Adaptive Color Filter Array Demosaicking Based on Constant Hue and Local Properties of Luminance

Chun-Hsien Chou¹, Kuo-Cheng Liu^{1,2,*}, and Wei-Yu Lee¹

¹ Department of Electrical Engineering, Tatung University, Taiwan

² Foreign Language and Information Educating Center, Taiwan Hospitality and Tourism
kcliu@ttht.edu.tw*

Abstract. Most commercial digital cameras use a single electronic sensor overlaid with a color filter array (CFA) to capture imagery. Since only one primary color is sampled in each pixel, the missing color primaries must be reconstructed by interpolation. In this paper, an adaptive demosaicking scheme for CFA interpolation is proposed. The scheme uses intra-channel correlation, color difference correlation, constant hue, and luminance-color difference correlation is proposed. A rough interpolation is first implemented by bilinear interpolation. Then the color difference correlation and constant hue are successively used to update the missing color primaries. To obtain high quality color images, an adaptive algorithm using luminance-color difference correlation and the information of edge direction is iteratively applied to improve the image quality around the edges. Simulation results demonstrate that the image quality of the proposed algorithm is better than that of the approach using color difference correlation in terms of peak signal-to-noise ratio (PSNR).

Keywords: Color filter array, demosaicking, hue, luminance.

1 Introduction

Digital still cameras have become indispensable in people's life and have been widely used as image input devices. To reduce cost and size, most commercial digital cameras use a single electronic sensor overlaid with a color filter array to capture imagery. Since only one primary color is sampled in each pixel, the missing color primaries must be reconstructed by interpolation sampled color primaries of the adjacent pixels. This color plane interpolation is known as demosaicking or CFA interpolation. Bayer color filter array is a popular format for digital acquisition of color images. The Bayer pattern [1] is shown in Fig. 1. Half of the total numbers of pixels are green, while a quarter of the total number is assigned to both red and blue. More pixels are dedicated to green than to red and blue, because the human eye is more sensitive to that color.

An immense number of demosaicking methods have been proposed in the literature [2]-[13]. To obtain more visually pleasing results, many adaptive CFA demosaicking methods that exploit the spectral and spatial correlations among neighboring

Fig. 1. Bayer pattern

Fig. 2. Reference Bayer CFA pattern

pixels were proposed. In [2], Pei and Tam present a color difference (green-red and green-blue) model for demosaicking technique. Based on this observation, several schemes (e.g., [3], [4]) have been devised to estimate missing color values. These methods normally do not perform satisfactorily around sharp edges and fine details. Besides color difference rule, many methods make use of color ratios [5], [6]. The ratios between the red and green values are highly similar, so are the ratios between the blue and green values. In [7], Gunturk *et al.* have proposed an effective scheme to exploit spectral correlation by alternately projecting the estimates of the missing color values onto constraint sets and spectral correlation. In general, the spatial correlation of neighboring pixels around edges is used to perform color interpolation. In [8], [9], several edge classifiers are proposed to identify the best directions for interpolating the missing color values. In [10], [11], authors attempt to estimate the luminance of an image and obtain good demosaicking image based on the luminance and chrominance signals of a CFA image in the frequency domain.

In this paper, the proposed demosaicking algorithm can be divided into four parts. The first part is an initial process using bilinear interpolation. The second part is designed to update the green channel by the constant color difference model. In the third part, red and blue channels are respectively updated by using the constant hue model. The fourth part is an adaptively iterative update algorithm that is developed based on luminance-color difference model and local properties. The paper is organized as follows. The interpolation algorithms for demosaicking are reviewed in Section 2. In Section 3, three image models including the constant color difference model, the constant hue model, and the luminance-color difference model are presented for the design of the proposed demosaicking algorithm. The proposed demosaicking algorithm is described including a CFA interpolation and an adaptively iterative update method in Section 4. The CFA interpolation is based on the constant color difference model and the constant hue model. The adaptively iterative update algorithm improves the image quality in edge regions using the luminance-color difference model and the edge information. Simulation results and conclusions are respectively described in Section 5 and 6.

2 Reviews of Interpolation Algorithms

In this section, the interpolation algorithms used in CFA are discussed. Common interpolation algorithms can be classified to two categories: non-adaptive and adaptive. Non-adaptive methods have low complexity and easy to be implemented because all pixels are processed in the same method. Adaptive methods use special method to process pixels on the edge. These algorithms can improve the artifact and increase image quality.

Bilinear interpolation is the simplest method. This method determines the value of a missing pixel based on a weighted average of the adjacent pixels in the CFA image. It is not considered good enough for photo quality images. Take the Bayer CFA pattern shown in Fig. 2 for example. The average of the upper, lower, left, and right pixel values is assigned as the G value of the interpolated pixel. The average of two adjacent pixel values in corresponding color is assigned to the interpolated pixel at green position.

$$G7 = \frac{G3 + G6 + G8 + G11}{4} \text{ and } R6 = \frac{R2 + R10}{2} \quad (1)$$

The average of four adjacent diagonal pixel values is assigned to the interpolated pixel at a red/blue position.

$$R7 = \frac{R2 + R4 + R10 + R12}{4} \quad (2)$$

Edge-sensing interpolation is an adaptive method and it is used to recover the missing green elements only. This method detects edge before the interpolation. Then, it classifies pixels into several categories using edge orientation and different interpolation schemes are applied to interpolate the missing components for different categories. By the edge patterns, the missing color elements can be reconstructed. The algorithm is given by

$$\begin{aligned} &\text{if } (\Delta H < T) \text{ and } (\Delta V > T) \\ &\quad G = G_H \\ &\text{else if } (\Delta H > T) \text{ and } (\Delta V < T) \\ &\quad G = G_V \\ &\text{else} \\ &\quad G = G_{AVG} \end{aligned} \quad (3)$$

where T denotes the threshold. Referring to Fig. 2, take G7 for example. The horizontal and vertical gradients are calculated as

$$\Delta H = |G6 - G8|, \Delta V = |G3 - G11| \quad (4)$$

The three estimations is calculated as

$$G_H7 = \frac{G6 + G8}{2} \quad (5)$$

$$G_V7 = \frac{G3+G11}{2} \quad (6)$$

$$G_{AVG}7 = \frac{G3+G6+G8+G11}{4} \quad (7)$$

Smooth-hue-transition interpolation method is used to recovery the missing chrominance (R and B) elements only. It performs the bilinear interpolation algorithm in the hue domain. First the green channel is interpolated by bilinear interpolation, and then the red and blue values are calculated trying to deliver a smooth transition in hue from pixel to pixel. First of all, a red hue value and a blue hue value must be defined as:

$$H_R = \frac{R}{G}, \quad H_B = \frac{B}{G} \quad (8)$$

where H_R denotes the red hue value and H_B denotes the blue hue value. As shown in Fig 2, the missing red channel can be reconstructed as

$$R3 = \frac{G}{2} \left(\frac{B2}{G'2} + \frac{B4}{G'4} \right) \quad (9)$$

$$R8 = \frac{G}{2} \left(\frac{B4}{G'4} + \frac{B12}{G'12} \right) \quad (10)$$

$$R7 = \frac{G}{4} \left(\frac{B2}{G'2} + \frac{B4}{G'4} + \frac{B10}{G'10} + \frac{B12}{G'12} \right) \quad (11)$$

where G' denotes the interpolated G value. The blue plane is interpolated in the same manner as the interpolation of red channel.

3 Color Image Models

The color image models presented in this section are use to develop the demosaicking algorithm. These models include the constant color difference model, the constant hue model, and the luminance-color difference model. We will introduce the basic ideas of these models and detail how these image models are utilized for our interpolation algorithm.

3.1 The Constant Color Difference Model

Because of the high correlation between the red, green and blue signals, the interpolation method of green signal can take advantage of the red and blue information. In [2], Pei and Tam developed an image model about the correlation between green and chrominance signals. Authors define K_R as green minus red and K_B as green minus blue, as shown below.

$$K_R = G-R \quad K_B = G-B \quad (12)$$

For natural images, the contrasts of K_R and K_B are quite flat over small region and this property is suitable for interpolation. Figure 3 illustrates an example of green channel image, K_R , and K_B images. Based on this observation, the authors reduce the interpolation error and the image quality is improved. From another viewpoint, this concept is similar to that of the smooth-hue-transition method.

Fig. 3. (a) G channel image, (b) K_R image, (c) K_B image

Fig. 4. (a) G channel image, (b) H_R image, (c) H_B image

3.2 The Constant Hue Model

Although CFA interpolation has been an object of study for a long time, almost algorithms focus on color ratios rule and inter-channel color differences rule. Based on these observations, several schemes have been devised to estimate missing color values with the aid of other color planes. However, these methods normally do not perform satisfactorily around sharp edges and fine details. By taking the advantages of both types of demosaicking methods, an image model using color difference ratios is proposed. After a lot of experiments, we discover that an image domain is suitable for interpolation. The image model is called constant hue model. In our hue model, we define H_R and H_B as

$$H_R = \frac{G-R}{G-B} \quad H_B = \frac{G-B}{G-R} \quad (13)$$

The same observation can be fined in our model. For real world images, the contrasts of H_R and H_B are quite flat over small region and this property is suitable for interpolation. Figure 4 helps account for the results. As the figures indicate, the gray parts are the results of H_R and H_B values. There are a few exceptions to the rule. The white parts include denominator is zero or the absolute values of H_R and H_B are greater than one. These exceptions are not processed in our algorithm.

Fig. 5. (a) Luminance image, (b) L_R image, (c) L_G image, (d) L_B image

Fig. 6. Flowchart of the proposed algorithm

3.3 The Luminance Color Difference Model

Luminance information plays an important role in dictating the quality of a color image. In [11], the luminance color difference model is proposed. They define three luminance color difference planes as

$$\begin{aligned}
 L_R &= L - R \\
 L_G &= L - G \\
 L_B &= L - B
 \end{aligned} \tag{14}$$

These three chrominance planes are generally smooth because the high-frequency contents of the luminance plane are highly correlated with that of each color plane. Figure 5 shows the luminance and the three chrominance planes of a test image. Through exploiting this strong spatial smoothness in these luminance color difference planes, the missing values can be estimated by adaptively combining the neighboring luminance color difference values.

4 The Proposed Demosaicking Algorithm

The proposed algorithm is shown in Fig 6. The whole algorithm can be divided into four parts. The first part is the initial interpolation. The bilinear interpolation is used in step 1. The second part is the update of the green channel. It employs constant color difference model to update the result of the green channel from step 1. The third part is that the red and blue channels are updated by the constant hue model. The fourth part is an iterative update algorithm. It uses luminance-color difference model and edge information. This step aims to suppress visible artifacts residing in the demosaicked images obtained from the aforementioned interpolation step.

4.1 Initial Interpolation

In this step, we will interpolate roughly the red, green, and blue channels to obtain initial estimates. Bilinear interpolation algorithm can be used for this initial interpolation. Because the CFA image has only one channel information per pixel, our algorithm cannot exploit inter-channel correlation in next step. That is why that the algorithm needs initial interpolation.

4.2 Update the Green Channel

Since the green channel plays an important role, the green component is updated first with the help of the red and blue channels. The reason is that the green channel is sampled at higher rate than the red and blue channels in Bayer color filter array. To find the missing green value, the constant color difference model presented before is used. We can update the green channel as follows. Before updating the green channel, we have to calculate the K_R and K_B values around the updated pixel. Figure 7 shows the reference neighboring samples of G channel. Referring to Fig. 7(a), we take the G value at pixel (x,y) for example. We can transform the operation into K_R domain.

$$G(x, y) - R(x, y) = K_R(x, y) \quad (15)$$

Since our algorithm have initialized the CFA image, we can use the average of four surrounding values at locations: $(x+1,y)$, $(x-1,y)$, $(x,y+1)$, and $(x,y-1)$ to estimate K_R at pixel (x,y) directly. The four surrounding K_R values can be calculated as green minus red at the position. As shown in following equations

$$K_R(x+1, y) = G(x+1, y) - R(x+1, y) \quad (16)$$

$$K_R(x-1, y) = G(x-1, y) - R(x-1, y) \quad (17)$$

$$K_R(x, y+1) = G(x, y+1) - R(x, y+1) \quad (18)$$

$$K_R(x, y-1) = G(x, y-1) - R(x, y-1) \quad (19)$$

The G value at pixel (x,y) can be presented as

$$G(x, y) = R(x, y) + \frac{1}{4}(K_R(x+1, y) + K_R(x-1, y) + K_R(x, y+1) + K_R(x, y-1)) \quad (20)$$

The update of green value at a blue pixel is similar to the update at a red pixel.

Fig. 7. Reference neighboring samples (x and y are row and column indexes, respectively)

Fig. 8. (a) Zipper effect, (b) false color

4.3 Update Red and Blue Channels

Since the green channel plays an important role, the green component is updated first with the help of the red and blue channels. The reason is that the green channel is sampled at higher rate than the red and blue channels in Bayer color filter array. In step 2, we update green channel using Pei's method. The red and blue channels are updated follow the green channel. In this step, the proposed constant hue model is used to update the red and blue channels. The basic idea is similar to the constant color difference model. We can transform the operation into the hue domain. Referring to Fig. 7(b), the red values at the locations (x,y) is presented as following equation.

$$R(x,y) = G(x,y) - H_R(x,y)(G(x,y) - B(x,y)) \quad (21)$$

The H_R value at the updated pixel is calculated first. We can use the average of its surrounding H_R values to estimate H_R at the updated pixel directly. The H_R values at three locations: $(x+1,y)$, $(x,y-1)$, and (x,y) are taken for example.

$$H_R(x+1, y) = \frac{1}{2}(H_R(x+1, y-1) + H_R(x+1, y+1)) \quad (22)$$

$$H_R(x, y-1) = \frac{1}{2}(H_R(x, y-1) + H_R(x, y+1)) \quad (23)$$

$$H_R(x, y) = \frac{1}{4}(H_R(x-1, y-1) + H_R(x-1, y+1) + H_R(x+1, y-1) + H_R(x+1, y+1)) \quad (24)$$

The interpolation of blue channel is similar to the interpolation of red channel.

4.4 Improve the Image Quality by Using Adaptively Iterative Update Algorithm

If demosaicking is not performed appropriately, the produced image quality will suffer degradation from highly visible artifacts, such as zipper effect and false color. As shown in Fig. 8(a), Zipper effect means to abrupt or unnatural change in color or intensity between neighboring pixels. False color corresponds to the fact that certain areas contain streaks of colors which do not exist in the original image, as shown in

(Left) Fig. 9. The basic block diagram of the proposed iterative algorithm

(Right) Fig. 10. The Sobel convolution masks used for edge detection

Fig. 8(b). To restore more accurate and visually pleasing results, the adaptive iterative method based on the luminance-color difference model is proposed. We will describe directly that how to use the luminance-color difference model for updating the red, green, and blue channels. Due to the artifacts usually occur in edge regions, the iterative algorithm will update the red, green, and blue channels around edges. The basic block diagram of the proposed iterative algorithm is given in Fig. 9.

The Sobel method is used to detect edges in this step. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image. Here we use the Sobel method in a luminance image. The luminance at pixel (x,y) can be expressed as the equation (25).

$$L(x, y) = \frac{1}{4} (R(x, y) + 2 \times R(x, y) + B(x, y)) \quad (25)$$

The Sobel operator consists of a pair of 3×3 convolution masks as shown in Fig. 10. The kernels can be applied separately to the input image and produce separate measurements of the gradient component in each orientation (G_x and G_y). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The approximate gradient magnitude is given by:

$$Grad = |G_x| + |G_y| \quad (26)$$

If the gradient magnitude is greater than a predetermined threshold, the pixel is considered the edge pixel.

Fig. 11. The absolute difference value of update in different images

Fig. 12. Images used in experiments. (These images are referred as Image1 to Image 16 in this thesis, enumerated from left-to-right, and top-to-bottom.).

After edge detection, we will update the red, green, and blue channels for these edge pixels. The other two channels are updated based on luminance-color difference model except for the original channel in the CFA image. If the green channel is updated, it uses the luminance-color difference correlation and edge direction. If the red and blue channels are updated, it only uses simple luminance-color difference correlation. The green channel is updated according to the edge direction. The algorithm is similar to edge-sensing interpolation. Referring to Fig. 7.(a) and (b), the algorithm is given by

If $(\Delta H < T)$ and $(\Delta V > T)$

$$G(x, y) = L(x, y) - \frac{1}{2}(G(x+1, y) + G(x-1, y))$$

else if $(\Delta H > T)$ and $(\Delta V < T)$

$$G(x, y) = L(x, y) - \frac{1}{2}(G(x, y+1) + G(x, y-1)) \quad (27)$$

else

$$G(x, y) = \frac{1}{4}(G(x, y+1) + G(x, y-1) + G(x+1, y) + G(x-1, y))$$

The horizontal and vertical gradients are calculated as

$$\Delta H = |G(x, y+1) - G(x, y-1)| \quad (28)$$

$$\Delta V = |G(x+1, y) - G(x-1, y)| \quad (29)$$

The red channel is updated based on luminance-color difference model. Referring to Fig. 7(b), the red values at three locations: $(x+1, y)$, $(x, y-1)$, and (x, y) are taken for example. The red values at these three locations can be calculated as following equations.

$$R(x+1, y) = L(x+1, y) - \frac{1}{2}(L_R(x+1, y) + L_R(x+1, y+1)) \quad (30)$$

$$R(x, y-1) = L(x, y-1) - \frac{1}{2}(L_R(x-1, y-1) + L_R(x+1, y-1)) \quad (31)$$

$$R(x, y) = L(x, y) - \frac{1}{2}(L_R(x-1, y-1) + L_R(x+1, y-1) + L_R(x-1, y+1) + L_R(x+1, y+1)) \quad (32)$$

The interpolation of blue channel is similar to the interpolation of red channel. The blue values can be similarly estimated using the L_R values. Based on the above discussion, we conclude that iterative demosaicking requires a stopping criterion in order to minimize the risk of artifacts. Due to the type of images is not the same, it needs

Table 1. Comparison of PSNR obtained by different methods

Img	Bilinear	ECI	Proposed	Img	Bilinear	ECI	Proposed
1	25.1983	31.2458	33.1116	9	30.5707	33.141	33.802
	29.3453	35.2724	36.1268		34.4121	39.2096	39.3059
	24.7342	33.0498	31.4955		29.1981	36.7883	36.176
	25.9942	32.8873	33.1842		30.8942	35.6616	35.877
2	32.9579	35.0892	37.5982	10	31.2803	39.1357	37.6562
	36.5939	41.4552	40.8904		34.4937	40.3678	41.4204
	31.9	38.5411	36.0545		31.1881	38.4157	36.8205
	33.4014	37.6007	37.7523		32.0745	39.2333	38.2237
3	25.7452	32.3681	33.7775	11	26.9332	34.88	34.1626
	29.3343	36.5174	37.4681		30.3703	36.2609	37.4117
	25.6496	34.3285	33.3527		27.2461	34.3573	33.358
	26.6073	34.0828	34.5171		27.9319	35.0945	34.6577
4	27.6818	34.6458	34.9566	12	28.1475	34.8555	35.3595
	32.0801	36.981	38.7807		32.8665	37.4168	38.9585
	27.8705	34.0785	32.637		28.1213	34.5623	33.3298
	28.7968	35.0655	34.7856		29.2202	35.4366	35.3129
5	33.5907	40.2112	39.9079	13	29.4347	38.8646	38.0733
	37.4494	42.129	43.2795		35.3101	40.53	41.442
	33.2664	39.6653	38.2019		31.413	35.2614	35.783
	34.402	40.5467	39.9938		31.4353	37.6435	37.8547
6	23.4831	31.004	31.1662	14	28.7127	35.6157	36.0041
	28.3377	33.9382	35.4533		32.581	37.5707	39.7511
	23.4509	30.9946	31.2809		28.5333	34.9765	35.1737
	24.5725	31.7767	32.233		29.5878	35.9214	36.5713
7	29.2908	35.5608	34.3917	15	35.6084	40.3241	40.0832
	33.2913	38.124	40.1086		39.1438	43.8424	45.5646
	29.2758	36.0815	36.2374		35.2393	41.1522	41.392
	30.2572	36.4559	36.3256		36.3454	41.5327	41.7948
8	22.7615	29.9026	31.4633	16	27.0526	34.368	34.4993
	26.2456	32.2465	33.6793		30.2049	35.542	37.757
	22.8442	30.5098	29.709		26.1341	33.1528	32.9299
	23.6776	30.7778	31.3236		27.479	34.2452	34.6351

different times to update. Our proposed stopping criterion is suitable for all images. We exploit the amount of update for the first time as the stopping criterion. It should be noted that the stopping criterion is based on the absolute difference of color channels. The stopping criterion is defined as

$$s = \frac{1}{\alpha} \sum_{x,y \in \text{edge pixels}} |P_c^1(x,y) - P_c(x,y)|, c = R, G, \text{ or } B \tag{33}$$

where P_c^1 denotes the updated color channel for the first time in step 4, P_c denotes the color channel of the input image from step 3, and α defines the stopping coefficient.

The stopping coefficient can control the time of the process. The iteration is terminated if the following condition is satisfied.

$$\sum_{x,y \in \text{edge pixels}} |P_c^{n+1}(x,y) - P_c^n(x,y)| < s, c = R, G, \text{ or } B \quad (34)$$

where P_c^n denotes the n th updated color channel and P_c^{n+1} denotes the $n+1$ th updated color channel. After experiments and observations, we discover that our iterative update algorithm is convergence. Figure 11 shows the absolute difference value of update in different images. The convergence of the iterative interpolation can be demonstrated by the decreases of the absolute difference value between the interpolated images obtained by two successive updates (refer to Fig. 11).

Fig. 13. The original and demosaicked results of a cropped region from image 8

Fig. 14. The original and demosaicked results of a cropped region from image 14

5 Simulation Results

In our experiments, we used the images shown in Fig. 12 [14]. These images are film captures and digitized with photo scanner. Full color channels are available, and the CFA is simulated by sampling the channels. The sampled channels are used to test the demosaicking algorithms. We use bilinear interpolation for the red, green, and blue channels to get initial estimates. Then the green channel is updated with the constant color difference model, and the red and blue channels are updated with the constant hue model. Finally, the data is update repeatedly until the stopping criterion is satisfied. The stopping coefficient was set to 16 for all images. The edge threshold was set to 60 for all images. The performance in terms of PSNR can be seen in Table 1 for our and other demosaicking algorithms. The results of the red, green, blue, and all planes for each image are shown in top to bottom, respectively. The better PSNR result of each row is shown in bold. We also provide some examples from the images used in the experiments for visual comparison. Figure 13-14 show cropped segments

from original images (Images 8 and 14 in Fig. 12), and the corresponding reconstructed images from the demosaicking algorithms that were used in comparison.

6 Conclusions

In this paper, an adaptive CFA interpolation algorithm with better performance is proposed. The proposed constant hue model based on color difference ratios rule is successively used to estimate the missing color primaries. Since most false colors and zipper artifacts occur in edge regions, an adaptive iterative algorithm using luminance-color difference correlation and the information of edge direction is used to improve the image quality around the edges. The proposed stopping criterion is suitable for all images. Simulation results show that the proposed algorithm can obtain better image quality compared with demosaicking method using color difference rule in 16 photographic color images.

References

1. Bayer, B.: Color Imaging Array. U.S. Patent, no. 3,971,065 (1976)
2. Pei, S.C., Tam, I.K.: Effective color interpolation in CCD color filter array using signal correlation. *IEEE Trans. on Circuits Systems Video Technology* 13, 503–513 (2003)
3. Li, X.: Demosaicing by successive approximation. *IEEE Trans. on Image Processing* 14, 370–379 (2005)
4. Chang, L., Tan, Y.P.: Effective use of spatial and spectral correlations for color filter array demosaicking. *IEEE Trans. on Consumer Electronics* 50, 355–365 (2004)
5. Kimmel, R.: Demosaicing: Image reconstruction from CCD samples. *IEEE Trans. on Image Processing* 8, 1221–1228 (1999)
6. Cok, D.R.: Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal. U.S. Patent, no. 4,642,678 (1987)
7. Gunturk, B.K., Altunbasak, Y., Mersereau, R.M.: Color plane interpolation using alternating projections. *IEEE Trans. on Image Processing* 11, 997–1013 (2002)
8. Li, X., Orchard, M.T.: New edge-directed interpolation. *IEEE Trans. on Image Processing* 10, 1521–1527 (2001)
9. Chang, H.A., Chen, H.: Directionally weighted color interpolation for digital cameras. *ISCAS* 6, 6284–6287 (2005)
10. Alleysson, S., Susstrunk, S., Herault, J.: Linear demosaicking inspired by human visual system. *IEEE Trans. on Image Processing* 14, 439–449 (2005)
11. Lian, N., Chang, L., Tan, Y.P.: Improved color filter array demosaicking by accurate luminance estimation. *ICIP* 1, 41–44 (2005)
12. Wu, X., Zhang, N.: Primary-consistent soft-decision color demosaicking for digital cameras. *IEEE Trans. on Image Processing* 13, 1263–1274 (2004)
13. Lu, W., Tan, Y.P.: Color Filter Array Demosaicking: New Method and Performance Measures. *IEEE Trans. on Image Processing* 12, 1194–1210 (2003)
14. Kodak Lossless True Color Image Suite (2007), <http://www.r0k.us/graphics/kodak/>