04$, and the iteration number was five ($K = 5$). We used the same filter banks for both the proposed algorithm and the alternative projection algorithm [14]. The low-pass and high-pass filters for decomposition and reconstruction were defined

TABLE 4: Effects of threshold (θ_1) and iteration number on edgy and smooth regions.

Iterations	Edgy region		Smooth region	
	$\theta_1 = \infty$	$\theta_1 = 5$	$\theta_1 = \infty$	$\theta_1 = 5$
0				
10				
20				
30				

as

$$\begin{aligned}
 h_L &= [1 \ 2 \ 1]/4, \\
 h_H &= [1 \ -2 \ 1]/4, \\
 g_L &= [-1 \ 2 \ 6 \ 2 \ -1]/8, \\
 g_H &= [1 \ 2 \ -6 \ 2 \ 1]/8,
 \end{aligned} \tag{18}$$

where h_L and h_H are the decomposition low-pass and high-pass filters, respectively, and g_L and g_H are the reconstruction ones, respectively.

3.1. Quality measure

Tables 2 and 3 show the CMSE and S-CIELAB metric results:

$$\text{CMSE} = \frac{\sum_{x=1}^M \sum_{y=1}^N \sum_{c=R,G,B} [I_o^c(x, y) - I_r^c(x, y)]^2}{3 \times M \times N}, \tag{19}$$

where M and N are the image dimensions, and I_o^R and I_r^R are the original and reconstructed red channels. The smallest values in each row are shown in bold. The proposed algorithm had the best average CMSE performance and the best average S-CIELAB performance.

The convergence of the cost function and the CMSE is shown in Figure 5. Five iterations were enough for convergence.

To further demonstrate the effectiveness of the proposed algorithms, we compared the results for three parts of image 19, the *lighthouse*. One part includes edges, one includes a smooth region, and the third includes thin, low-contrast edges. Most demosaicking algorithms work well in smooth regions, but some of the iterative algorithms [14–16] deform smooth regions as shown in Figure 6, and thus produce lower visual quality than the other algorithms in the smooth regions. In this figure, we can see that the proposed algorithm overcomes the problem of the iterative algorithms in smooth regions. In edgy regions, however, the Hirakawa [9] and Zhang [10] algorithms achieve the best visual results, as shown in Figure 7. In the low-contrast, edgy regions, the proposed algorithm achieved better visual results, as shown in Figure 8. This is because the Hirakawa and Zhang algorithms try to overcome discontinuity in the thin edges by smoothing which leads to blurred edges. In short, in terms of CMSE and S-CILAB measurements, the proposed algorithm is the best although the visual results for edgy regions have fewer

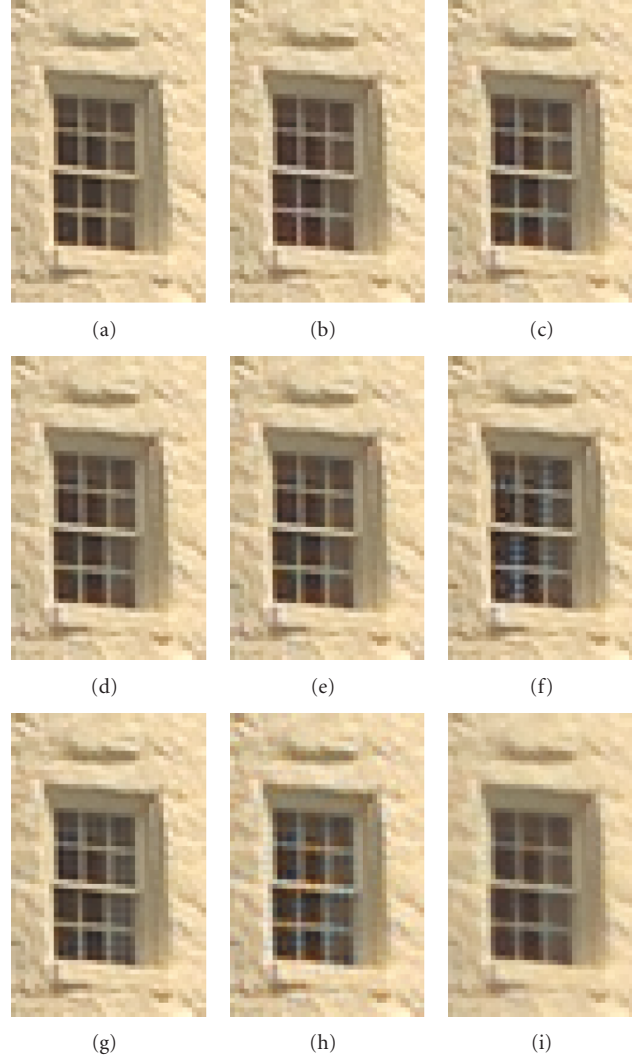


FIGURE 8: Part of image 19 containing low-contrast, edgy region: (a) original, (b) proposed, (c) POCS [14], (d) Su [16], (e) Li [15], (f) Hiraakawa [9], (g) Zhang [10], (h) Pei [4], and (i) Lu [3] algorithms.

artifacts with the Hirakawa [9] and Zhang [10] algorithms. These algorithms fail in regions that contain thin, low-contrast edges. This suggests the need to use subjective evaluation along with objective measures.

Table 4 shows the effects of the threshold and number of iterations in our algorithm. We used a part of the *light-house* image containing both smooth and edgy regions and performed 30 iterations. The table shows the resulting images with and without the classification step (i.e., threshold θ_1 equals 10 and ∞ , resp.). Without using the classification step, the smooth region is deformed as the number of iterations increase.

3.2. Complexity

We compared the complexity of the proposed algorithm with that of the other iterative algorithms [14–16] and the optimal demosaicking solutions [9, 10]. For our algorithm, we

used $P = 1$ in (7), and we set the number of iterations for all the iterative algorithms to five, which is enough for most of the test images. For Hirakawa's algorithm [9], we used two iterations in the postprocessing step as suggested by the author. For Zhang's algorithm [10], we used a constant, pre-computed, low-pass filter as used in the default parameters in the author's MATLAB code. Therefore, in the complexity computation, we did not include the computations required to determine the adaptive filter.

Due to the similarity of the computations required for red and blue values, the computational load for the proposed algorithm is greatly reduced, as shown in Table 5. The total number of addition operations required for the proposed algorithm is about 50% of that required by POCS [14], and the number of multiplication operations required is about 45% of that required by POCS. However, the proposed algorithm increases the total number of absolute evaluation and comparison operations from 1 MN to 6 MN and from 2 MN to 9

TABLE 5: Computation complexity.

Algorithm	Steps	Addition	Multiplications	Comparison	Absolute	Division	LUTs ⁽¹⁾
Proposed	Initial	25.67 MN	12 MN	1 MN	6 MN	—	—
	Update green	24 MN	33 MN	—	—	—	—
	Weight eval.	11 MN	—	—	3 MN	2 MN	—
	1 iteration	23 MN	20 MN	1 MN	—	—	—
	Total (5 iter.)	199.67 MN	181 MN	6 MN	9 MN	2 MN	—
	Exact total ⁽²⁾	115.5 MN	92.7 MN	3.3 MN	9 MN	2 MN	—
Su [16]	Initial	8.5 MN	4.67 MN	1 MN	—	—	—
	Refinement	11 MN	2 MN	—	—	—	—
	1 Iteration	23 MN	5 MN	—	—	—	—
	Total (5 iter.)	134.5 MN	31.67 MN	1 MN	2 MN	—	—
Li [15]	Initial	11 MN	6.17 MN	2 MN	1 MN	—	—
	Update	23 MN	5 MN	—	—	—	—
	Termination	13 MN	7 MN	3 MN	—	—	—
	Total 1 Iter.	36 MN	12 MN	3 MN	—	—	—
	Total (5 iter.)	191 MN	66.17 MN	16 MN	2 MN	—	—
POCS [14]	Initial	9.3 MN	3.6 MN	1 MN	2 MN	—	—
	Update green	36 MN	36 MN	—	—	—	—
	1 iteration	103 MN	75 MN	2 MN	—	—	—
	Total (5 iter.)	560.3 MN	414.6 MN	11 MN	2 MN	—	—
Hirakawa [9]	Interpolation	15 MN	5 MN	—	—	—	—
	Color space conv.	16 MN	18 MN	—	—	—	6 MN
	Homogeneity	48 MN	24 MN	49 MN	12 MN	—	—
	Adaptive parameters	0 MN	0 MN	6 MN	—	—	—
	1 iter. of post-proc.	9 MN	1 MN	16 MN	—	—	—
	Total (2 iter.)	97 MN	49 MN	87 MN	12 MN	—	6 MN
Zhang [10]	Initial	26 MN	28 MN	—	—	—	—
	Green	20 MN	12 MN	—	—	1.5 MN	—
	Red and Blue	12 MN	1.5 MN	—	—	—	—
	Post-proc.	—	—	2 MN	—	—	—
	Total	58 MN	41.5 MN	2 MN	—	1.5 MN	—

⁽¹⁾LUTs: look-up tables operation.

⁽²⁾Exact total: the total number of operations when we exclude the smooth regions from computations.

MN, respectively, where M and N are the image dimensions. In addition, the proposed algorithm required 2 MN division operations to evaluate the weights and to weight the chrominance. The overall operations required by the proposed algorithm is about 50% of that required by POCS. Moreover, one iteration of the proposed algorithm requires only 44 MN operations which is less than the number required by both the alternative projection (144 MN operations) [14] and successive approximation (51 MN operations) [15] algorithms. This means that, if the number of iterations increases, the cost of the proposed algorithm will be less than that of both algorithms. Moreover, if we take into consideration the fact that the proposed algorithm performs the iteration step for only the edgy regions rather than the whole image, we see why the number of computations required by the proposed algorithm is reduced so much, as shown in the last row of Table 5. Experimentally, we found that only 47.7% of the total area of the 20 test images is edgy.

4. CONCLUSION

We have developed an iterative demosaicking algorithm based on optimization and projection in different frequency bands. For the low-frequency band, the assumption that chrominance varies slowly in an object region is used in the quadratic cost function minimization to enhance the initially interpolated image. For the high-frequency bands, projection of the high-frequency bands of the estimated values onto the high-frequency bands of the corresponding observed values is used. The projection in the high-frequency bands is based on the observation that high-frequency bands for different color channels are highly correlated. Comparison of the performance of the proposed algorithm with that of seven state-of-the-art demosaicking algorithms showed that the proposed algorithm outperforms all of them on average for 20 images in terms of objective quality. Moreover, it is competitive with these algorithms from the subjective

quality and complexity points of view. The proposed algorithm overcomes the problem some iterative algorithms have in smooth regions, the problem noniterative algorithms have in edgy regions, and the problem some optimal demosaicking solutions have in the low-contrast, edgy regions.

ACKNOWLEDGMENTS

The authors thank Dr. Bahadır K. Gunturk of Louisiana State University for sending them the MATLAB code for his algorithm and Dr. Xin Li at Western Virginia University, Dr. Chung-Yen Su at National Taiwan Normal University, Dr. Lei Zhang at Polytechnic University, Hong Kong, and Dr. Keigo Hirakawa at Cornell University for making the MATLAB code for their algorithms available on their web pages. Also, the authors would like to thank the reviewers for their useful comments which have greatly improved the paper. This work was supported in part by the Ministry of High Education, Egypt, and in part by the Telecommunications Advancement Foundation, Japan.

REFERENCES

- [1] B. E. Bayer, "Color imaging array," US patent no. 3 971 065, July 1976.
- [2] D. Su and P. Willis, "Demosaicing of colour image using pixel level data-dependent triangulation," in *Proceedings of the Theory and Practice of Computer Graphics*, pp. 16–23, Birmingham, UK, June 2003.
- [3] 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, 2003.
- [4] 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, 2003.
- [5] M. R. Gupta and T. Chen, "Vector color filter array demosaicing," in *Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications II*, vol. 4306 of *Proceedings of SPIE*, pp. 374–382, San Jose, Calif, USA, January 2001.
- [6] L. Chang and Y.-P. Tan, "Adaptive color filter array demosaicking with artifact suppression," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '04)*, vol. 3, pp. 937–940, Vancouver, BC, Canada, May 2004.
- [7] R. Lukac, K. Martin, and K. N. Plataniotis, "Demosaicked image post-processing using local color ratios," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 914–920, 2004.
- [8] D. D. Muresan and T. W. Parks, "Demosaicing using optimal recovery," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 267–278, 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, 2005.
- [10] 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, 2005.
- [11] K. Hirakawa and T. W. Parks, "Joint demosaicing and denoising," *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2146–2157, 2006.
- [12] R. Kimmel, "Demosaicing: image reconstruction from color CCD samples," *IEEE Transactions on Image Processing*, vol. 8, no. 9, pp. 1221–1228, 1999.
- [13] B. 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, 2002.
- [14] M. Gevrekci, B. K. Gunturk, and Y. Altunbasak, "POCS-based restoration of bayer-sampled image sequences," in *Proceedings of IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP '07)*, vol. 1, pp. 753–756, Honolulu, Hawaii, USA, April 2007.
- [15] X. Li, "Demosaicing by successive approximation," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 370–379, 2005.
- [16] 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, 2006.
- [17] K. H. Chung and Y. E. Chan, "Color demosaicing using variance of color differences," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 2944–2955, 2006.
- [18] 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, 2004.
- [19] R. Lukac and K. N. Plataniotis, "Data adaptive filters for demosaicking: a framework," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 560–570, 2005.
- [20] 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, 2005.
- [21] B. C. de Lavarène, D. Alleysson, and J. Hérault, "Practical implementation of LMMSE demosaicing using luminance and chrominance spaces," *Computer Vision and Image Understanding*, vol. 107, no. 1-2, pp. 3–13, 2007.
- [22] 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, 2005.
- [23] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Robust shift and add approach to super-resolution," in *Applications of Digital Image Processing XXVI*, vol. 5203 of *Proceedings of SPIE*, pp. 121–130, San Diego, Calif, USA, August 2003.
- [24] 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, 2006.
- [25] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, New York, NY, USA, 2nd edition, 2006.
- [26] <http://white.stanford.edu/~brian/scielab/scielab.html>.