Enhancement of Color Images By Efficient Demosaicing

Liron D. Grossmann* and Yonina C. Eldar†

September 23, 2005

Abstract

We propose a simple yet powerful method for reconstructing a full-color image from its partially sampled version. The suggested algorithm is non-iterative and is based on the properties of the human visual system. While several state-of-the-art algorithms invest a great deal of computational effort in the enhancement of the reconstructed image to overcome color artifacts, we focus on eliminating the majority of the them in the initial stage of the algorithm. We tested our algorithm on several problematic images and found it to often be significantly superior to state-of-the-art algorithms, without consuming high computational power.

Index Terms—Demosaicing, interpolation, color spaces, computational complexity.

1 Introduction

It is well known that in order to display a digital color image, one needs to specify only three values for each pixel, i.e. the red, green, and blue component (often called the color channels) at the corresponding location. Due to technology limitations, however, most charge-coupled-device (CCD) cameras provide a single value for each pixel, that is, either red, green, or blue. The resulting image is often called a "mosaic". The problem of demosaicing is to reconstruct the original, full-color, image from its subsampled version.

*Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa 32000, Israel. E-mail: lirongr@tx.technion.ac.il.

†Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa 32000, Israel. E-mail: yonina@ee.technion.ac.il.

Many demosaicing algorithms have been proposed over the last decade; see, e.g. [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] to mention a few. In [14] the existing methods were broadly divided into three categories. The first group includes the heuristic approaches and contains the largest number of existing algorithms. The second class is based on formulating the demosaicing problem as an optimization problem, while the third category relies on a model of the image formation process and solves an inverse problem. Our algorithm belongs to the first class, and combines several common ideas into one method, yielding better visual results, while maintaining a low computational demand.

Roughly speaking, the algorithms in the first category can be further classified into three groups. In the first group standard interpolation techniques, such as nearest neighbor, bilinear, or cubic interpolation are used to interpolate the missing pixels. Such methods are computationally fast and simple to implement, but result in severe color artifacts. The algorithms belonging to this category completely ignore the color attributes of the image, explaining their degraded visual quality.

The second class of methods takes into account the cross-correlation among the three color channels [2, 5, 7, 8]. Common use of this correlation is to assume that the hue changes smoothly between neighboring pixels, and therefore interpolation is performed on the color ratios instead of on the channels themselves. However, the assumption of constant hue may not hold around certain kinds of edges. As a result, some algorithms also incorporate an edge-detection mechanism to avoid interpolating over edges. These techniques lead to higher quality images than the algorithms in the first category, demonstrating that combining knowledge of the human visual system can improve the quality of the reconstructed image. Nevertheless, the majority of these methods still yield poor results in certain problematic regions.

The third category of algorithms includes those whose visual results are considered the best [1, 4, 9, 10, 11, 12, 13]. Most of them (except [4]) start with an initial estimation of the original image (such as bilinear interpolation), followed by a correction stage, which is intended to eliminate most of the color artifacts caused by the initial interpolation scheme. The correction methods exploit additional properties of color images, which may require a transformation to another color space. Despite the good quality obtained by this class of algorithms, there are still several artifacts that cannot be completely eliminated by them. Usually it happens because of a poor initial interpolation method creates artifacts, which are hard to track and compensate.

What is lacking, therefore, is a demosaicing method, which is not too complex (both conceptually and computationally), but at the same time can reduce most of the common problems arising from the subsampling of the color channels. It is clear that a good demosaicing algorithm should be

driven by the interaction between color image attributes and the human visual system. In addition, existing algorithms demonstrate that a correction method is a useful component in the demosaicing chain. Nevertheless, the input to this stage should not contain too many visual artifacts, so computational effort should be split between the initial estimation and the correction method.

In this paper we present an algorithm which takes into account the properties of color images, and whose running time is fast when compared to the existing methods. Our method follows the same pattern as that of the third category: it starts with an initial estimation of the color image, and then applies a correction method. In the first stage a classification of the missing pixels is made, according to which different interpolation schemes are applied. This strategy allows us to isolate specific visual artifacts and treat them separately, without affecting the estimation of the rest of the pixels. In the second stage a correction method, which relies on the smooth behavior of the hue component of the image, is carried out on problematic pixels. Simulation results verify that the proposed method succeeds in eliminating most of the typical artifacts.

The paper is organized as follows. In Section 2, we introduce the problem of demosaicing in a way suitable for the development of the algorithm. The main steps of the algorithm are outlined in Section 3. Computational complexity analysis is made in Section 4. Finally, comparisons with existing demosaicing methods are discussed in Section 5.[1]

2 Problem Formulation and Main Results

The problem of demosaicing is that of reconstructing a multi-color image from its subsampled version, given in the form of a single array of pixels. The most popular pattern is the Bayer Color Filter Array (CFA), which is shown in Fig. 1, and will be the one considered here [15]. In this pattern, 50% of the pixels are green, 25% are red, and 25% are blue.

As noted in the introduction the demosaicing problem can be regarded as an interpolation problem. The solution method should yield an outcome, that is as close as possible to the original image. By close, we mean subjectively close, as there is no universal metric for images, which is in accordance with the human visual system [16]. Therefore, throughout the paper, when we refer to a "good" quality image, we mean subjectively good as seen by the viewer of the image.

A major obstacle in every demosaicing algorithm is handling regions with rapid changes, where aliasing occurred in the green channel (and therefore

[1]The images in this paper are best viewed on a monitor. An on-line version is available at http://www.ee.technion.ac.il/Sites/People/YoninaEldar/ under Journal Publications.

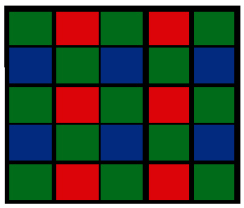
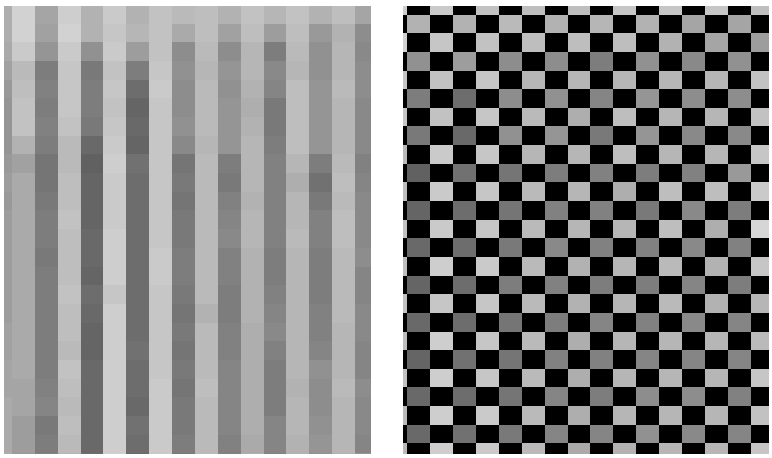


Figure 1: The Bayer array pattern

in the red and blue as well). A typical example is demonstrated in Fig. 2, in which the green channel of black and white vertical stripes is shown together with its corresponding subsampled version.



(a) (b)

Figure 2: A region in the green channel that is aliased after subsampling (a) original (b) the corresponding region in the mosaic image.

The main attribute of our algorithm is that it succeeds in interpolating such regions without causing color artifacts. It does so by "forcing" a good initial estimation of the image, and then correcting these aliased regions in subsequent stages.

A block diagram of our algorithm is shown in Fig. 3. The first two blocks correspond to an adaptive initial interpolation of the color image from the partially sampled one. The last two blocks constitute the correction part, which eliminates visual artifacts that remain in the image.

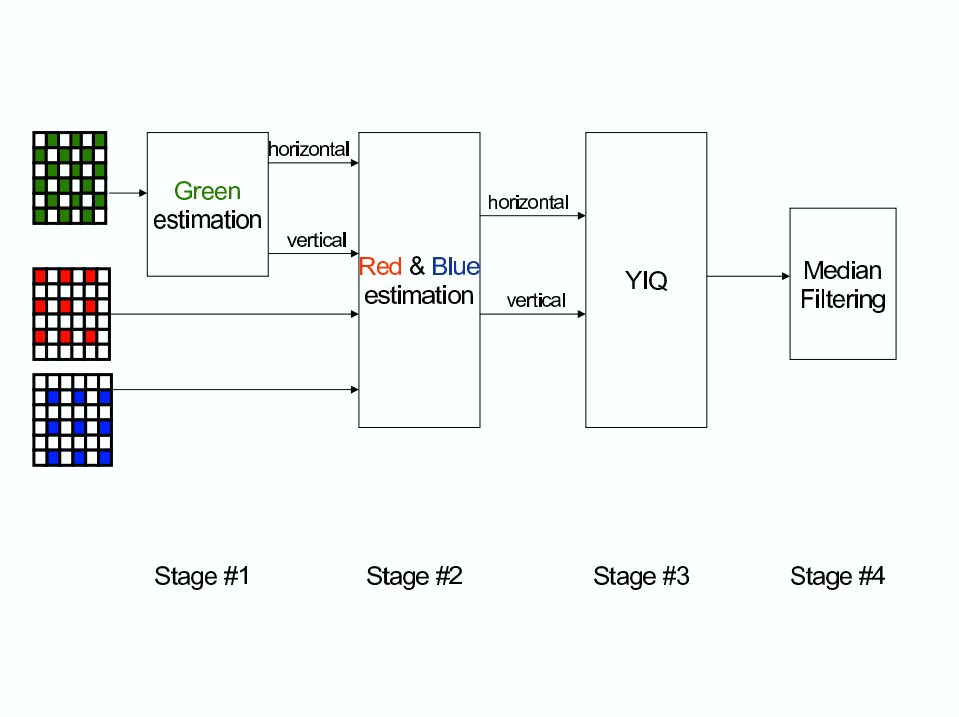


Figure 3: A block diagram of the algorithm.

The initial estimation procedure starts with the green channel by classifying each missing pixels into one of three groups. The classification rule is based on the the nature of the pixel's neighborhood, and leads to different interpolation methods. The output of this stage are two green channel images, where in the one certain pixels, which we call "special" pixels, are interpolated in the horizontal direction, while at the second they are interpolated vertically. This splitting aims at treating the situation of aliasing that was shown in Fig. 2. The determination of the right direction is left to subsequent stages.

Following the green estimation, the red and blue are interpolated according to the inter-channel correlation. Since we have two green estimates, horizontal and vertical, we interpolate the red and blue channels twice, once for each direction.

The resulting two color images are passed through the correction stage, in which color artifacts are reduced. The goal of this step is to determine the right direction of the "special" pixels from the first stage. The assumption we make is that pixels which possess a smoother chrominance value, indicate

the correct interpolation direction. One way to test the chrominance of a pixel is to transform its RGB values into the YIQ space, where the I and Q components stand for the chromaticity of the pixel [3, 17]. The resulting I and Q components are then compared with their neighbors to evaluate the relative chromatic smoothness of the pixels. The output of this stage is a color image where the "special" pixels are interpolated correctly. Finally, the image is median filtered in order to reduce possible pointwise artifacts.

A more detailed description of the above method is given in the next section.

3 A Description of the Algorithm

The following steps describe the proposed algorithm together with figures corresponding to the outputs of each stage. We chose the Lighthouse image, which is considered to be a very difficult image to reconstruct from its mosaic, in order to emphasize the power of our technique.

I. Green Estimation. This step consists of two part: classification, and interpolation.

Each missing green pixel is classified to one of three classes according to the degree of smoothness of its neighborhood, where smoothness means that the neighboring pixels are "close" to each other in a sense to be explained below. The three classes are shown in Fig. 4. Pixels with smooth

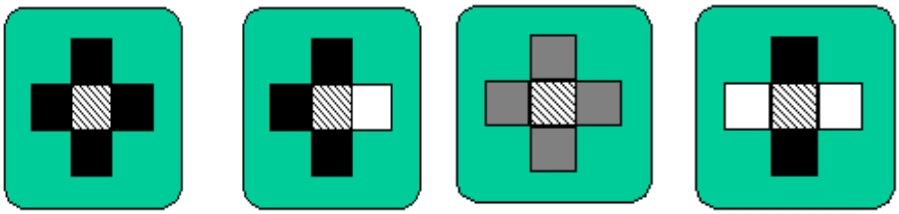


Figure 4: The three classes used in the classification process.

neighborhood are said to be class A pixels. Class B contains pixels that do not belong to A, but whose value can be inferred with the aid of pixels from other channels, i.e. the red and blue, in an enlarged neighborhood. In Class C we find all the rest of the pixels, that we call "special". These are pixels, whose value cannot be determined using their neighbors in the green channel nor in the other channels (like the stripes in Fig. 2).

Denoting the surrounding pixels of the missing green by $\{G_1, G_2, G_3, G_4\}$ (see Fig. 5), a smooth neighborhood is one, where at least three of them

are "close". Pixels in class C will be those where G_1 is close G_3, while G_2 is close to G_4 but different from G_1. The rest of the pixels will belong to class B.

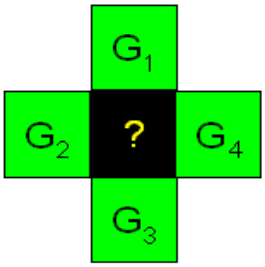


Figure 5: The neighborhood of a missing pixel.

We now explain what we mean by "close" pixels. Two pixels, denoted by G_1 and G_2 with $G_1 < G_2$, are said to be close if their relative difference,

$$\frac{G_2 - G_1}{G_1}, \tag{1}$$

is below a predetermined threshold, which is device dependent. The thresholds we use also depend on the intensity level of G_1, and were obtained via visual experiments. Therefore, they may change according to the user's own camera. In order to avoid division by zero, if G_1 is smaller than some threshold, then the difference, $G_2 - G_1$ is used instead of the ratio in (1). The thresholds we use are summarized in Table 1 for G_1 and G_2 assuming values between 0 and 255. Note that the first value corresponds to the difference and not to the ratio.

The discrimination rule in (1) is reminiscent of the Weber's law, which states that the ratio of the increment in the light intensity to the background is constant [17]. However, it is known that in practice this ratio is not perfectly constant, justifying the thresholds being dependent on G_1. Furthermore, when (1) is used on a raw image, i.e. a nonrendered one, the green channel is highly correlated with the light intensity, and this further motivates the use of Weber law to discriminate green pixels. The thresholds used may change as well using rendered and nonrendered images.

We shall now describe how to interpolate each class. Since pixels of class A have a "smooth" environment, each missing green is replaced by the average of its neighbors, in order to preserve the smoothness. If the neighborhood is completely smooth, that is, all of its pixels are "close",

Intensity level	Threshold
$0 \leq G_1 \leq 30$	20
$30 < G_1 \leq 75$	0.7
$75 < G_1 \leq 120$	0.27
$120 < G_1 \leq 141$	0.16
$141 < G_1 \leq 161$	0.11
$161 < G_1 \leq 201$	0.08
$201 < G_1 \leq 235$	0.06
$235 < G_1 \leq 255$	0.08

Table 1: Thresholds for using the discrimination rule.

then the estimated green, $\hat{G}$, is given by

$$\hat{G} = \frac{G_1 + G_2 + G_3 + G_4}{4}. \tag{2}$$

If the neighborhood is partially smooth (such as the right one in class A of Fig. 4), then we use the interpolation

$$\hat{G} = \frac{G_1 + G_2 + G_3}{3}. \tag{3}$$

In class B the estimation is carried out using a larger neighborhood that contains pixels from the red and blue channels as well. Fig. 6 shows the pixels, which are included in the interpolation of this class. Note that we could also use this scheme for class A, but it is not necessary there, since the intra-channel correlation suffices. In fact, using the inter-correlation in class A may even decrease the visual performance. We adopt the approach of [5] to exploit the inter-channel correlation. The value of the estimate, $\hat{G}$, is computed by adding R (which is given) to the estimated difference $G - R$, leading to the interpolation formula

$$\hat{G} = R + \frac{(G_1 - \frac{(R_1+R)}{2}) + (G_2 - \frac{(R_2+R)}{2}) + (G_4 - \frac{(R_4+R)}{2}) + (G_3 - \frac{(R_3+R)}{2})}{4}. \tag{4}$$

Each summand in the numerator is an estimate of the difference $G - R$, and the final estimate is their average. A similar procedure is applied when the neighborhood contains pixels of the blue channel.

The "special" pixels (class C) are treated differently. Their neighborhood is a result of the aliasing created in the subsampling of the green channel

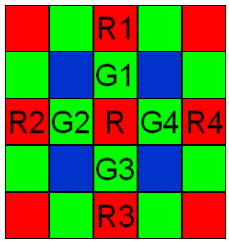


Figure 6: The mosaic image, where the marked pixels are used in the interpolation of class B.

itself. In Fig. 2 the effect of aliasing in the green channel was shown, where it is clear that the right direction of the stripes can no longer be inferred from the subsapmled green. The aliasing phenomenon is even more disastrous in the red and blue channels, since their sampling rate is lower than that of the green. We are therefore led to a situation, where the inter-channel correlation can no longer be exploited in the interpolation process. Enlarging the size of the neighborhood in the green channel is not helpful either. As the correct interpolation is not known, we assume it may take on two possible directions, horizontal or vertical. Therefore, each "special" pixel is interpolated twice: horizontally by,

$$\hat{G} = \frac{G_2 + G_4}{2}, \tag{5}$$

and vertically by,

$$\hat{G} = \frac{G_1 + G_3}{2}. \tag{6}$$

This choice of reconstruction stems from two reasons: the first is driven by the simplicity of implementation, while the other is a psychophysical one. Using only two possible directions for interpolation reduces the complexity of the comparison and selection process in the correction stage. In addition, it is a known fact that the human eye is most sensitive to edges in the horizontal and vertical directions, and less sensitive to diagonal edges [18].

As a consequence, we end up with two versions of the reconstructed green. One, called the horizontal image, has all its aliased regions estimated in the horizontal direction. The second image, the vertical one, has them all vertically interpolated. The pixels belonging to the other two classes remain the same in both images. Figure 7 shows the outputs of this stage.

(a) (b)

Figure 7: The outputs of the green estimation stage (a) horizontal green channel (b) vertical green channel.

II. Red and Blue Estimation. In this step we "fill in" the missing red and blue pixels. Using the two green versions of the previous step, the red and blue channels are estimated by adding the green value at the missing red or blue location to an estimated difference. The red and blue pixels' neighborhoods differ from the green ones and are shown in Fig. 8 for the red channel. Concentrating on the top red neighborhood in this figure, we

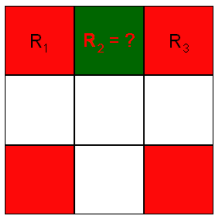


Figure 8: The missing red environment.

interpolate R_2 using the following rule,

$$\hat{R_2} = \begin{cases} G_2 + (R_1 - \hat{G_1}) & \text{if } R_1 < R_3 \\ G_2 + (R_3 - \hat{G_3}) & \text{otherwise,} \end{cases} \tag{7}$$

where $\hat{G}_1, \hat{G}_3$ are the estimated values of the green channel. This rule takes into account the possible difference in the red and green pattern, and prevents misalignment of edges due to phase shift between the given samples. Suppose that there is an increase in the green level, i.e $G_2 > \hat{G}_1$, while at the same time, the intensity of the red pixel (which is missing) R_2 decreases compared to R_1. Using the difference $R_1 - \hat{G}_1$ instead of $R_3 - \hat{G}_3$ would cause an increase in the value of R_2, instead of a desired decrease. A similar situation occurs when the missing red has two vertical neighbors. The case where the estimate has four diagonal neighbors is treated by taking into account only one diagonal. The missing red is interpolated along it as in (7).

The interpolation of the missing red and blue pixels is performed twice - once based on the horizontal green image, and once based on the vertical green image. The resulting two color images at the output of this stage are shown in Fig. 9.

(a) (b)

Figure 9: The output of the red and blue estimation stage (a) horizontal interpolation (b) vertical interpolation.

III. YIQ transformation. It remains to decide on the right direction of the "special" pixels. This is done in three steps. First, transforming the two images at the output of stage **II** to the YIQ space. Second, grouping neighboring "special" pixels into clusters. Third, selecting the cluster version (horizontal or vertical), which results in a smoother I component. We now elaborate on each of theses steps.

In order to detect the smoothness of the chrominance we first transform

the image to the YIQ space, using the formula,

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.2755 & 0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{8}$$

In this space, as opposed to the RGB one, the achromatic behavior of the image (the Y component) is separated from its chromatic behavior (the I and Q components). Since the human eye is more sensitive to abrupt changes in the luminance than to the changes in the chrominance [19], it is reasonable to expect that the right direction of the "special" pixel should yield smoother I and Q components. In the proposed method we shall test the smoothness only in the I component due to two reasons. First, it reduces the number of comparisons to only one component, and therefore accelerate the running time of the algorithm. Second, it is known that the human eye is more sensitive to the changes in the I component then in the Q component. This fact is also exploited in analog TV transmission, where the I component is allocated more bandwidth than the Q component [17].

Considering the I component of each version, i.e. horizontal and vertical, "special" pixels which are in a 3×3 environment are clustered together. It is the spatial correlation assumption that stands behind the idea of clustering.

Our next step is to compute the relative smoothness of each cluster version and to select the smoother one. The relative smoothness of each version is taken to be the ratio between the mean value of the pixels inside the cluster and the mean value of the pixels, which are outside the cluster, but are close to its boundary. Specifically, denote by C_h and by C_v the horizontal and vertical version of a cluster, respectively, and denote the average value of their I components by AvC_h and AvC_v. We denote the averages of the horizontal and vertical environments outside the clusters by AvE_h and AvE_v, respectively. The selected cluster, denoted by C_s, is determined by the following rule:

$$C_s = \begin{cases} C_h & \text{if } \frac{AvC_h}{AvE_h} < \frac{AvC_v}{AvE_v} \\ C_v & \text{otherwise.} \end{cases} \tag{9}$$

The selection of a cluster involves inserting its pixels' values (that is the corresponding Y, I and Q values) into the final image.

After the selection of each cluster is performed, we obtain one color image, which is a mixture of clusters, either horizontal or vertical, together with pixels belonging to class A and B (whose values were already determined in the previous stage). In Fig. 10, the Lighthouse image (in its RGB representation) at the output of this stage is presented.

IV. Median Filtering. The resulting image from the last step may still contain splotches, that are very annoying to the human eye. To elim-

Figure 10: The output of the YIQ stage.

inate these false color points, we perform median filtering on the I and Q components of the image, using a 3×3 neighborhood median filter. The final Lighthouse image is shown in Fig. 11.

4 Computational Complexity

One of the attractive features of the proposed algorithm is that it is conceptually easy to understand and implement. In addition, simulations show that the running time of our algorithm is comparable with several algorithms (i.e. [2, 7, 11, 12]), i.e. it takes about ten to twenty seconds to run on a pentium four PC for medium size images. This is, for example, in contrast with the algorithm of [1], which has a relatively slow running time. Our code was not optimized, so complexity can be further reduced.

Table 2 summarizes the computational complexity of our algorithm. It shows the number of calculation performed for each pixel in the worst case. In practice, the numbers will usually be smaller.

In the interpolation stage, the most computationally expensive pixels are class B ones. The classification to this group requires 8 comparisons (6 for class A and 2 for class C), where each comparison gives rise to one substraction and one division. The interpolation results in 12 additions (or subtractions) along with 5 fixed multiplications (multiplications by a constant).

The conversion from RGB to YIQ requires 2 additions and 3 fixed multiplications. Since pixels from class C are interpolated twice, the number of

Figure 11: The final output of the algorithm.

Stage	adds	fixed multiplies	compares	divides
Interpolation	20	5	8	8
YIQ transformation	6	10	0	0
Clustering	38	2	17	2
Median filtering	0	0	16	0

Table 2: Computations performed per pixel.

operations is doubled. Moreover, color transformation is performed twice, from RGB to YIQ and back to RGB.

In the clustering stage, each "special" pixel is associated with its 3×3 environment, leading to 8 comparisons for checking possible pixels to be clustered. The calculation of the average of its neighbors requires 9 additions and 1 fixed multiplication. The same number of operations to compute the average of its border (if the pixel's neighborhood is on the border of the cluster). Computation of the ratio between the average of the cluster and its border requires one division. This is done for both horizontal and vertical versions of the image, resulting in a total of 36 additions, 4 fixed multiplications, and 2 divisions. The two versions are then compared yielding and additional comparison to the clustering stage. Finally, the I and Q plane of the unified image is median filtered leading to a number of 16 comparisons.

It should be mentioned that if one is willing to trade off computational

complexity for visual quality, then the algorithm can be made even more efficient. For example, the number of classes in the green interpolation can be reduced from three to two, by dropping class A, turning the classification into a binary decision process. The reason for not using two classes in the original algorithm is that in smooth regions, the average of the green pixels results in better visual performance than using pixels from the other channels. The clustering step can also be made less computational demanding, due to possibly large number of clusters may contain only one pixel. Instead of using a 3×3 environment and compare it with its border, we can avoid the comparisons in single pixel clusters, and interpolate them later. The interpolation of single pixels clusters is performed using the Y component of the partial image. In the Y component, the missing pixel is replaced by the average of its 3×3 neighborhood. The I and Q components may be taken to be the average of the horizontal and vertical versions. This, however, may create splotches in the resulting image, which may be partially reduced by the median filtering. In addition, the median operation can exploit a smaller neighborhood leading to less comparisons.

It is interesting to compare the above table with the results in [13], one of the state-of-the-art algorithms. Our algorithm has less stages and requires a smaller number of calculations per pixel. In addition, comparing our method to [20], our computational effort is substantially lower (in [20] method about 348 additions and multiplications are needed per pixel), while our visual results are better. Finally, the algorithm of [21] requires an order of 5000 operations per pixel, proving again that our method is much less computationally demanding.

5 Examples

We now compare our algorithm with several state-of-the-art algorithms. In our comparison, we consider the following reconstruction methods: bilinear [2] interpolation, a constant hue based algorithm [7], a color correction algorithm [11], a color gradient based algorithms [12], Kimmel's method [1] and ours.

5.1 Lighthouse

The Lighthouse image is a very difficult image to demosaic, due to the aliasing that occurs during the subampling of the green channel. The house and fence are excellent examples where this aliasing occurs. Figure 12 shows the original Lighthouse. Figures 13 -18 show the result of the above mentioned algorithms, and Fig. 19 shows our algorithm.

The bilinear interpolation method creates many color artifacts, especially in the aliased regions, i.e. the fence and the house. Better results are achieved by all of the human visual system based algorithms. Assuming

Figure 12: The original Lighthouse.

smooth hue transitions improves the visual quality of the image in certain regions, but still results in sever color artifacts in the fence and house regions, where the aliasing of the three channels occurs. In all of the above images, we can see that the fence and house suffer from false color artifacts. In Figs. 16 and 18 the artifacts are less severe. In our outcome, Fig. 19, both the fence and the house have no color artifacts, and our result is the closest to the original.

5.2 Sails

The Sails image is also a benchmark. A problematic region is the number "14255". Figure 20 shows our outcome, which is very close to the original Sails. Kimmel's method and the variable gradients method are shown in Fig. 21. It can be seen that false colors are created by these methods, in particular across the edges of the digits.

Figure 13: Bilinear interpolation.

5.3 Window

The Window image exhibits a rapid transition region in the stem of the flower. An enlarged part of this image containing that region is shown in Fig. 22. The variable gradients method are shown in Fig. 23. It can be seen that splotches of various colors are created by these methods across the stem. In addition false colors appear in the shutters of the window in the variable gradients method.

5.4 Fruits

The Fruits is a raw image. This image possesses a lot of details, and color changes. In Fig. 26 an enlarged part of the bowl is shown. Splotches of various colors are created around the golden strip of the bowl and on the glasses. Our method eliminates them.

Figure 14: Smooth hue transition.

6 Conclusions

We have proposed a simple algorithm for the demosaicing algorithm. The solution is based on the interaction between color attributes and the human visual system. We have shown that our algorithm outperforms many of the existing methods. Our design consists of two basic steps, an initial interpolation followed by an enhancement stage. Our approach differs from the existing methods in that it focuses on accomplishing a better initial reconstruction of the image, rather than a better enhancement method. This strategy has proven to be successful in both preventing most of the color artifacts, and in controlling the inevitable ones. In addition, we introduced a simple method for detecting and correcting color artifacts, relying on the orientational sensitivity of the human eye and on the smooth color transition assumption. The method may be extended to other color image applications, such as image enhancement and denoising.

A computational analysis shows that our algorithm requires a relatively

Figure 15: Edge detection algorithm.

small number of operations and therefore can be implemented directly as a part of the camera chip.

7 Acknowledgment

The authors wish to thank Gil Rosenfeld and Noam Lipcer for implementing the code, and for their contribution to the algorithm. We also wish to thank Nimrod Peleg and the SIPL lab at the Technion for supporting this research, and for Creo for providing us with the Fruits image. Finally, we wish to thank Dr. Yacov Hel-Or for fruitful discussions.

References

[1] R. Kimmel, "Demosaicing: Image reconstruction from color CCD samples," *IEEE Trans. on Image Processing*, vol. 8(9), pp. 1221–1228, Sept.

Figure 16: Color correction algorithm.

1999.

[2] J. E. Adams, "Interactions between color plane interpolation and other image processing functions in electronic photography," *Proceedings of SPIE*, vol. 2416, pp. 144–151, Feb. 1995.

[3] Y. Hel-Or and D. Keren, "Demosaicing of color images using steerable wavelets," Tech. Rep. HPL-2002-206R1 20020830, HP.

[4] S. Susstrunk D. Alleysson and J. Herault, "Linear demosaicing inspired by the human visual systems," *IEEE Trans. Image Processing*, vol. 14, pp. 439–449, 2005.

[5] S.C. Pei and I.K. Tam, "Effective color interpolation in CCD color filter array using signal correlation," *Proc. ICIP*, pp. 488–491, Sept. 2000.

[6] X. Wu et.al., "Color restoration from digital camera data by pattern matching," *Proceedings of SPIE*, vol. 3018, pp. 12–17, 1997.

Figure 17: Gradient based algorithm.

[7] D. R. Cok, "Reconstruction of CCD images using template matching," *Proc. of IS & Ts Anual Conference/ICPS*, pp. 380–385, 1994.

[8] 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, 1994.

[9] D. Keren and M. Osadchy, "Restoring subsampled color images," vol. 11, pp. 197–202, 1999.

[10] R. W. Schafer J. W. Glotzbach and K. Illgner, "A method for color filter array interpolation with alias cancellation of images," *Proc. IEEE Int. Conf. Image Processing*, vol. 1, pp. 141–144, 2001.

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

Figure 18: Kimmel's algorithm.

[12] Ed. Chang et.al., "Color filter array recovery using a threshold-based variable number of gradients," *Proceedings of SPIE*, vol. 3650, pp. 36–43, Jan. 1999.

[13] K. Hirakawa and T. W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," *IEEE Trans. on Image Processing*, vol. 14, pp. 360–369, 2005.

[14] Y. Altunbasak R. W. Schafer B. K. Gunturk, J. Glotzbach and R. M. Merseau, "Demosaicking: color filter array interpolation," *IEEE Signal Processing Magazine*, vol. 22, pp. 44–54, 2005.

[15] B. E. Bayer, "Color imaging array," U.S. Patent, 3,971,065, 1975.

[16] J. L. Manos and D. J. Sakrison, "The effects of a visual fidelity criterion on the encoding of images," *IEEE Trans. on Information Theory*, pp. 525–536, 1974.

Figure 19: Our Lighthouse.

[17] A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression*, New York, NY: Plenum Press, 1988.

[18] M. D. Levine, *Vision in Man and Machine*, New York, NY: McGraw-Hill, 1985.

[19] M. D. Fairchild, *Color Appearance Models*, Reading, Mass:Addison-Wesley, 1988.

[20] Y. Altunbasak, B. Gunturk and R. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transactions on Image Processing*, vol. 11, pp. 997–1013, 2002.

[21] D. D. Muresan and T. W. Parks, "Demosaicing using optimal recovery," *IEEE Trans. on Image Processing*, vol. 14, pp. 267–278, 2005.

(a) (b)

Figure 20: (a) The original Sails (b) Our Sails.

(a) (b)

Figure 21: (a) Gradient based Sails (b) Kimmel's Sails.

(a) (b)

Figure 22: (a) The original Window (b) Our Window.

(a) (b)

Figure 23: (a) Gradient based Window (b) Kimmel's Window.

Figure 24: The mosaic Fruits.

Figure 25: Our Fruits.

(a)

(b)

(c)

(d)

Figure 26: (a) Gradient Based Fruits (b) Color correction Fruits (c) Edge detection Fruits (d) Our Fruits.