Color Interpolation for Cross-Talk Noise Reduction

Wen-Han Chen(陳文漢), Chiou-Shann Fuh(傅楸善)

Graduate Institude of Networking and Multimedia, National Taiwan University, Taipei, Taiwan

***Abstract*—Most digital cameras use single electronic sensors such as CCD (Charge-Coupled Device) or CMOS (Complementary Metal-Oxide-Semiconductor). To reduce cost, we always use Bayer pattern to get the other color information. Since white light is composed of light of different wavelengths. Since silicon has different absorption rates for different wavelengths. It will cause blocky cross-talk artifact. In this thesis, we propose a new color interpolation to reduce blocky information and successfully keep image detail.**

Keywords- color interpolation, cross-talk noise, false color

1. INTRODUCTION

Digital camera usually uses Color Filter Array (CFA) to capture scene color value. Using color filter array reduces cost. Because of color array, we face a special image structure, Bayer pattern [1]. Raw image contains mosaic data. Each pixel only has one color channel value, such as Green pixel between two Red pixels in the same row. If we want to get a complete color and good quality image, demosaicking Bayer pattern value is necessary, also called color interpolation. A good color interpolation algorithm can reduce false color and keep the detail successfully. In this thesis, we present a new color interpolation algorithm and it can remove common artifact, such as cross-talk which creates blocky noise or image edge zipper effect and blurred edge.

1.1 Color Filter Array

To reduce cost, Color Filter Array (CFA) arranges red, green, and blue pixel sensor as in Figure 1.1 instead of using three sensors at the same location to catch three primary colors of light. When the digital camera sensor catches the light, the color filter array will absorb the light based on different wavelengths of light. Furthermore, to transform the photon to electron signal, we get raw image, the original data which sensor captures. If we want to get the other color channel value for corresponding pixel, use color interpolation algorithm to get the other color information.

Initially, there are many types to arrange color pixel photosensor. But now most popular on consumer digital camera is Bayer filter. In 1976 Bryce Bayer's patent [1], he uses retina characteristic where rod cells are more sensitive to green light and human eye are sensitive to three primary color light. Therefore, he uses 50% green, 25% red, and 25% blue elements. The final image result is called Bayer pattern image.

Similarly, the other alternative filter on digital camera, such as RGBE (Red, Green, Blue, and Emerald) used in Sony digital camera serious, CYGM (Cyan, Yellow, Green, Magenta) uses alternative pattern to record three primary colors.

1.2 Pixel Cross-Talk

Figure 1 Blocky artifact.

For Charge-Coupled Device (CCD) or Complementary Metal-Oxide Semiconductor (CMOS) image sensor to capture scene value, cross-talk usually occurs. Neighbors may interfere with central pixel value. Many possible reasons, such as incident light angle may cause adjacent sensor to absorb redundant photons to convert into electronic signal. Moreover, silicon absorption ratios for light of different wavelengths are different. Horizontal and vertical adjacent pixels have different influences on neighbor pixels. Due to pixel layout direction, the horizontal neighbors have higher influence than vertical neighbors. Result image at the smooth region usually has blocky zigzag.

2. RELATED WORK

2.1 Color Interpolation Method

2.1.1 Nearest-Neighbor Interpolation

Nearest neighbor is the simplest color interpolation algorithm. Its concept is rounding-off error to get an approximation to get the nearest-neighbor integer value. Advantages are simple, easy, and fast implementation. On the other hand, disadvantages are zigzag between neighbors and obvious line artifact between each pixel.

2.1.2 Bilinear Interpolation

In one dimension, we usually use linear interpolation to get an unknown value. Most digital image use F(x): Z2 → N3 to describe one position with three integer color channel values. Thus at two dimensions, we use linear interpolation twice on horizontal and vertical directions.

2.1.3 Edge-Sensing Interpolation

Most common color interpolation algorithm has the same problems which produce false color on edge region. In human visual system, most people are sensitive at edge pixel and luminance change. According to the characteristic, we refer to the original Bayer pattern value to compute image gradient. Edge pixel has higher continuity than the other. Thus we interpolate blank pixel based on gradient.

2.2 Cross-Talk Compensation Method

Generally, there are two algorithms to compensate cross-talk [7]. Because human eyes are sensitive to green color, in Bayer pattern, we adjust Gr and Gb channel values. We estimate the gradient and local average to keep image sharpness and reduce cross-talk phenomenon.

2.2.1 Interpolation-Based Method

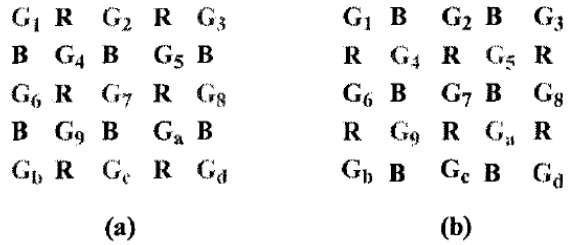


Figure 2 Interpolaton-based method [7]. (a) Bayer pattern has four possible arrangements, i.e. first column: one red value between Gr channel. (b) First column: one blue value between Gb channel.

In [7], based on an assumption, we try to equalize Gr and Gb channels. In Figure 2.4.a, if we use Gb as a reference channel, we want to get Gr at G7 location. We can assume that

$$G_7^{new} = G_7 + \Delta G_7$$

Where

$$\Delta G_7 = \frac{(\Delta G_4 + \Delta G_5 + \Delta G_9 + \Delta G_a)}{4}$$

$$\Delta G_4 = G_4 - \frac{(G_1 + G_2 + G_6 + G_7)}{4}$$

$$\Delta G_5 = G_5 - \frac{(G_2 + G_3 + G_7 + G_8)}{4}$$

$$\Delta G_9 = G_9 - \frac{(G_6 + G_7 + G_b + G_c)}{4}$$

$$\Delta G_a = G_a - \frac{(G_7 + G_8 + G_c + G_d)}{4}$$

Or we can simplfy equations

$$G_7^{new} = \frac{G_4 + G_5 + G_9 + G_a}{4} + \frac{12G_7 - [2(G_2 + G_6 + G_8 + G_c) + G_1 + G_3 + G_b + G_d]}{16}$$

choose Gr (Figure 2.4.b) or Gb as a reference channel, the most important thing is to keep the two channels equalized.

2.2.2 Average-Based Method

In [7], the other method to remove cross-talk exists. Use a local average to keep green channel balanced, without adjusting one of them. In Figure 8.a we can derive formula

$$G_7^{new} = \frac{G_7 + (\overline{G_r} - \overline{G_b})}{2}$$

$$\overline{G_r} = \frac{G_4 + G_5 + G_9 + G_a}{4}$$

$$\overline{G_b} = \frac{G_1 + G_2 + G_3 + G_6 + G_7 + G_8 + G_b + G_c + G_d}{9}$$

These algorithms can be implemented as an independent function before color interpolation. It helps for pipeline function implementation and performance improvement.

2.3 Color Difference Planes

In [8], for Bayer pattern, each position only has one channel value. But image has high correlation between each channel in a small region. We can use the characteristic to develop image model.

$$K_r = G - R$$

$$K_b = G - B$$

Kr means green value subtracted by red value, and Kb means green value subtracted by blue value. Use the new color domains Kr and Kb to interpolate red and blue planes. By transformation, we interpolate green channel and KrKb channel. Finally, we similarly interpolate blue and red channels. In Figure 10 we can find out it is smoother in hue domain. This is good for interpolation and reduces false color.

2.4 False Color Removal

In a full color image, we use a vector X(p,q) = (X(p,q)1, X(p,q)2, X(p,q)3) ∈ Z2 to describe three color components where X(p,q)1 means Red color value; X(p,q)2 means Green value; and X(p,q)3 means blue value. We post-process to reduce false color in final image [9]. Color interpolation aims to use original raw data for final color correction.

At Section 2.3, color correlates significantly in a small region. After getting the final image, we use three channels to correct color value for each position.

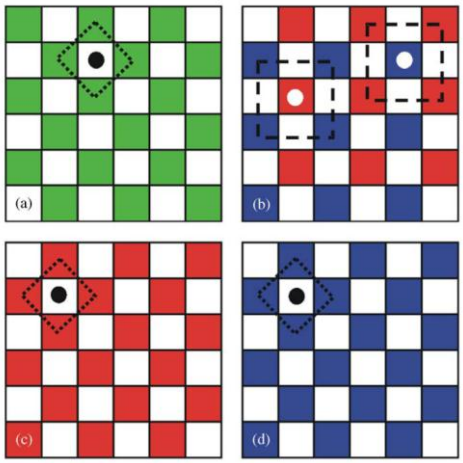


Figure 3 (a) First, fill green value with dimond mask. (b) Second, fill each blank blue or red central value with square mask. (c) and (d) Fill the other blank pixels [9].

In Figure 3(a) we select the original green value position at the corresponding raw data. Then use the following formula.

$$X_{(p,q)2} = X_{(p,q)k} + ? \frac{\sum_{(i,j)\in\zeta} w_{(i,j)}(x_{(i,j)2} - x_{(i,j)k})}{\sum_{(i,j)\in\zeta} w_{(i,j)}}$$

Moreover, weighting coefficient calculation is

$$w(i,j) = \frac{1}{1 + d(i,j)}$$

$$d(i,j) = \sum_{(g,h)\in\zeta} \left| x_{(i,j)2} - x_{(g,h)2} \right|$$

where $\zeta = \{(p-1,q),(p,q-1),(p,q+1),(p+1,q)\}$

Choose a diamond mask to select neighbors and adjust green value by color correlation.

Second, we find the red/blue corresponding position at original raw data. Select a square mask and fill central pixel value as follows.

$$X_{(p,q)k} = X_{(p,q)2} + \frac{\sum_{(i,j)\in\zeta} w_{(i,j)}(x_{(i,j)k} - x_{(i,j)2})}{\sum_{(i,j)\in\zeta} w_{(i,j)}}$$

where $\zeta = \{(p-1,q-1),(p-1,q+1),(p+1,q-1),(p+1,q+1)\}$ After second step, we use the previous diamond to fill red/blue plane and get the final image. It is good to implement pipeline and improve performance after we get the final image.

2.4 Edge Sensing Mechanism

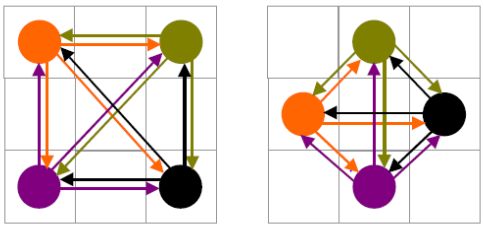
In Section 2.4, the weighting calculation uses edge sensing mechanism. We calculate gradient to decide weighting coefficient. If we get a great gradient value, it means that there is an edge across the small region. On the other hand, small gradient value means that mask covers a smooth region.

$$d(i,j) = \sum_{(g,h)\in\zeta} \left| x_{(i,j)k} - x_{(g,h)k} \right|$$

With large gradient magnitude, we need to reduce neighbor effect for central pixel. Practically, we usually use a simple function as follows

$$w(i,j) = \frac{1}{1+d(i,j)}$$

Use inverse function to control coefficient effect. Moreover, denominator always adds one to avoid *d(i,j)* to be too small.



(a) (b)

Figure 4 (a) Square lattice. (b) Quincunx lattice [10].

3. Our Proposed Method

In this chapter, we will present my method. First, we separate raw data in three planes: red, green, and blue planes. To reduce cross-talk noise in raw data, we use a reference channel such as Gr or Gb. Then adjust each green channel value according to local average value. Then, we use blue and red information to build color correlation planes Kr and Kb to avoid false color, but Kr and Kb have many blank positions. Thus we use linear interpolation twice to fill these blank pixels. Finally we inverse Kr and Kb planes to corresponding red and blue values and use a false color removal function to correct final image. this property, we can set a mask with border, if we find many edge pixels around middle pixel, but we can not find any edge pixel in the border of mask. By this judgment, this mask presumes it is a noise cluster, and eliminates it. In H.Y. Shen's method [5], this function can be canceled if we want to keep more detail. Usually, the mask size is 5*5~15*15 pixels.

3.1 Calculate Gradient Difference

$$\begin{array}{ccccc} G_1 & R_2 & G_3 & R_4 & G_5 \\ B_6 & G_7 & B_8 & G_9 & B_{10} \\ G_{11} & R_{12} & G_{13} & R_{14} & G_{15} \\ B_{16} & G_{17} & B_{18} & G_{19} & B_{20} \\ G_{21} & R_{22} & G_{23} & R_{24} & G_{25} \end{array}$$

Figure 5 Calculate four directions.

In real world, raw data usually have cross-talk noise. The artifact always rises at smooth region. We want to separate image region into smooth and non-smooth regions.

In this step, we calculate gradient in four directions. In Figure 3.1.2, interesting pixel is G_{13}.

$$H = |G_{11} - G_{15}|$$
$$V = |G_3 - G_{23}|$$
$$C_1 = |G_7 - G_{19}|$$
$$C_2 = |G_9 - G_{17}|$$

Symbol H means horizontal direction; V means vertical direction; C_1 and C_2 mean two diagonal directions. If the 5X5 mask covers a smooth region, these gradient differences will be small. We use the information to avoid false color with green channel compensation.

$$\text{If } \frac{(H + V + C_1 + C_2)}{4} > Threshold$$
$$\Rightarrow \textit{edge cross } 5\times5 \textit{ mask}$$
else
$$\Rightarrow \textit{smooth region}$$

3.2 Set Kr and Kb Planes and Interpolate

In this step we use original red and blue data to subtract corresponding green plane value. In Figure3.1.2, we want to set K_r plane first.

$$K_{r2} = G_2 - R_2$$
$$K_{r4} = G_4 - R_4$$
$$K_{r12} = G_{12} - R_{12}$$
$$K_{r14} = G_{14} - R_{14}$$

Previously, we have gotten an entire green channel plane to use this information to estimate K_r and K_b planes. Thus we will get K_r and K_b planes. There are many blank positions. We fill horizontal and vertical blank positions by linear interpolation between two K_r/K_b pixels.

$$K_{r3} = \frac{(K_{r2} + K_{r4})}{2}$$
$$K_{r7} = \frac{(K_{r2} + K_{r12})}{2}$$
$$K_{r9} = \frac{(K_{r4} + K_{r14})}{2}$$
$$K_{r13} = \frac{(K_{r12} + K_{r14})}{2}$$

Final remaining blank position will be near four K_r/K_b pixels. We use bilinear interpolation to fill these blank pixels.

$$K_{r8} = \frac{(K_{r3} + K_{r7} + K_{r9} + K_{r13})}{4}$$

3.3 Re-Interpolate G, Kr, and Kb

In Section 3.2, we set K_r/K_b by reference green channel value at corresponding position. To promote precision, we recalculate three channel values.

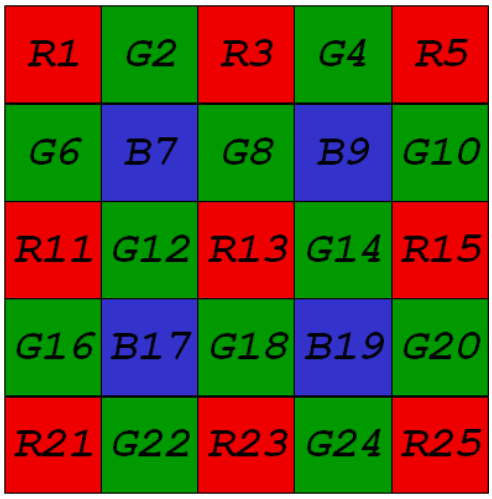


Figure 6 Original raw data format.

First we re-interpolate green channel plane. In Figure 3.1.3, we can find out B_7 has no green value in original data.

$$K_{b6} = G_6 - B_6$$
$$K_{b8} = G_8 - B_8$$
$$K_{b2} = G_2 - B_2$$
$$K_{b12} = G_{12} - B_{12}$$

Then we can use original blue value B_7 to estimate G_7 as follows

$$G_7 = B_7 + \frac{\left(K_{b6} + K_{b8} + K_{b2} + K_{b12}\right)}{4}$$

The red position is the same.

Second, we want to re-interpolate K_r and K_b planes. To estimate K_r and K_b, we estimate G minus original R/B to set up corresponding planes. Now we have three color planes to rebuild it. In Figure2.1.3 we can reset K_r as follows

$$K_{r3} = G_3 - R_3$$
$$K_{r7} = G_7 - R_7$$
$$K_{r9} = G_9 - R_9$$
$$K_{r13} = G_{13} - R_{13}$$

Similarly for K_b. In the new K_r and K_b, we can use bilinear interpolation to interpolate blank position.

$$K_{r8} = \frac{\left(K_{r3} + K_{r7} + K_{r9} + K_{r13}\right)}{4}$$

4. Experiments and Results

Experimental Environment

CPU: AMD Athlontm II X2 245 Processor 2.90 GHz

Memory: 4 GB

OS: Windows 7 Professional

Programming Language: Microsoft Visual Studio 2008 with OpenCV 2.0

Original Image1

Raw data: 10 bits/pixel

Image size: 2592*1944 pixels

Bayer pattern: *BG_bG_rR*

Bilinear: 0 votes CDb: 2 votes Our Method: 19 votes

(CD: Color Difference with bilinear interpolation)

Original Image2
Raw data: 10 bits/pixel
Image size: 2592*1944 pixels
Bayer pattern: *BG_bG_rR*

Our Method: 20 votes CDb: 0 votes Bilinear: 1 votes

Original Image3
Raw data: 10 bits/pixel
Image size: 2592*1944 pixels
Bayer pattern: *BG_bG_rR*

CDb: 4 votes Our Method: 17 votes Bilinear: 0 votes

5. Conclusion and Future Work

5.1 Conclusion

Traditional color interpolation uses Kodak sample images to calculate PSNR (Peak Signal-to-Noise Ratio). Our method designs for real raw data but Kodak sample image has no common artifact, such as false color and cross-talk noise. Thus, we use real raw data and votes to show our method

results. In real raw data, our method reduces cross-talk noise and false color successfully. But bad pixel or fixed pattern noise affect our method to determine which region is smooth or not. Noise has outlier pixel characteristic. In gradient calculation, we may determine it as an edge across this mask. If we can reduce noise effect, image after color interpolation will get higher quality than before.

5.2 Future Work

In Section 3.1, we determine pixel is an edge or not. If we can develop a function to determine whether each pixel in the mask is outlier pixel or not, it will be useful for our method to determine whether central pixel needs compensation or not. Thus we can further enhance image quality.

REFERENCES

[1] B. E. Bayer, "Color Imaging Array," US Patent #3971065, 1976.

[2] T. Chen, "Bilinear Interpolation," http://scien.stanford.edu/pages/labsite/1999/psych221/projects/99/tingchen/algodep/bilinear.html, 2011.

[3] T. Chen, "Edge Sensing Interpolation Algorithm I," http://scien.stanford.edu/pages/labsite/1999/psych221/projects/99/tingchen/algodep/edgesense.html, 2011.

[4] T. Chen, "Nearest Neighbor Replication," http://scien.stanford.edu/pages/labsite/1999/psych221/projects/99/tingchen/algodep/nbreplica.html, 2011.

[5] W. Li, P. Ogunbona, Y. Shi, and I. Kharitonenko, "CMOS Sensor Cross-Talk Compensation for Digital Cameras," *IEEE Transactions on Consumer Electronics*, Vol. 48, No. 2, pp. 292-297, 2002.

[6] R. Lukac and K. N. Plataniotis, "A Robust, Cost-Effective Post-Processor for Enhancing Demosaicked Camera Images," *Journal of Real-Time Imaging*, Vol. 11, Issue 2, pp.139–150, 2005.

[7] R. Lukac and K. N. Plataniotis, "Data Adaptive Filters for Demosaicking: A Framework," *IEEE Transactions on Consumer Electronics*, Vol. 51, No. 2, pp. 560-570, 2005.

[8] S. C. Pei, I. K. Tam., "Effective Color Interpolation in CCD Color Filter Arrays Using Signal Correlation," *IEEE Transactions on Consumer Electronics*, Vol. 13, No. 6, pp. 503-513, 2003.

[9] Wikipedia, "Bayer Filter," http://en.wikipedia.org/wiki/Bayer_filter, 2011.

[10] Wikipedia, "Color Filter Array," http://en.wikipedia.org/wiki/Bayer_filter, 2011.