Color demosaicing in YUV color space

Jayanta Mukhopadhyay
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur
India 721302.
Email: jay@cse.iitkgp.ernet.in

Manfred K. Lang
Institute for Human-Machine Communication,
Technical University of Munich,
D-80290 Munich, Germany
Email: lang@mmk.ei.tum.de

Sanjit K. Mitra
Department of Electrical and Computer Engineering,
University of California, Santa Barbara,
CA, 93106, USA.
Email: mitra@ece.ucsb.edu

ABSTRACT
In a single-chip digital color imaging sensor, a color filter array (CFA) is used to obtain sampled spectral components (red, green and blue) in an interleaved fashion. Color demosaicing is the process of interpolating these regularly spaced sampled values into the dense pixel maps for each spectral components. In this paper we present techniques for interpolating the color images in the YUV color space. The resulting interpolated images could be directly used in the DCT based JPEG compression scheme. As the final results are desired in the DCT space, we have also used the concept of subband DCT computation for interpolating the individual components of color images in the DCT domain. Based on a simple strategy for computing Y, U and V components, several modifications are also proposed for improving the quality of the reconstructed images. We have observed that median filtering of the chrominance components improves the end results remarkably.

KEY WORDS
Digital Color Imaging, Color Interpolation, Color Demosaicing, Color Filter Array (CFA), RGB color space, YUV color space, Discrete Cosine Transform (DCT), Subband DCT.

1. Introduction

Single-sensor cameras [1], [2], [3], [4], for color image acquisition use color filter arrays (CFAs) to obtain sampled red, green and blue pixel data (or *luminance* and *chrominance* signals) in an interleaved fashion. To this end, different checker-board patterns are used as color filter arrays (CFAs) [3], of which the Bayer pattern CFA [5] (shown in Figure 1) is more commonly employed and is considered in this paper. In Figure 1 the sampled color components are denoted by R (for red), G (for green) and B (for blue).

G	R	G	R
B	G	B	G
G	R	G	R
B	G	B	G

Figure 1. The Bayer pattern.

From the sampled color pixel data, the missing color pixel values are interpolated to obtain dense pixel maps in all three spectral components. The process of interpolating these sparse data into dense pixel maps is commonly known as *color interpolation* or *color demosaicing*. Interestingly many of the digital cameras provide the output interpolated image in the DCT based JPEG compression standard. Most of the existing interpolation algorithms are based on the RGB color space. This implies that further computations are required for converting these interpolated images into the YUV space, which are subsequently transformed into DCTs and subjected to different encoding stages of the JPEG compression scheme [6]. In our work we propose to reduce this overhead by directly interpolating in the YUV space. Moreover, we have used the concept of subband DCT computation [7] for image interpolation in the DCT-domain. In [8], algorithms for image resizing using subband DCT computations are presented. We have used the image-upsampling (or image-doubling operation as called in [8]) algorithm for the purpose of interpolation in this work. In this algorithm, DCT coefficients for every 8×8 block are converted to DCT coefficients of a 16×16 block using subband DCT theory [7]. Later, inverse DCTs of 16×16 blocks provide the upsampled images. The details are discussed in [8].

In the next section, we present our algorithms of color interpolation. Finally, we have discussed the merits and demerits of the proposed techniques against other known methods.

2. Color Interpolation in YUV space

Let us partition the CFA in smaller blocks of sizes 2×2, where each block consists of the color masks as depicted in Figure 2.

Let us also denote the pixels in the corresponding 2×2 blocks in the mosaiced image as shown in Figure 3. Let the functions for converting a pixel having r, g and b

G	R
B	G

Figure 2. A 2×2 block in the CFA.

g_{11}	r_{12}
b_{21}	g_{22}

Figure 3. Pixels in the mosaiced image for the corresponding 2×2 block.

values (for R, G and B spectral components respectively) to a pixel in YUV space be denoted as:

$$\begin{aligned} y &= Y(r,g,b) \\ u &= U(r,g,b) \\ v &= V(r,g,b) \end{aligned} \quad (1)$$

The transformation is linear and given by the following equation:

$$\begin{bmatrix} y \\ u \\ v \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.173 & -0.339 & 0.511 \\ 0.511 & -0.428 & -0.083 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad (2)$$

Now, the algorithm is described below:
Algorithm *Simple_YUV_interpolation (SYUV)*
Input: Mosaiced image obtained through the Bayer CFA.
Output: Interpolated image in the compressed domain.
Begin

1. For each 2×2 block of the input image do the following:
   {
   (a) Compute y, u and v values from r_{12}, g_{av} and b_{21} as follows,

   $$\begin{aligned} g_{av} &= (g_{11} + g_{22})/2 \\ y &= Y(r_{12}, g_{av}, b_{21}) \\ u &= U(r_{12}, g_{av}, b_{21}) \\ v &= V(r_{12}, g_{av}, b_{21}) \end{aligned} \quad (3)$$

   (b) Store y, u and v values as downsampled Y, U and V components respectively.

   }
2. Compute DCT of Y and upsample it using SBDCT computation as described in [8].
3. Compute DCT's of downsampled U and V components.

Table 1. Recovery of individual components using SYUV interpolation

images	PSNR		
	Y (dB)	U (dB)	V (dB)
Statue	28.49	31.16	31.88
Lighthouse	24.92	28.68	29.40
Window	27.71	30.25	30.88
Sail	27.67	31.19	31.75
Pepper	26.83	29.95	29.90

End *Simple_YUV_interpolation (SYUV)*

It may be noted that as in the JPEG compression standard, downsampled U and V components are used, it is not necessary to upsample them. However, for the Y component it is necessary to upsample. For experimentations, we created the Bayer pattern array from a number of original color images. Next, full color images were generated from each of the Bayer pattern arrays using the SYUV algorithm described above. In Figure 4 we present a typical example of image-reconstruction. It may be noted that in presenting our results (also in subsequent sections) we do not perform any kind of quantization or compression on the DCT coefficients. One can observe that the quality of the reconstructed image is poor and many of the details are blurred. In some cases, false colors also appear near the edges (specially near 'achromatic' edges). The performance of the algorithm in recovering individual Y, U and V components for various images are shown in Table 1.

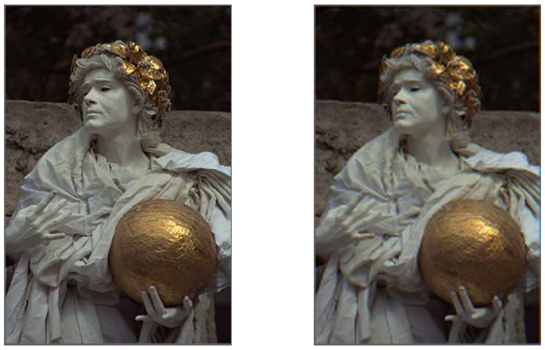
(a) Statue (original) (b) by SYUV

Figure 4. Reconstructed images by SYUV

Interestingly, one could observe from Table 1 that U and V components are reconstructed more reliably than the Y component. The justifications for this property could be given from the fact that the transformations of the image in YUV from the RGB space are linear. For a downsampled U and V components, values are formed by the averages over red, green and blue components in a 2×2 block. Hence, the expected deviations of the representative (true) sample values in the mosaiced image from the average values are less (following the *central limit theorem* and assuming that the probability distribution of a spectral component in a small neighborhood is

Table 2. Recovery of individual components using YUVG interpolation

images	PSNR		
	Y (dB)	U (dB)	V (dB)
Statue	32.85	31.50	32.13
Lighthouse	30.68	28.90	29.67
Window	32.50	30.41	31.25
Sail	33.16	31.46	32.14
Pepper	29.35	30.22	30.50

Gaussian). But this is not true for Y. Here, we have to interpolate them in full resolution. Hence we have considered following strategies for improving the performance of our proposed technique.

2.1 Computations through 'green' interpolation

For improving the performance of SYUV algorithm, we have to improve the quality of reconstruction of the Y component. As the green component plays the dominant role in determining the luminance (Y) component (see Eq. (2)), we propose to carry out interpolation of the green component in the first step using any conventional technique and then use these values to determine Y, U and V subsequently. In our work we have used an edge correlated interpolation technique for interpolating the green component [9]. In these techniques [10], [11], horizontal and vertical gradient values are used in the interpolation computation. The algorithm makes use of only the pixel values in the sampled array in the interpolation which are lying along the least gradient path (either horizontal or vertical). We have implemented the method proposed by Hamilton et al. [9]. In this case the interpolated values are corrected from the second order derivatives of the spectral components. For a missing green value at a blue (red) pixel in the sampled array, the second order derivatives of the blue (red) pixel values along the same direction are added with the average green values.

In Figure 5(a), the reconstructed image obtained by this algorithm (to be called the YUVG algorithm) is shown. One could observe that there is a considerable improvement in the quality of the reconstruction. But in some cases one could observe false colors near the edges. The improvement in the recovery of Y components (around $\simeq 4$ dB gain) could be observed in Table 2. Interestingly, marginal improvements in the recoveries of U and V components are also observed on account of using interpolated 'green' values.

2.2 Post-processing U and V components

One of the advantages in having the images in YUV space is that achromatic and chromatic components are well separated in this color space. Here Y represents the luminance or achromatic component. On the other hand, U and V represent chrominance components. This helps in tackling the appearances of false colors near the achromatic edges of the color image. We have adopted a very simple strategy for suppressing false colors. We model false colors as noisy samples of the U and V components. Visibly they occur in isolated regions with sharp discontinuities. Hence, we modeled them as 'salt and pepper' noise. We have reduced this noise in our work using median filtering [12] (with a 3×3 mask). We found remarkable improvements in the recoveries of the U and V components (see Table 3). In some cases the gains in the PSNR value (with respect to YUVG results) are as high as 9 dB (approximately). Presences of false colors also are significantly suppressed in these cases (Figure 5(b)). In our work we refer the modified YUVG algorithm using median filtering of U and V components as the YUVGM algorithm.

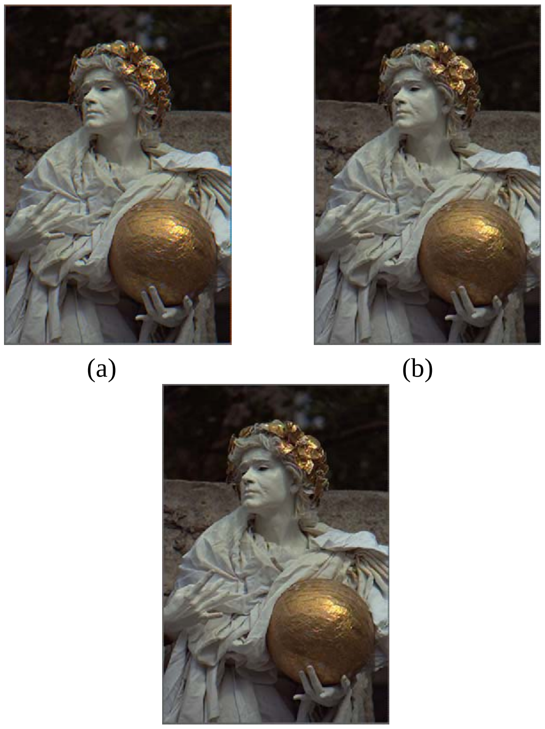

Figure 5. Reconstructed images by : (a) YUVG (b) YUVGM and (c) YUVGMSB

Table 3. Recovery of individual components using YUVGM interpolation

images	PSNR		
	Y (dB)	U (dB)	V (dB)
Statue	32.91	40.38	41.82
Lighthouse	30.77	35.23	35.97
Window	32.53	36.52	37.96
Sail	33.25	39.60	40.90
Pepper	29.46	34.25	33.65

Table 4. Recovery of individual components using YUVGMSB interpolation

images	PSNR		
	Y (dB)	U (dB)	V (dB)
Statue	32.91	41.05	42.58
Lighthouse	30.77	36.41	37.09
Window	32.54	37.72	39.11
Sail	33.25	40.42	41.74
Pepper	29.47	34.66	34.03

Table 5. Average pixel processing times (in μs)

SYUV	YUVG	YUVGM	YUVGMSB
1.75	1.07	4.37	11.99

2.3 Sub-band Interpolations of U and V components

It has been observed that the interpolated images could be further improved if one performs upsampling of the U and V components (instead of replications as done in the JPEG compression scheme) using subband DCT computations [8]. In Table 4 we present these improved performances. We call the modified YUVGM algorithm with interpolations of the U and V components using sub-band DCT computations as the YUVGMSB algorithm. We found in some cases the gains in the recoveries (with respect to the YUVGM results) are more than 1 dB. The interpolated images using this technique are shown in Figure 5(c). However, the interpolations of U and V components using subband DCT computations imposes considerable overhead in the computation. We have elaborated these observations in the next section.

2.4 Computational overheads

Each modification of the algorithm is likely to increase the computational overhead. We present the average pixel processing times for each technique in Table 5. The algorithms were implemented on a Linux-7.3 Work Station. The processor is Pentium-III with 550 MHZ clock. One can observe that subband DCT computations take considerable time in interpolating the components. Even the simple algorithm SYUV, where Y is interpolated using SBDCT, takes more time than YUVG, where no such computation is carried out. similarly one may note that YUVGMSB takes considerable time in interpolating U and V components using subband DCT computations.

3. Evaluation of algorithms

To compare the performances of the proposed interpolation algorithms, We have considered the following aspects of the algorithms in our study:

1. Quality of image reconstructions,
2. Memory requirements,
3. Speed of computations, and
4. False color suppression.

For judging the quality of the reconstructed image we have used the CPSNR (Composite-Peak-Signal-to-Noise-Ratio) measures. Let $I_s(x,y), s = 1,2,3$ be the spectral components of a benchmark image of size $M \times N$ and $I_s^{'}(x,y), s = 1,2,3$ be the respective reconstructed spectral components. Then CPSNR is defined as:

$$CPSNR = 20 * \log_{10}\left(\frac{255}{\sqrt{\frac{\Sigma_{\forall s}\Sigma_{\forall x}\Sigma_{\forall y}(I_s(x,y)-I_s^{'}(x,y))^2}{3MN}}}\right) \quad (4)$$

Since CPSNR measures do not always reflect the quality of the images in terms of edge reconstructions, we have also used here another measure, PEINR (Peak-Edge-Intensity-to-Noise-Ratio) for reflecting how edges are recovered in the interpolated images. For defining PEINR we have used the binary edge map of an image, which is computed from the gradient image of the sampled array. In an edge map $e(x,y)$ of an image, if the value at a pixel location is 1 it shows the presence of edge pixels and otherwise the value is 0. Then, PEINR is defined as:

$$PEINR = 20*\log_{10}\left(\frac{255}{\sqrt{\frac{\Sigma_{\forall s}\Sigma_{\forall x}\Sigma_{\forall y}e(x,y)\times(I_s(x,y)-I_s^{'}(x,y))^2}{3\Sigma_{\forall x}\Sigma_{\forall y}e(x,y)}}}\right) \quad (5)$$

We present our experimental results with a set of images used earlier by the researchers. In our comparative studies, we have considered two typical existing algorithms. One of them is the simple *bilinear interpolation technique*, as many work reported earlier compared their results with this one. The other one is a very good interpolation technique, which provide good quality of color image reconstruction as well as good suppression of false colors. This is the technique proposed by Kimmel [13] who has used the principles of edge directed cross-ratio averaging and corrections. In his technique Kimmel used the directional derivatives for assigning the weights of the cross-ratio averaging and color adjustments. We will refer his technique in our work as EDCRAC (as noted earlier also in [14]). Among our proposed algorithms, we present the results for both YUVGM and YUVGMSB. Though the quality of reconstructions for YUVGMSB is found to be marginally better than YUVGM, YUVGMSB takes significantly large pixel processing times. Hence we felt it is worth to consider the applicability of YUVGM also.

3.1 Quality of Reconstructions

In Figures 6 we present the reconstructed images using the bilinear interpolation technique as well using Kimmel's algorithm. In Tables 6 and 7 we present the CPSNR and PEINR values for different images obtained by different techniques. In the tables, the maximum values obtained by any of the methods considered here are highlighted by bold numerals. It may be noted that EDCRAC

Table 6. **CPSNR for Different Interpolation Techniques**

Images	YUVGM (dB)	YUVGMSB (dB)	Bilinear (dB)	EDCRAC (dB)
Statue	31.87	**31.97**	27.85	31.58
Lighthouse	28.85	29.10	25.09	**30.40**
Window	30.16	30.55	26.84	**32.69**
Sail	31.82	32.02	27.69	**33.45**
Pepper	26.52	**26.80**	25.45	25.04

Table 7. **PEINR for Different Interpolation Techniques**

Images	YUVGM (dB)	YUVGMSB (dB)	Bilinear (dB)	EDCRAC (dB)
Statue	**17.95**	17.92	12.91	13.57
Lighthouse	18.80	18.59	**19.61**	17.35
Window	16.44	16.26	15.17	**24.18**
Sail	18.79	18.96	18.46	**20.97**
Pepper	**14.43**	14.40	11.51	11.77

performs very well in most of the cases. But, the performances of YUVGM and YUVGMSB are also quite close to the performance measures achieved by EDCRAC. In some cases, in fact, they have marginally outperformed it.

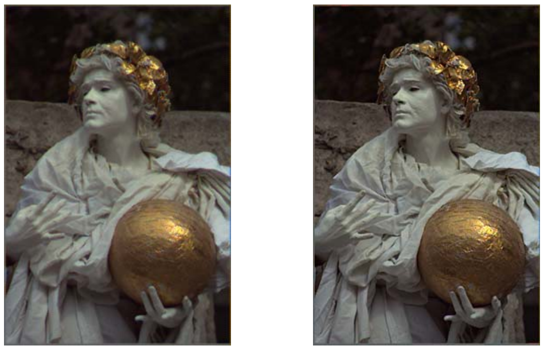
(a) Statue (BI) (b) Statue (EDCRAC)

Figure 6. Reconstructed images by bilinear interpolation and Kimmel's algorithm (EDCRAC)

3.2 Computation speed

We have also measured the computation times for different interpolation techniques. We present here the average pixel processing times (see Table 8). It may be noted that we do not claim that all methods have been efficiently implemented. Unless there is a significant difference in the computation time one should not conclude about their relative speed of computation. One should also consider the overhead due to the conversion from RGB space to YUV color space, downsampling the U and V components and finally transforming them in DCTs. Interestingly, though the performances of YUVGM are close to EDCRAC, it takes significantly less amount of time in computation.

Table 8. Average pixel processing times (in μs)

YUVGM	YUVGMSB	Bilinear	EDCRAC
4.37	11.99	2.97	9.65

3.3 Memory requirements

One could easily observe that the memory requirements in the proposed schemes are significantly less. This is due to the fact that only the downsampled versions of U and V are produced. Hence for a mosaiced pattern of size $M \times N$, the memory requirement for both YUVGM and YUVGMSB is $2 \times (M \times N) + 2 \times (\frac{M}{2} \times \frac{N}{2})$. But on the other hand for the conventional algorithms in the RGB space the memory requirement is $4 \times (M \times N)$. It may be noted however that in computing the memory requirements we have considered one additional buffer for storing the mosaiced pattern obtained using the CFA. One may translate this advantage of the smaller memory requirements for the proposed schemes in efficient VLSI implementations.

3.4 False Color Suppression

It has been pointed out earlier that median filtering of the U and V components significantly reduce the appearances of false colors. We demonstrate these features by enlarging the extracted portions of the interpolated images (see Figure 7), where false colors are visibly perceptible in different interpolation algorithms. However, it should be noted that Kimmel's algorithm (EDCRAC) also significantly reduces false colors.

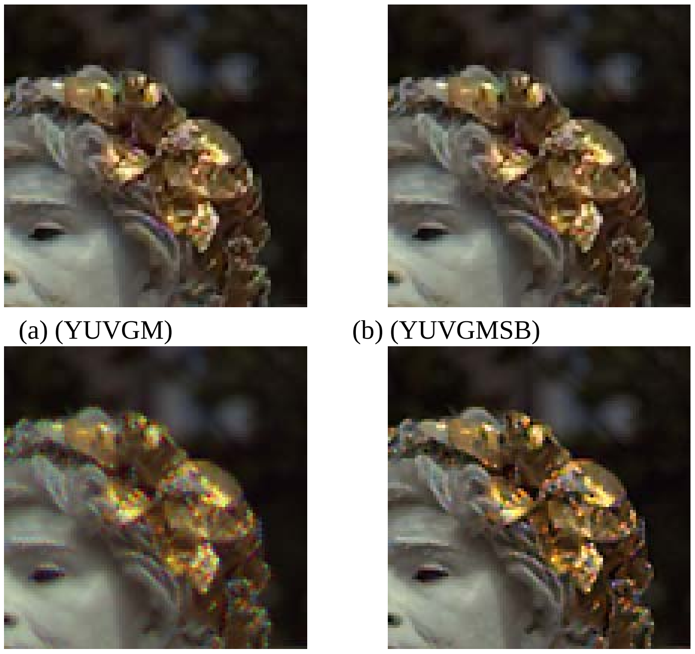
(a) (YUVGM) (b) (YUVGMSB)
(c) (BI) (d) (EDCRAC)

Figure 7. Examples of false color suppressions in an enlarged part of Statue

4. Conclusion

In this paper we present techniques for interpolating the color images in the YUV color space. The resulting interpolated images could be directly used in the DCT based JPEG compression scheme. As the final results are desired in the DCT space, we have also used the concept of subband DCT computation for interpolating the individual components of color images in the DCT domain. Based on a simple strategy for computing the Y, U and V components, several modifications are also proposed for improving the quality of the reconstructed images. We have observed that median filtering of the chrominance components improves the end results remarkably. The results are also compared with the existing good interpolation algorithms and they are found to have competitive advantages considering the speed of computations, quality of image reconstruction, storage requirement and false color suppression.

Acknowledgement

The first author gratefully acknowledges the Alexander von Humboldt Foundation , Germany for awarding him the research fellowship, under which the present work has been carried out.

References

[1] J. A. Weldy, “Optimized design for a single-sensor color electronic camera system,” *Proceedings of SPIE*, vol. 1701, pp. 300–307, 1988.

[2] K. A. Parulski, “Color filters and processing alternatives for one-chip cameras,” *IEEE Transaction on Electron Devices*, vol. ED - 32, no. 8, pp. 1381 – 1389, Aug. 1985.

[3] Y. T. Tsai, “Color image compression for single-chip cameras,” *IEEE Transaction on Electron Devices*, vol. 38, no. 5, pp. 1226 – 1232, 1991.

[4] C. H. Wu, “Reconstruction of color images from a single-chip ccd sensor based on markov random field models,” *Proceedings of SPIE*, vol. 2564, pp. 282–288, 1995.

[5] B. E. Bayer, “Color imaging array,” , U.S. Patent No., 3971065, July, 1976.

[6] R. C. Gonzalez and R.E.Woods, *Digital Image Processing*, Addison-Wesley, 1993.

[7] S. -H. Jung, S.K. Mitra, and D. Mukherjee, “Subband DCT: Definition, analysis and applications,” *IEEE Transactions on Circuits and systems for Video Technology*, vol. 6, no. 3, pp. 273–286, June 1996.

[8] J. Mukherjee and S.K. Mitra, “Image resizing in the compressed domain using subband dct,” *IEEE Transactions on Circuits and systems for Video Technology*, to appear.

[9] J. F. Hamilton, Jr., and J. E. Adams, Jr., “Adaptive color plane interpolation in single color electronic camera,” , US Patent No., 5629734, May, 1997.

[10] R. H. Hibbard, “Apparatus and method for adaptively interpolating a full color image utilizing luminance gradients,” , US Patent No., 5382976, January, 1995.

[11] D. R. Cok, “Signal processing method and appratus for sampled image signals,” , U.S. Patent No., 4630307, December, 1986.

[12] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall of India Private Limited, New Delhi, 1995.

[13] R. Kimmel, “Demosaicing : Image reconstruction from color ccd samples,” *IEEE Transactions on Image Processing*, vol. 8, no. 9, pp. 1221 – 1228, Sep. 1999.

[14] J. Mukherjee, R. Parthasarathi, and S.K.Goyal, “Markov random field processing for color demosaicing,” *Pattern Recognition Letters*, vol. 22, pp. 339–351, 2001.