

# OPTIMAL RECOVERY DEMOSAICING

*D. Darian Muresan and Thomas W. Parks*

School of Electrical and Computer Engineering, Cornell University  
Ithaca, NY 14853  
darian, parks@ece.cornell.edu

## ABSTRACT

Digital color images from single chip digital cameras are obtained by interpolating a color filter array. Color information is encoded by means of a CFA, which contains different color filters (i.e. red green and blue), placed in some pattern. The resulting sparsely sampled images of the three-color planes are interpolated to obtain dense images of the three-color planes and thus the complete color image. Interpolation usually introduces color artifacts due to the phase shifted, aliased signals introduced by the sparse sampling of the CFAs. In this paper we discuss a non-linear interpolation scheme based on edge information, that produces better visual results than those obtained by linear and other published interpolation methods.

## 1. INTRODUCTION

Due to hardware limitations, single CCD arrays in digital cameras do not capture the full red, green and blue color planes. Instead, they capture a sparsely sampled image of each of the color planes and interpolation is then used to reconstruct the original colors. In this paper we analyze the reconstruction of a single digital color image from the information provided by the Bayer CFA of Fig. 1

In previous work [2] we considered the problem of gray scale image interpolation from an adaptive optimal recovery point of view. We showed how a training set  $S$  determines a quadratic signal class and how to use this signal class to perform image interpolation. In this paper we extend our gray scale interpolation ideas to CFA interpolation and through examples show that the new interpolation scheme outperforms linear and other published interpolation methods.

This paper is organized as follows. In section 2 we review the theory of optimal recovery as it applies to gray scale image interpolation [2]. In section 3 we present a unified view of some of the non-linear demosaicing methods existent in the current literature [4, 7, 5]. Finally, in section 4 we present our interpolation algorithm, which is a hybrid of different interpolation ideas. We conclude with

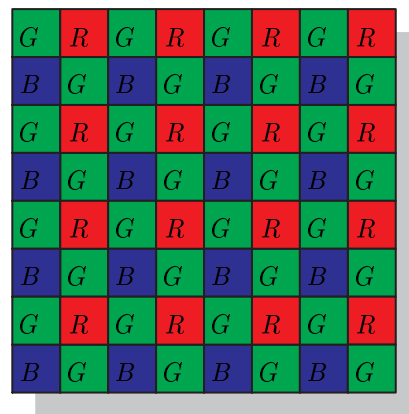


Fig. 1. Bayer Array.

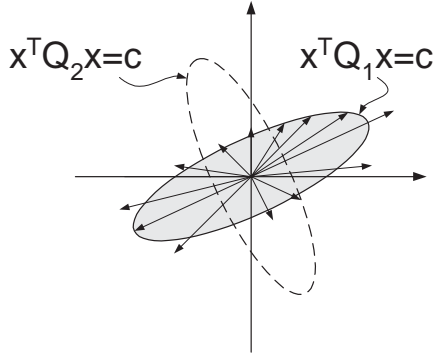
section 5 where we present our results and provide some final thoughts on future work.

## 2. OPTIMAL RECOVERY

We now review the interpolation method presented in [2], which is primarily an interpolation method for gray scale images. Locally, at pixel location  $y$ , we model the image as belonging to a certain ellipsoidal signal class  $K$

$$K = \{x \in R^n : x^T \mathbf{Q}x \leq \epsilon\} \quad (1)$$

where  $\mathbf{Q}$  is derived from a training set or may be assumed known. Vector  $x$  is any subset of the image containing the missing pixel  $y$ . Vector  $x$  is chosen such that **any**  $L$  linear functionals ( $F_i, i = 1, \dots, L$ ) of  $x$  are assumed known. If we note the actual values of the functionals by  $f_i$  we have  $F_i(x) = f_i$ . In this paper we assume that the functionals are based on derivatives and/or actual pixel values of the decimated image. From [9] the optimal vector  $\bar{u}$  belonging to class  $K$  and satisfying the given functionals  $F_i$  is a linear combination of the representors  $\phi_i$  of the given functionals



**Fig. 2.** Training set  $S$  and contours corresponding to  $x^T \mathbf{Q}_1 x = c$  and  $x^T \mathbf{Q}_2 x = c$ . The training set  $S$  is better represented by the contour corresponding to  $x^T \mathbf{Q}_1 x = c$  than the one corresponding to  $x^T \mathbf{Q}_2 x = c$ .

$$F_i \quad \bar{u} = \sum_{i=1}^L \alpha_i \phi_i \quad (2)$$

The challenge then, is to design a  $\mathbf{Q}$  for which the class  $K$  is a proper representation of the local signal. Assume that a given training set  $S$  is to be used for learning our matrix  $\mathbf{Q}$ . For convenience, we further assume that  $S$  is a matrix with the training vectors as columns. These training signals lie in a particular region of the space which we assume to be representative of the signal we are trying to interpolate. Our aim is to find a  $\mathbf{Q}$  such that the ellipsoid

$$x^T \mathbf{Q} x = \text{constant}$$

is representative of the training set  $S$  as shown in Fig. 2. What does this mean? We want a  $\mathbf{Q}$  for which the contour of  $x^T \mathbf{Q} x = c$  models the locus of the training set  $S$ . Intuitively, the contour  $x^T \mathbf{Q} x = c$  is an ellipsoid stretched in the direction of the eigenvectors of  $\mathbf{Q}$ . The stretch is largest in the direction where  $\mathbf{Q}$  has the smallest eigenvalues and smallest in the direction where  $\mathbf{Q}$  has the largest eigenvalues. In [2] we showed that  $\mathbf{Q} = (SS^T)^{-1}$ . By properly choosing the training set  $S$  this method performs well for gray scale interpolation, as it was shown in [2].

### 3. DEMOSAICING REVIEW

The Bayer array (Fig. 1) contains more green (or luminance) pixels than red or blue in order to provide high spatial frequency in luminance at the expense of chrominance signals. Our task is to interpolate each of the R, G and B planes.

The most basic idea is to independently interpolate the R, G and B planes. In other words, to find the missing green

values use only neighboring green values, to find the missing blue values use only neighboring blue pixels and so on for red. More specifically, for a linear interpolation, to obtain the missing green pixels, calculate the average of the four known neighboring green pixels. To calculate the missing blue pixels, proceed in two steps. First, calculate the missing blue pixels at the red location by averaging the four neighboring blue pixels. Second, calculate the missing blue pixels at the green locations by averaging the four neighboring blue pixels. The second step is equivalent to taking 3/8 of each of the closest pixels and 1/16 of four next closest pixels. This type of interpolation, which we call linear interpolation, introduces serious aliasing artifacts.

The demosaicing approach of [3] improves the interpolation of the red and blue colors by adding high-frequency information from the green image to the red and blue images. It also uses the green high-frequency information to estimate the aliasing in the red and blue images, thus providing a means of reducing the amount of aliasing in the reconstructed image.

A different approach to the interpolation of the red and blue planes is the following. To improve the red and blue interpolation we would like to take out the effects of varying light intensity. Since green is very close to luminance, interpolation should be done based on the ratios of blue to green and red to green. For non-uniform lighting, the ratios of blue to green and red to green remain constant within an object of a given color [8].

An improvement over the linear approach would be the following. Since green pixels are the most abundant, do a linear interpolation over the green pixels first. Next, to obtain the missing blue, use the green pixels together with the known blue to do a linear interpolation of the ratio blue to green. Similarly, use the ratio red to green for interpolating the red channel. This method improves the results of linear interpolation, but still smears edges, since the green interpolation is nothing else but a low pass filter.

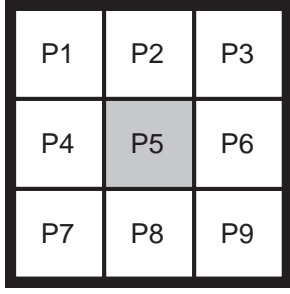
An improvement over this method still, is to interpolate along edges and not across them. Looking at Fig. 3, let  $E_i$  denote the likelihood that pixel  $P_5$  and pixel  $P_i$  belong to the same edge. In other words, if  $P_5$  and  $P_i$  are part of the same object then  $E_i$  is close to one, otherwise it's close to zero. With this  $E_i$ , the improved interpolation algorithm would then be:

- First, interpolate the greens using only the green information:

$$G_5 = \frac{E_2 G_2 + E_4 G_4 + E_6 G_6 + E_8 G_8}{E_2 + E_4 + E_6 + E_8} \quad (3)$$

- For the blues, we proceed in two steps. First interpolate the missing blues at the red locations

$$B_5 = G_5 \frac{E_1 \frac{B_1}{G_1} + E_3 \frac{B_3}{G_3} + E_7 \frac{B_7}{G_7} + E_9 \frac{B_9}{G_9}}{E_1 + E_3 + E_7 + E_9} \quad (4)$$



**Fig. 3.** Pixel  $P_5$  is the pixel that we are trying to interpolate.

and second, interpolate the missing blues at the green locations.

$$B_5 = G_5 \frac{E_2 \frac{B_2}{G_2} + E_4 \frac{B_4}{G_4} + E_6 \frac{B_6}{G_6} + E_8 \frac{B_8}{G_8}}{E_2 + E_4 + E_6 + E_8} \quad (5)$$

- Interpolate red the same way we did blue.

All that is left to do now, is to choose a proper  $E_i$  function. In [7] the function  $E_i$  was defined as follows:

- 1). For green (i.e. we are trying to interpolate the green at pixel  $P_5$ ):

$$\begin{cases} E_2 = E_8 = 1; E_4 = E_6 = 0 & \text{if } \begin{cases} |G_4 - G_6| > T \\ |G_2 - G_8| < T \end{cases} \\ E_2 = E_8 = 0; E_4 = E_6 = 1 & \text{if } \begin{cases} |G_4 - G_6| < T \\ |G_2 - G_8| > T \end{cases} \\ E_2 = E_4 = E_6 = E_8 = 0 & \text{else} \end{cases}$$

- 2). For red and blue (i.e. we are trying to interpolate the red or blue at location  $P_5$ ):

$$E_1 = E_3 = E_7 = E_9 = 1;$$

$$\begin{cases} E_2 = E_8 = 1; E_4 = E_6 = 0 & \text{if } P_2 \text{ and } P_8 \text{ are blue in CFA} \\ E_2 = E_8 = 0; E_4 = E_6 = 1 & \text{if } P_4 \text{ and } P_6 \text{ are blue in CFA} \end{cases}$$

The above edge classification scheme is somewhat simplistic. In [4], function  $E$  is enhanced by using gradients in a more sophisticated way. From Fig. 3 we define our derivatives at pixel  $P_5$ , in the  $x$ ,  $y$ ,  $x$ -diagonal and  $y$ -diagonal directions as follows:

$$\begin{aligned} D_x(P_5) &= \frac{P_4 - P_6}{2} & D_y(P_5) &= \frac{P_2 - P_8}{2} \\ D_{xd}(P_5) &= \frac{P_3 - P_7}{2\sqrt{2}} & D_{yd}(P_5) &= \frac{P_1 - P_9}{2\sqrt{2}} \end{aligned}$$

Notice that differences are always from the same color plane. If pixel  $P_5$  is a green pixel then so is  $P_1, P_3, P_7$  and  $P_9$ . To obtain the derivatives in the diagonal directions at a green

pixel, we can do a little better by defining our diagonal derivatives as:

$$\begin{aligned} D_{xd}(P_5) &= \max \left\{ \left| \frac{P_3 - P_5}{\sqrt{2}} \right|, \left| \frac{P_7 - P_5}{\sqrt{2}} \right| \right\} \\ D_{yd}(P_5) &= \max \left\{ \left| \frac{P_1 - P_5}{\sqrt{2}} \right|, \left| \frac{P_9 - P_5}{\sqrt{2}} \right| \right\} \end{aligned}$$

With the above derivatives, [4] defines function  $E_i$ , for red, green and blue, as:

$$E_i = \frac{1}{\sqrt{1 + D(P_5)^2 + D(P_i)^2}}, \quad (6)$$

where  $D$  is the derivative in the direction of  $P_i$ . As two examples,  $E_6 = (1 + D_x(P_5)^2 + D_x(P_6)^2)^{-1/2}$  and  $E_3 = (1 + D_{xd}(P_5)^2 + D_{xd}(P_3)^2)^{-1/2}$ .

Next, if the ratio blue to green is constant within an object, so must be the ratio green to blue. If the ratio green to blue is constant within an object, then locally the ratio green to blue must be the average of the neighboring ratios. The green values are then adjusted appropriately. Of course, this offsets the original blue to green ratio. The authors of [4] go back and fourth three times correcting for the ratio rule of both blue and red. The correction step of [4] is approximately the following.

- Repeat three times:
- Correct the Green values to fit the green over blue ratio test

$$G_5^B = B_5 \frac{E_2 \frac{G_2}{B_2} + E_4 \frac{G_4}{B_4} + E_6 \frac{G_6}{B_6} + E_8 \frac{G_8}{B_8}}{E_2 + E_4 + E_6 + E_8} \quad (7)$$

$$G_5^R = R_5 \frac{E_2 \frac{G_2}{R_2} + E_4 \frac{G_4}{R_4} + E_6 \frac{G_6}{R_6} + E_8 \frac{G_8}{R_8}}{E_2 + E_4 + E_6 + E_8} \quad (8)$$

and average between the Blue and Red interpolation results

$$G_5 = \frac{G_5^B + G_5^R}{2}$$

- Correct the Blue and Red values via the ratio rule

$$B_5 = G_5 \frac{\sum E_i \frac{B_i}{G_i}}{\sum E_i}, \text{ with } i \neq 5 \quad (9)$$

$$R_5 = G_5 \frac{\sum E_i \frac{R_i}{G_i}}{\sum E_i}, \text{ with } i \neq 5 \quad (10)$$

- End of loop.

The enhancement steps of equations (7)-(10) tend to introduce some slight artifacts that can almost be classified as noise. In [4] this noise is taken out by a smoothing step.

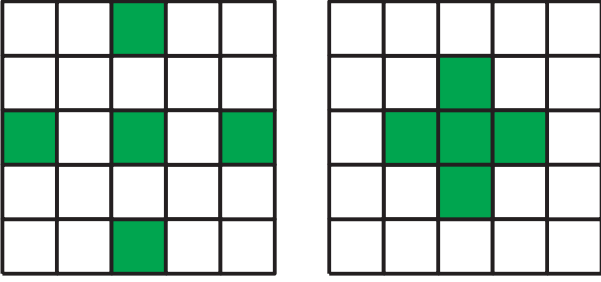


Fig. 4. Coarse (left) and fine scale training vectors.

#### 4. OPTIMAL DEMOSAICING

The approach of our interpolation is different than those of the previously mentioned approaches. Specifically, we interpolate the greens using optimal recovery and then interpolate the differences (instead of ratios) between reds and greens and the blues and greens, to find the reds and blues. The key advantage to our approach is that the greens are interpolated better. The interpolation of green is done in two steps.

- Interpolate the greens using optimal recovery and coarse scale training vectors. The training set  $S$  contains coarse scale vectors, Fig. 4, taken from the neighborhood of the vector we are trying to estimate.
- Once we estimate the missing pixels, rerun the optimal recovery interpolation, but this time use a training set based on the fine scale vectors of Fig. 4. Experimentally, the second step is necessary only for high frequency patterns. It tends to improve the results, but it may not always be necessary.

As we have discussed in [1], the selection of the training set  $S$  is key to the performance of optimal recovery interpolation. For the demosaicing case, training set  $S$  was obtained by selecting the nearest vectors (both in space and value) to the vector we were estimating. More specifically, the training set  $S$  was formed as follows.

- Decide on a window size for selecting the training set  $S$ . If the size is too large, local feature of the image are missed. Smaller window sizes seem to perform better than larger ones. Experimentally, a window of  $15 \times 15$  pixels around the pixel we are estimating seems to perform well for a  $256 \times 256$  image.
- In this window, which defines closeness in space, select all the possible training vectors. Call this set  $S$ .
- Prune  $S$  by selecting only half the vectors. Take the half of the vectors that are closest to the vector we are trying to estimate, based on the known samples. This defines a closeness in value.

The last step of the formation of the training set  $S$  is critical, especially at boundary regions of two different textures. For example, in the light house picture, at the boundary between the fence and the grass, if the last step of the selection of  $S$  is not performed, the fence pattern seems to be repeated into the grass.

Once the green pixels are estimated, to find the reds and blues we can run optimal recovery on the difference of the pairs of red and green and blue and green. Alternatively, we could also use any other edge directed interpolation method, reviewed in the previous section, to interpolate the differences. Experimentally, we have found that the most important interpolation is getting the green right. In our interpolation we used optimal recovery in interpolating the greens and then applied a modified method of [4] to interpolate and then correct the blues and reds. In particular, instead of interpolating the ratios we interpolated the differences between red and green and blue and green. The complete interpolation algorithm is this:

- First, interpolate the greens using optimal recovery, possibly taking a second pass using fine scale training vectors.
- For the blues, we proceed in two steps. First interpolate the missing blues at the red locations

$$B_5 = G_5 + \frac{\sum_{i=1,3,7,9} E_i (B_i - G_i)}{E_1 + E_3 + E_7 + E_9} \quad (11)$$

and second, interpolate the missing blues at the green locations.

$$B_5 = G_5 + \frac{\sum_{i=2,4,6,8} E_i (B_i - G_i)}{E_2 + E_4 + E_6 + E_8} \quad (12)$$

- Interpolate red the same way we did blue.

Next, add the correction step:

- Repeat three times:
- Correct the Green values to fit the green over blue ratio test

$$G_5^B = B_5 + \frac{\sum_{i=2,4,6,8} E_i (G_i - B_i)}{E_2 + E_4 + E_6 + E_8} \quad (13)$$

$$G_5^R = R_5 + \frac{\sum_{i=2,4,6,8} E_i (G_i - R_i)}{E_2 + E_4 + E_6 + E_8} \quad (14)$$

and average between the Blue and Red interpolation results

$$G_5 = \frac{G_5^B + G_5^R}{2}$$

- Correct the Blue and Red values via the ratio rule

$$B_5 = G_5 + \frac{\sum E_i (B_i - G_i)}{\sum E_i}, \text{ with } i \neq 5 \quad (15)$$

$$R_5 = G_5 + \frac{\sum E_i (R_i - G_i)}{\sum E_i}, \text{ with } i \neq 5 \quad (16)$$

- End of loop.

## 5. RESULTS

We applied our interpolation algorithm to the lighthouse and sails image. The PSNR values for the reconstructed lighthouse image are given in Table 1 and the PSNR values for

Method	Red	Green	Blue
Linear	23.71	27.76	23.93
Kimmel	32.72	36.53	31.30
OR	35.16	38.12	35.77

**Table 1.** Light House PSNR values (db)

the reconstructed sails image are given in Table 2. Although

Method	Red	Green	Blue
Linear	26.36	29.72	26.58
Kimmel	34.94	39.12	35.71
OR	37.54	40.06	38.17

**Table 2.** Sails PSNR values (db)

the PSNR is not necessarily a good measure of image quality, both tables reflect an average improvement of at least 2db over [4] and an average improvement of at least 10db over linear interpolation.

For completeness, we also included image results. In Fig. 5 we plotted the blue channels for the lighthouse and the red channels for the sails. (The gray scale results are not as convincing as the color images and we strongly encourage the reader to see the color results on line at [http:// dsplab.ece.cornell.edu](http://dsplab.ece.cornell.edu).) Notice how the lighthouse image in the optimal demosaicing approach has completely removed the aliasing artifacts, on the house, existent in the other approach. On the sails image, the numbers of the optimal mosaic image are sharper and there is less discoloration around number one.

## 6. CONCLUSION

In this paper we applied the optimal recovery interpolation model, presented in [2], to the problem of CFA interpolation. Through examples we have shown an improvement of at least 10db over the linear interpolation model and 2db over a recently published directional interpolation algorithm [4].

## 7. REFERENCES

- [1] D. D. Muresan and T. W. Parks, "Training Set Selection in Optimal Recovery Interpolation," *Submitted to: IEEE ICIP 2002*. Also at <http://dsplab.ece.cornell.edu>.
- [2] D. Darian Muresan and Thomas W. Parks, "Optimal Recovery Approach to Image Interpolation," *ICIP 2001, Greece*, 2001.
- [3] Glotzbach, J.W.; Schafer, R.W.; Ilgner, K. , "A method of color filter array interpolation with alias cancellation properties," *Image Processing, 2001. Proceedings. 2001 International Conference on*, Volume: 1 , pages: 141 -144, 2001.
- [4] Ron Kimmel, "Demosaicing: Image Reconstruction from Color CCD Samples," *IEEE Transactions on Image Processing*, Vol. 8, No. 9, pp. 1221-1228, (September 1999).
- [5] J.E. Adams, Jr., "Design of Practical Color Filter Array Interpolation Algorithms for Digital Cameras," *Proceedings of SPIE*, D. Sinha, ed. Vol. 3028, pp. 117-125, SPIE, Bellingham, WA. 1997.
- [6] D. R. Cok, "Reconstruction of CCD Images Using Template Matching," *Proc. of IS&T's Annual Conference/ICPS*, 380-385, 1994.
- [7] D. R. Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal," U.S. Patent 4,642,678, 1987.
- [8] Gunter Wyszecki and W. S. Stiles, "Color Science: Concepts and Methods, Quantitative Data and Formulae," *John Wiley and Sons* , 1982.
- [9] M. Golomb and H. F. Weinberger, "Optimal approximation and Error Bounds," *On Numerical Approximation*, R. E. Langer ed., The University of Wisconsin Press, Maddison, pp. 117-190, 1959.



**Fig. 5.** Optimal Demosaic (top row) and Kimmel (bottom row)