Constant-Hue-Based Color Filter Array Demosaicking Sensor for Digital Still Camera Implementation

Yu-Cheng Fan, *Member, IEEE*, Yi-Feng Chiang, and Yin-Te Hsieh

***Abstract*— In this paper, we propose an edge-oriented constant-hue scheme for a color filter array demosaicking sensor to reduce costs and the complexity of the digital camera system. Most digital still cameras use color filter arrays to sample the scenery. The image information needs proper interpolation to present a complete image. This paper proposes a novel design for color image interpolation to improve the result of such processing without great hardware and processing cost. The edge-oriented scheme and constant-hue interpolation are performed to process diagonal edge information. Experimental results show that our algorithm only adds some exquisite comparison function and provides a fine image quality. It is easy to implement in hardware design by simple operation.**

***Index Terms*— Color filter array, constant-hue interpolation, edge-oriented scheme.**

I. Introduction

IMAGE sensor arrays can capture the required single image information in one shot because they adopt a specifically-designed color filter array. To compensate for the lack of information on the other two colors, a proper color interpolation method must be used to calculate reasonable values for the missing colors by means of neighboring real sampled pixel information of every pixel. In other words, the color interpolation method enhances the definition of down sampling.

Color filter arrays have been provided many types [1]. Each one must adopt its own color interpolation methods according to its type. Among them, the Bayer pattern [2], shown in Fig. 1, has been commonly used and extensively researched. Using an inappropriate color interpolation method will result in many color artifacts and produce many error estimations, especially across the edges of objects in the image. A color interpolation algorithm using the Bayer pattern has been discussed and adopted widely because of its ease, simplicity, and the maturity of the research. The goal is to obtain an image that is close to the original image in a short time, with higher performance. This is the same principle applied to photography equipment. As long as the interpolation method is proper, it can obtain a distinct image. Currently, many methods that provide good results are available [3]–[5].

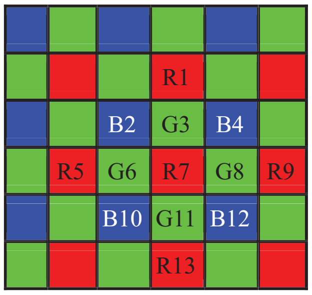

Fig. 1. Bayer pattern.

A. Spatial Domain Image Color Interpolation Algorithm

The spatial domain color interpolation technique has a fast processing speed and explicit theory concepts. Because of its simple operation, it is easy to implement in hardware.

1) Extensively Used Color Interpolation (ECI): This is the simple image color interpolation group that includes Constant-Hue Interpolation [6]. The algorithm has a very important premise, i.e. a high correlation of red, green, and blue. In an image object, the mutual color rate of the three colors is a fixed constant or changes very slowly. Thus, a mutual correlation can be used to obtain the best interpolation result. With the perfect assumption in hand, it can estimate the interpolation pixel values of red and blue in the green image plane, which has a large amount of sample information, and is quite helpful to enhance the image quality. Effective Color Interpolation [7] defines two parameters for assisting interpolation processing in every pixel, as expressed in the formulas below

$$K_R = G - R \tag{1}$$

$$K_B = G - B. \tag{2}$$

The step of the ECI algorithm also needs to obtain a complete green plane first. The chief characteristic of the method is that the interpolation of all color planes requires complete inter-color correlation. Therefore, in the first step before interpolating for $G'7$, it is necessary to obtain K_R around the green sample. After obtaining all K_R values, using a method

Manuscript received September 16, 2012; revised February 20, 2013; accepted March 15, 2013. Date of publication April 4, 2013; date of current version May 29, 2013. This work was supported by the Taiwan E-learning and Digital Archives Programs Sponsored by the National Science Council of Taiwan under Grant NSC 97-2631-H-027-002 and Grant NSC 98-2622-E-027-033-CC3. The associate editor coordinating the review of this paper and approving it for publication was Dr. Alexander Fish.

Y.-C. Fan is with the Department of Electronic Engineering and Graduate Institute of Computer and Communication Engineering, National Taipei University of Technology, Taipei 106, Taiwan (e-mail: skystarfan@ntu.edu.tw).

Y.-F. Chiang is with the Graduate Institute of Computer and Communication Engineering, National Taipei University of Technology, Taipei 106, Taiwan (e-mail: arvin_chiang@hotmail.com).

Y.-T. Hsieh is with ELITEGROUP (ECS) Technology Company, Taipei 791-8042, Taiwan (e-mail: t5418072@ntut.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSEN.2013.2256779

similar to constant-hue interpolation obtain $G'7$, and after completing the green plane image information interpolation, the blue and red interpolation data are obtained with the similar method. Regularization Approach [8] uses a novel quadratic strategy that avoids computational demanding iterations and an adaptive technique to improve the reconstruction near the edges and the discontinuities of the images [8]. Joint Color Decrosstalk and Demosaicking [9] counters the effects of channel crosstalks by adaptive least-squares inverse filtering [9]. The new technique integrates the operations of deconvolution for crosstalk removal and interpolation for color demosaicking [9].

2) Edge-Directed Color Interpolation: Unlike the above method, the algorithm judges if an edge of an object exists at a pixel that was not sampled. If so, the interpolation direction proceeds along the judged edge direction. Before actually implementing the interpolation action, in order to implement interpolation along the edge direction, the Edge-Directed Interpolation [10] creates two gradient variation values for up/down and left/right sample information of the un-sampled green pixel. It is the same as subtracting down from up to get an absolute value, and subtracting a right neighboring pixel from left to get an absolute value, and further comparing the two values. The larger gradient value indicates that the direction might cross the edge of an object, and the smaller gradient value indicates that the direction including an un-sampled pixel might be an edge of the object. Thus, the interpolation action should proceed in this direction and ignore the image information of the larger gradient. As to the red and blue, they use constant-hue interpolation.

The green plane interpolation proceeding concept of Adaptive Color Plane Interpolation [11]–[19] is similar to the Edge-Directed Interpolation. But only in this method, besides a neighboring green sample pixel value, both the gradient value and follow-up interpolation operation value have used the calculating position sample pixel and the color information of up/down and left/right, which is the same as the sample color of the operation position. Such second order use has a better color correction result for vertical or horizontal edges. Spatial Correction Interpolation [12] is an interesting edge judgment aid method. It can be created on green plane vertical and horizontal judgment of any kind of edge interpolation algorithm. Its major function is to prevent edge judgment errors in a repeated high frequency line. Gradient Edge Detection Masks and Adaptive-Projection [13] extracts more accurate gradient/edge information on mosaic images directly [13]. A novel adaptive heterogeneity-projection with proper mask size for each pixel is presented [13] too, which combines the extracted gradient/edge information and the adaptive heterogeneity-projection values, and is an interesting research issue to replace the bilinear solution with another more accurate method [13]. New Joint Demosaicing and Zooming Algorithm [14] uses adaptive heterogeneity projection masks and luminance estimation–based masks to extract edge information of each pixel in terms of the direction of variation, and the gradient forms the mosaic image directly and accurately [14].

B. Frequency Domain Image Color Interpolation Algorithm

Although an image color interpolation algorithm of frequency domain is more complex, it can usually provide outstanding image quality, especially in the higher frequency of a graph. This kind of method is more able to operate the distortion portion due to alias which is hard to cancel in spatial domain and then restrain the alias phenomenon, thereby reducing the production of artificial colors. The Alias Canceling Interpolation [15] is that since the colors red, green, and blue have similar characteristics in the high frequency region at the same position in an image, this method can use the green channel information of which the sample frequency is one-fold higher than the red or blue to help eliminate the aliasing distortion that produces by less red and blue color samples. The Color Plane Alternating Projection Interpolation [16] consists of five steps. The first step is an initial interpolation for obtaining complete red, green, and blue image color planes. Second, the green image plane is refreshed. It picks all the red sample pixels from the original observation (Rr), and collects the green interpolation value of the red sample position in the last step (Gr) to form a sample plane. Whereupon it combines the two sample planes Rr and Gr to make four sub-bands (LL, LH, HL, HH) by formula (3) where $(n_1 n_2)$ is pixel coordinate, low pass filter$h = [1\ 2\ 1]/4$, and high pass filter $h_1 = [1 - 21]/4$. S is the two sample planes Rr and Gr. It allows the Rr to replace the high frequency sub-band

$$\begin{aligned} S_{LL}(n_1, n_2) &= h_0(n_1) * [h_0(n_2) * S(n_1, n_2)] \\ S_{LH}(n_1, n_2) &= h_0(n_1) * [h_1(n_2) * S(n_1, n_2)] \\ S_{HL}(n_1, n_2) &= h_1(n_1) * [h_0(n_2) * S(n_1, n_2)] \\ S_{HH}(n_1, n_2) &= h_1(n_1) * [h_1(n_2) * S(n_1, n_2)] \qquad (3) \end{aligned}$$

$$\begin{aligned} Gr(n_1, n_2) &= \mathrm{Rec}\big(Gr_{LL}(n_1, n_2), Rr_{LH}(n_1, n_2), \\ &\qquad Rr_{HL}(n_1, n_2), Rr_{HH}(n_1, n_2)\big) \\ &\equiv g_0(n_1) * [g_0(n_2) * Gr_{LL}(n_1, n_2)] + g_0(n_1) \\ &\quad * [g_1(n_2) * Rr_{LH}(n_1, n_2)] \\ &\quad + g_1(n_1) * [g_0(n_2) * Rr_{HL}(n_1, n_2)] \\ &\quad + g_1(n_1) * [g_1(n_2) * Rr_{HH}(n_1, n_2)] \qquad (4) \end{aligned}$$

(LH, HLand HH) of the Gr and then rebuilds the Gr by the reconstruction formula (4), low pass filter $g = [-1262 - 1]/8$, and high pass filter $g_1 = [12 - 621]/8$. It adds the green sample plane Gr after reconstruction to the integrated green interpolation plane. The blue portion is processed in the same way. Third, the detailed projection divides the three integrated color planes into four sub-bands and then uses the three green high frequency sub-bands to replace the red and blue planes. Fourth, in the observation value projection, since the most accurate information is in the original observation, it must put the original observation values into the three integrated image planes. Fifth, the third and fourth steps are repeated, usually three to five times, to produce the accurate results.

Our method of spatial image color interpolation processing is a novel and convenient method to improve image processing results. Thus, it does not need such large frequency operation processing as convolution integration or large amounts of multiplication and division operations. It considers high image quality, low operation complexity, and high efficiency system demand, and therefore is more suitable for hardware implementation.

II. Constant-Hue-Based Color Filter Array Demosaicking Algorithm

Constant-Hue-Difference Interpolation [6] and Edge-Oriented Interpolation [10] are the bases of the algorithm for improvement. It considers the theoretical assumption of a uniform color rate constant and that the green sample rate of the Bayer color filter array arrangement manner is the highest. If green plane has the highest sample rate, proper interpolation can produce the best result. The remained red and blue color images can be conducted with interpolation by the processed green image information to reduce the aliasing distortion of red and blue components for obtaining higher image quality. Thus we will proceed with the two steps and correction for green image information. It detects the vertical and horizontal edge interpolation in the first step and obtains the complete green component. The second step is to use our design for judging the probably oblique edges and to correct the results of the first step.

A. Vertical and Horizontal Edges Detection Algorithm

In order to process interpolation algorithm along the object edge direction in the image, it is necessary to estimate the edge direction. We use a Second Order Laplace Filter to judge the edge direction gradient in this stage and then by which conduct interpolation [19]. The Laplace Filter includes five coefficients. It can add the correcting conditions of red or blue sample contents to the edge direction to judge gradient value. And it adds this correcting condition after determining the interpolation direction to reduce alias of the output image.

It considers the vertical or horizontal directions of Bayer sample image information and uses the green image information of a sampled red color array [19] for an example

$$\ldots R_{-2}\ G_{-1}\ R_0\ G_1\ R_2 \ldots .$$

And then the coefficients of the filter can be calculated. Since the Bayer pattern array samples the dimidiate green image signals, we can show the frequency domain [19] as

$$G_S(\omega) = \frac{1}{2}G(\omega) + \frac{1}{2}G(\omega - \pi) \tag{5}$$

where $G_S(\omega)$ represents green information of down sampling, and $G(\omega)$ and $G(\omega - \pi)$ show original green information and alias portion information. Thus, the formula of the perfect interpolation filter can delete the alias information completely [19] as

$$H_{\text{ideal}}(\omega) = 2\text{rect}\left(\frac{\omega}{\pi}\right) \tag{6}$$

The rect is the rectangle function.

Traditional Bilinear Average Interpolation uses a three-coefficient FIR filter $h = [0.5\ 1\ 0.5]$, which cannot change the average intensity of the image for $G_S(\omega)$. The frequency response trajectory can reveal the obvious difference from perfect value clearly and does not eliminate aliasing information effectively. Thus, the signal which passes through the h_0 filter is actually like the condition [19] as

$$\begin{aligned}\hat{G}(\omega) &= G_S(\omega)H_0(\omega)\\ &= \frac{1}{2}G(\omega)H_0(\omega) + \frac{1}{2}G(\omega - \pi)H_0(\omega).\end{aligned} \tag{7}$$

The image hues that are in the high correlation of high frequency can be used to add red or blue color information to reduce aliasing. We use a red sample image for an example [19]. The red sample signal in the frequency domain denotes

$$R_S(\omega) = \frac{1}{2}R(\omega) - \frac{1}{2}R(\omega - \pi). \tag{8}$$

If the red sample signal is put through another filter h_1 to interpolate and formula (7) is added [19], then we can obtain

$$\begin{aligned}\hat{G}(\omega) &= \frac{1}{2}G(\omega)H_0(\omega) + \frac{1}{2}G(\omega - \pi)H_0(\omega)\\ &\quad + \frac{1}{2}R(\omega)H_1(\omega) - \frac{1}{2}R(\omega - \pi)H_1(\omega).\end{aligned} \tag{9}$$

If the h_1 filter design is low frequency $H_1(\omega) = 0$, high frequency $H_1(\omega) = H(\omega)$, and the hues rate change slowly, the result of $G(\omega)H_1(\omega)$ is $R(\omega)H_1(\omega)$. The two aliasing distortion terms are similar for cancellation [19]. We obtain

$$\hat{G}(\omega) = \frac{1}{2}G(\omega)H_0(\omega) + \frac{1}{2}R(\omega)H_1(\omega). \tag{10}$$

As shown in Fig. 2, a proper five-coefficient FIR filter design is $h_1 = [-0.2500.50 - 0.25]$. The loss of green pixel values due to the Bayer filter array [19] can be denoted as

$$G_0 = \frac{1}{2}(G_1 + G_{-1}) + \frac{1}{4}(-R_{-2} + 2R_0 - R_2). \tag{11}$$

The above is aimed at one-dimensional image sample of a red row and column condition for deriving. The image sample blue row and column also use the same method. Furthermore, the coefficient arrangement of rows and columns is the same as in one dimension in the plane use. The only difference is that the plane processing needs to be added the interpolation direction decision.

B. Green Plane Oblique Edge Detection Correction Algorithm

If we can obtain higher quality of the green interpolation, the red and blue interpolation result will also be better. This is because the red and blue portion of the image applied in the hues constant algorithm needs to use the green plane interpolation result.

1) First Green Plane Oblique Edge Correction Algorithm: After the complete green plane of the first stage is acquired, the original un-sampled green position is continued to correct. Considering a non-green sample (red or blue sample position) pixel, we use an oblique four-neighbor different color (red or blue color) sample value for defining the two oblique gradients, left-up to right-down ΔDLR and right-up to left-down ΔDRL, and determine the smaller one as the candidate direction of oblique edge operation to prepare for the interpolation correction. In order to prevent judgment errors or detection problems due to oversensitivity at this time, we

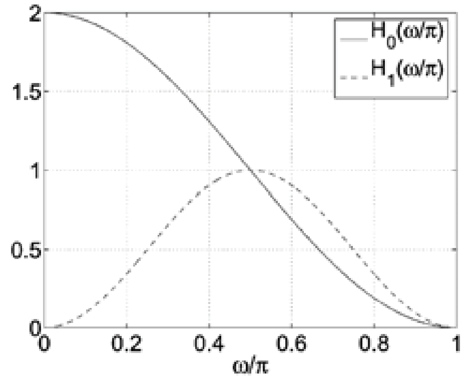

Fig. 2. Green interpolation filter [10].

define a threshold *TCD* to aid in deciding if the correction process will start. The threshold that is obtained from the experimental result is set at 220.

Comparing the obtained candidate oblique gradient and vertical & horizontal gradients (the vertical & horizontal gradients are by simple average definition), if the candidate direction gradient is smaller than the vertical gradient and the horizontal gradients surpass the threshold that we established, we will consider the green value that is from the two interpolations of the oblique direction and add the two oblique pixel values up and then average them to obtain the new green pixel value. If this pixel does not conform to the above conditions, the green pixel value keeps the first interpolation result without change. Since the interpolation of the vertical and horizontal directions has already finished in the first step, and if the pixel of the oblique edge is not defined in the oblique correction step, the value will not be changed, and the vertical and horizontal information will not be modified. After such continuous comparison, a most correct green image plane can be obtained, which will provide significantly sharper edge performance.

2) Second Green Plane Oblique Edge Correction Algorithm: For convenient and fast implementation of circuit, we provide another simple oblique edge correction. The processing method is quite similar to the previous one. It is equally considered the non-green sample pixel. As in the above description, we use the different color neighbor sample value to define the left-up to right-down and right-up to left-down two oblique gradients, ΔDLR and ΔDRL, and the same threshold gradient, *TCD*. After the same comparison step of the oblique gradient difference and the threshold, if the position has an oblique edge, the smaller oblique gradient is selected for the oblique edge correction.

The correction step below is different to the last method that uses the correcting oblique including adding the three pixel values up to get an average. It changes to add the two oblique pixel values up to average to get the last green color value, ignoring the pixel value of its position. With such a step, the number of operations in a hardware implementation will be decreased greatly. The main difference is that the dividing by two circuits is allowed a direct shift operation, eliminating a complex circuit required for division. It increases the system speed at the same time.

C. Red and Blue Image Interpolation Algorithm Design

When processing the operation of the red or blue interpolation at the blue or red pixels, due to the oblique position of the neighbor area, which has four of its own color samples, we use the neighboring green and the color itself sample values of the pixel to build up the four direction gradients that is the same as the green plane correction step. It wants to achieve an oblique edge definition that is the same as the green interpolation correction operation and proceeds to the color difference constant hues algorithm interpolation of this color by this direction. Equally, if the designated edge estimation does not achieve the established condition, the general color difference constant hues algorithm interpolation shall be used. In the second portion, considering that the image sample components are insufficient in the red or blue planes, especially in green sample pixels of the image, the desired interpolation color pixel value in the neighboring area has only two same color information. Thus in the red or blue interpolation of the image green sample pixel, we still use the traditional color difference constant hues algorithm.

III. Framework Design of the Algorithm

This paper mainly divides the circuit into four stages of pipeline for process by function. The four memory control units control the access of respective corresponding memory groups to complete the function of the algorithm.

A. First-Stage Framework Implementation

The two function circuits and their corresponding memory groups are all established in the same framework, as shown in Fig. 3. It includes the two groups of $Q_0 \sim Q_{31}$ and $R_0 \sim R_{31}$ that are 32 parallel-in-parallel-out shift registers respectively.

The image data are saved in M_{INPUT} and M0 concurrently, and the flowing data in the shift register group Q is eliminated naturally at this time, and the shift register group R does not have data. When the first image area row (block(0)~block(3)) finishes input, and M0 is full, the register group R starts to prepare to read the M0 data and the memory of the input signal for writing is also switched to M1. When the first pixel of the image area block(4) is input to Q and M1, the first pixel of the image area block(0) also enters R from M0. After 31 cycles, the stored data of Q and R will be the 8×8 interpolation block of image blocks(0), (1), (4) and (5). Thus they can be used for the green image interpolation in the next stage. After 16 cycles, the two shift register groups will finish another 8×8 interpolation block of blocks(1), (2), (5), (6). Every 16 cycles recovers an interpolation block until the process reaches the boundary of the image width. In following steps, it is repeated for switching rows of the blocks and writing in and reading out of M0 and M1, and allowing the shift register group Q and R to continue providing the image data for the interpolation. This continues to the last input of the image. An MN size picture will establish interpolation blocks of $(M/4–1) \times (N/4–1)$ times and begin the second stage process.

B. Second-Stage Framework Implementation

The second-stage framework has the initial interpolation circuit of the image green plane and the oblique edge

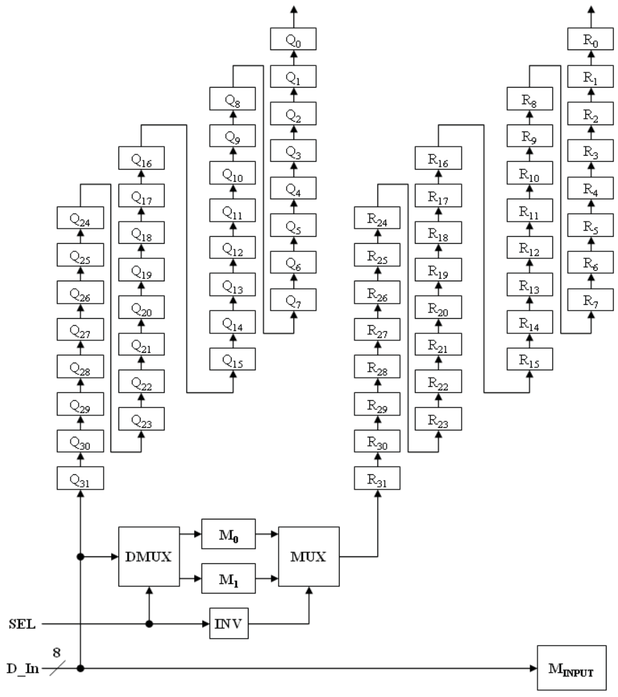

Fig. 3. Green plane interpolation circuit.

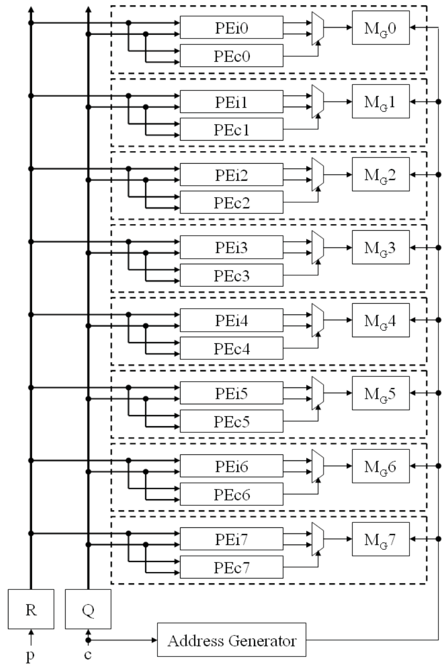

Fig. 5. Eight-cells process framework of the green interpolation.

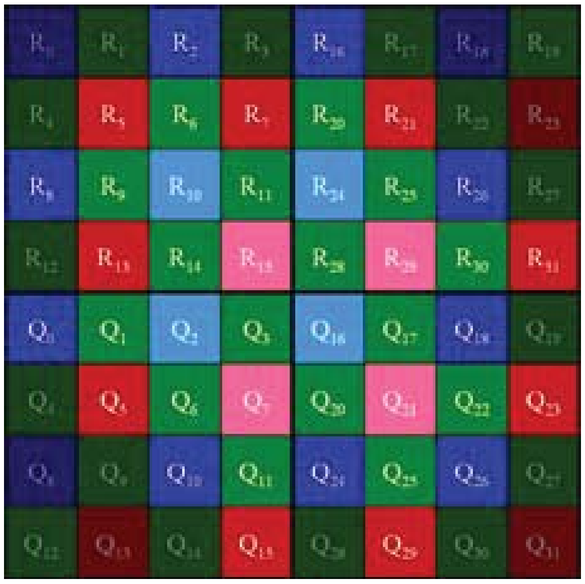

Fig. 4. 8 × 8 interpolation block is composed of the shift registers.

pre-detection circuit of the non-green sample pixel. The data are all from Q and R which preserve the pixel data of 8 × 8 interpolation block in a fixed time interval. As can be seen in Fig. 4, the data arrangement of the figure can be composed of $Q_0 \sim Q_{31}$ and $R_0 \sim R_{31}$ at the completion time of the interpolation block. The second-stage process of the system focuses on the eight non-green pixel positions (the positions of R_{10} and R_{15} and R_{24} and R_{29} and Q_2 and Q_7 and Q_{16} and Q_{21}). It is represented by the light color of which the pixel corresponds to the sample color. There are 22 unused positions which are represented by dark color of which the pixel corresponds to the sample color.

In the next fixed cycle interval and the completion time of another interpolation block, the un-changed array situation of the block is 16 cycles and the change situation is 32 cycles. And the data in the register will be pushed back. The present R_{26} will be pushed to the R_{10} position, and Q_{23} will be pushed to the Q_7 position. The second-stage circuit will proceed with the same action. Eventually, it will only have two rows and columns of the fringe and is not a block array here. Due to insufficient neighbor pixel data, it will not proceed with the algorithm providing the interpolation process, which needs to be replaced by another basic method. The first interpolation circuit of the green plane focuses on foregoing eight interpolation positions through two process elements (PE). The two PEs are direction gradient values comparison process elements (PEc0~PEc7) and direction interpolation values calculation process elements (PEi0~PEi7). Every group comprises eight PEs, which correspond to an interpolation position respectively for proceeding. The whole process framework of this portion can be seen in Fig. 5. After obtaining from Q, R shift registers in the previous-stage preparation circuit, it will respectively operate the needed position data and be parallel to input in every PE for the interpolation process.

For every PEc and PEi corresponding to same interpolation position, adding a position memory M_G can compose a cell (the portion of the dotted line in Fig. 5). Every cell processes a constant interpolation position in the interpolation block. Cell0 interpolates the green pixel of the R_{10} position. Cell1 is responsible for R_{15}, analogizing R_{24}, R_{29}, Q_2, Q_7, Q_{16}, and cell7 is responsible the position of Q_{21} in the end. Every cell necessarily total nine data include itself and up, down, left and right peripheral positions for interpolation. Since all data are registered in the shift register group, it can directly perform parallel input and fast processing.

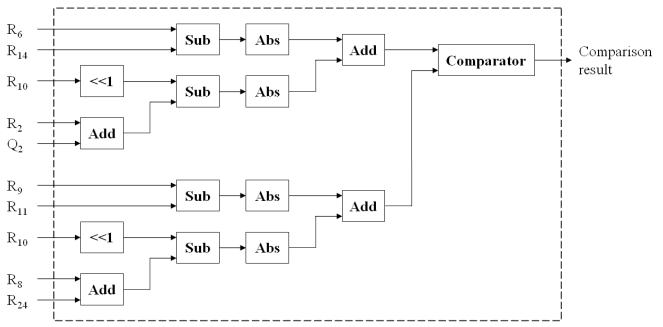

Fig. 6. Direction gradient compare process element PEc framework.

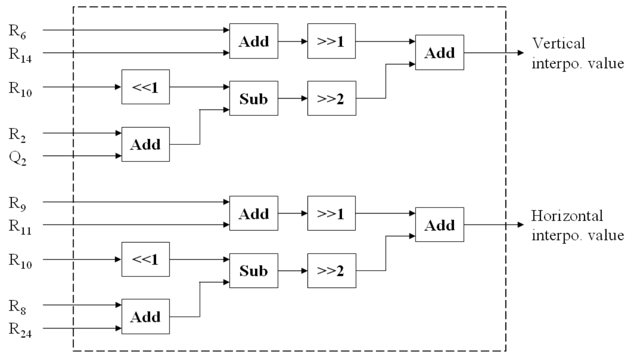

Fig. 7. Vertical and horizontal directions interpolation producing process element PEi framework.

Figs. 6 and 7 are respective PEc and PEi frameworks, and we process the cell0 corresponding to R_{10} position here for example. After one input of the nine needed data, PEi can calculate the green interpolation value of the vertical and horizontal directions, and PEc can produce vertical and horizontal gradient comparison results at the same time. Hence the result decides the input to memory M_G, which should be a vertical or horizontal interpolation value. The memory M_G saved in the green interpolation value needs a depth size, which is its input image size MN. The amount of the interpolation block is $(M/4–1) \times (N/4–1)$. Every character set size is the same as 8 bits. Data addresses saved in M_G are produced by an address generator and assign addresses from 0 to start, and the address will be added by 1 until $(M/4–1) \times (N/4–1)–1$ to end when process every interpolation area block. If it is a 16×16 image, it can build eight memories that all the depths are nine for saving interpolation data.

The main function of the oblique edge pre-detection circuit of the non-green sample pixel is the first to judge if an oblique edge exists on a green non-sample pixel when the original image is input. It only does simple detection at this time and does not accurately determine whether the accurate direction of the oblique is left or right.

The oblique edge pre-detection circuit and the green interpolation circuit proceed at the same time. Although the corresponding process position is still the same as the front description, the data that the PE selects here are the eight peripheries of the interpolation position. This is different from the nine data of the cross green interpolation. This is the same as the green interpolation of the front section. The process can be explained with the 8-cell framework in Fig. 5. The differences between them are the PE functions and memory size. Fig. 8 is a PE framework used for the oblique edge pre-detection circuit. It is the position of R_{10} as an example and also analogous to the seven others. The two comparators compare individually with that if the gradient of the oblique direction is smaller than the value that it subtracts the threshold TCD value from the vertical and horizontal direction values. If at least one of them is established, it assumes an oblique edge, and output value is 1. The result is saved synchronously in the contrast memory M_D of the pre-detection circuit and for follow-up use.

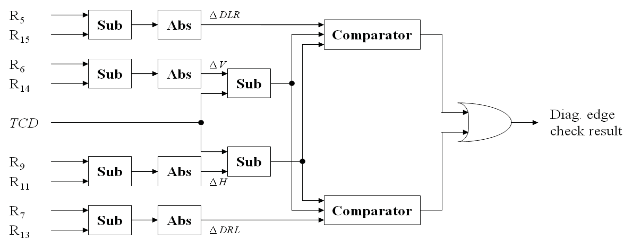

Fig. 8. Oblique edge predetection process unit PE framework.

C. Third-Stage Framework Implementation

The flow of the edge correction circuit is shown in Fig. 9. Besides M_D, the resource of the oblique edge information, it has the image information that will be used from the two memory groups. They are M_{INPUT}, the input original image memory and M_G, the green interpolation memory. First, the circuit serially inputs the oblique edge information by M_D. The Check1 circuit skips over the value 0 in M_D and sends only the memory address of the value 1 for an oblique edge. It allows the oblique green correction to follow the address and find the needed image information for correction in the corresponding positions of M_{INPUT} and M_G. Due to the format of the two memory groups, M_G and M_D are the same (just the size of the character sets is different), so it can directly correspond after getting the address that the Check1 circuit sends. The format, amount, and address of M_{INPUT} and M_D are different, but they are constant to the pixel position of the original image. So it just needs a simple constant address converter circuit and can correspond to the image interpolation position in M_{INPUT} to get the image information of the judgment correction direction.

D. Fourth-Stage Framework Implementation

The red interpolation on process position has the condition of the oblique edge. We can know that it has the oblique edge from the M_D memory group. The circuit gets the needed neighbor pixel information by M_{INPUT} and M_G to execute interpolation. The circuit then compares the sizes of the two oblique gradients and selects the gradient direction of smaller variation as the interpolation direction. The interpolation values of the left and right gradient directions are thus ready. They will be output by the selection of the interpolation direction.

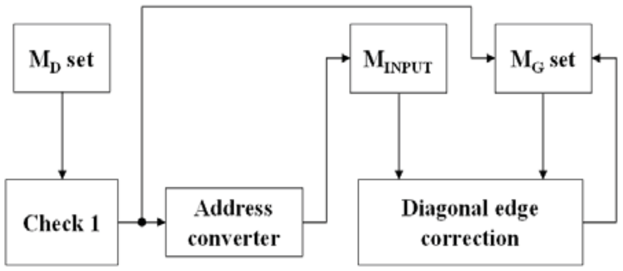

Fig. 9. Green oblique edge correction circuit framework.

Fig. 10. Twenty-four standard pictures.

The complexity of the circuit here is still not high. Only when the system presumes that it has the oblique direction edge condition and will initiate a special process. The portions of the interpolation process framework of the blue and red plane are completely the same. The corresponding address only needs to be shifted during implementation, so a duplicate illustration is unnecessary.

IV. Simulation Result and Design Flow

In order to collect statistical information and compare the experimental results, we use the 24 pictures, shown in Fig. 10, released by Kodak and frequently used in related research papers, such as [7], [12], [17], to perform the simulation of our algorithm. The original pictures obtained by [18] has a file format of PNG (Portable Network Graphics) and a picture size of 768 × 512 or 512 × 768. In addition, they have most image characteristics and are suitable for every kind of experiment and review of image processing.

A. Visual Comparison

We choose several pictures with obvious differences and focus on the detailed portions for comparison. We especially

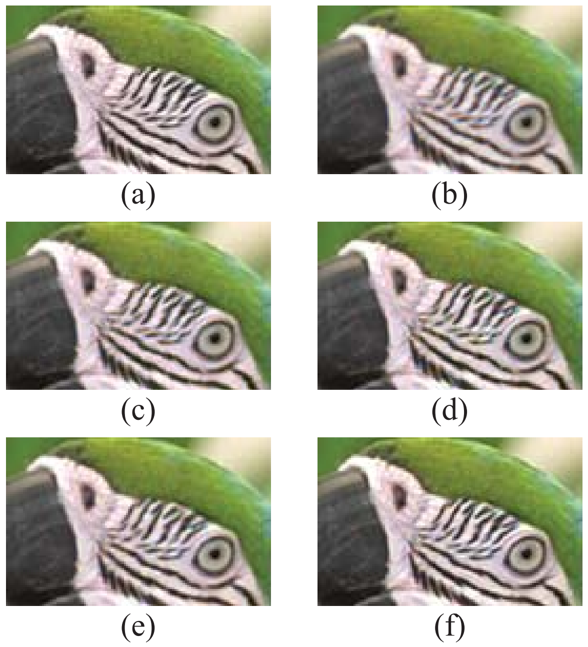

Fig. 11. Parrot (IM23) test image comparison and simulation. (a) Original picture. (b) Bilinear edge interpolation. (c) Constanthue interpolation. (d) Edgedirected interpolation. (e) Adaptive color plane interpolation. (f) Our method.

focus on the comparisons of our method with Constant-Hue Interpolation and Adaptive Color Plane Interpolation. The reason is that the main focus of our technique is to improve the green plane of Adaptive Color Plane Interpolation and enhance the red and blue planes of Constant-Hue Interpolation.

Two standard pictures with quite obvious area differences between them are used for demonstration. First, we compare the unnatural color of the stripe on the face of the parrot in the pictures in Fig. 11. It can be seen as a clear image in (a), but it is very blurred in (b). The processing results of the different methods become progressively finer through (b), (c), (d) and (e). In (f), however, it can be seen that our method can eliminate all the unnatural color. In another standard house image, the image block discussed in Fig. 12 has a great deal of object edges. The bi-linear interpolation technique in (b) and constant hue interpolation technique in (c) will produce a great deal of zipper effects on the edges, causing a blurred image. Although the edge detection interpolation technique almost clears up the zipper effect in (d), some unnatural color grains persist in the image. Our technique in (f) and edge adaptive color plane interpolation in (e) have very good results. Among the above methods, ours can limit unnatural color to the lowest occurrence. It is only some color errors in the crisscross.

B. Quantification Comparison

First, we must define the formula for comparing data and use a PSNR (Peak Signal to Noise Ratio) value, as is the norm in image processing research. It is corresponds to defining the eight-bit image as

$$\mathrm{PSNR} = 10 \times \log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right) \tag{12}$$

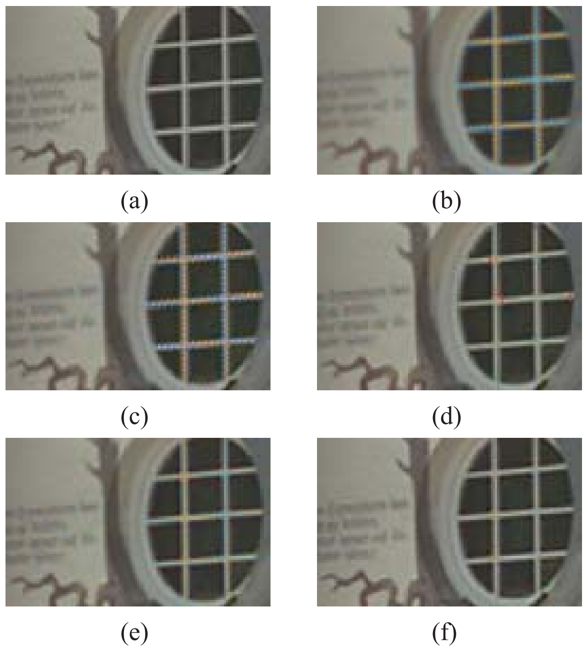

Fig. 12. House (IM24) test image comparison and simulation (a) Original picture (b) Bilinear edge interpolation (c) Constanthue interpolation (d) Edgedirected interpolation (e) Adaptive color plane interpolation (f) Our method.

where the unit is the decibel (dB), 255 is the largest value of every pixel in the eight-bit image, and MSE is Mean Square Error, defined as

$$\mathrm{MSE} = \frac{\|x - x'\|}{N} \tag{13}$$

In the formula, N denotes the total amount of the image, x is the value of the original picture, and x' denotes the value of the picture after the interpolation process. The PSNR results are compared. We list several algorithms used in standard image testing, including Bilinear Interpolation, Constant Hue Difference Interpolation [6], Color Difference Coefficient Algorithm (ECI) [7], Edge Detection Algorithm [10], and Color Plane Edge Adaptive Algorithm [11]. They are spatial domain algorithms. Enhancement Interactive Projection Algorithm (EAP) [17] is frequency domain interpolation. A complete list of comparisons is presented in Table I.

Interactive Projection Interpolation and our method can obtain superior image quality. The low frequency block has a larger portion in the image. Although our method processes the spatial domain, it is still able to obtain a high-quality result. Table II describes the computational operations comparison of different methods. The proposed method does not adopt multiplication and reduce a lot of area cost. At the same time, the proposed scheme still provide high quality image in Table I. The method provides an optimized solution between computation operations and performance.

We develop a hardware description of our algorithm. Among them includes three 8-bit data outputs, R_out, G_out, and B_out; a 1-bit output start pin, S_ST; and a 1-bit output end pin, S_End, etc. The total is 26 bits of output. Clock signal CLK and reset signal RST_N are, respectively, 1 bit; an 8-bit image information input D_In; and 1-bit input enable pin In_En. The total is 11 bits of input. The circuit specifications are 100 MHz operation frequency. The memory is thirty-two 8192 × 8 bit cells. Then a scan-chain of DFT is used to corresponding pins for the follow-up chip test. The fault coverage of the design core is 99.89%. The design contains 84 I/O pads and 4 corner pad chips, and another thirty-two 8192 × 8 bit macro cells. The follow-up post-layout verification is the last chip verification.

TABLE I
IMAGE INTERPOLATION ALGORITHM COMPARISON-OBTAINED PSNR (dB) THAT CORRESPOND TO STANDARD PICTURE

Image	Bilinear	Constant Hue	Edge Directed	Adaptive Color Plane	ECI	EAP	Proposed
IM01	29.58	30.82	30.16	37.27	35.65	**40.02**	39.27
IM02	36.26	40.56	40.61	42.27	41.28	40.2	**44.18**
IM03	37.17	37.56	37.02	42.67	43.11	43.23	**45.08**
IM04	36.53	40.87	40.82	41.99	42.16	42.15	**44.2**
IM05	29.32	30.09	30.54	35.45	36.84	39.42	**39.98**
IM06	31.05	32.10	31.82	38.51	36.98	**41.21**	40.33
IM07	36.47	37.26	37.99	42.4	42.16	43.32	**45.47**
IM08	27.40	28.21	29.52	36.24	33.09	**37.99**	37.87
IM09	35.72	36.66	37.48	43	41.37	44	**45.13**
IM10	35.37	36.31	37.48	41.77	42.07	44.3	**44.79**
IM11	32.22	33.12	33.16	39.16	38.06	40.9	**41.56**
IM12	36.82	37.84	38	43.94	42.4	44.3	**45.95**
IM13	26.49	27.35	26.37	32.51	32.64	**36.79**	35.25
IM14	32.01	36.6	36.86	38.36	37.87	37.75	**41.19**
IM15	32.01	36.54	36.17	40.88	41.11	40.09	**42.91**
IM16	34.73	35.62	35.37	41.7	40.14	**44.39**	43.08
IM17	34.54	35.48	35.35	40.72	40.35	43.65	**43.98**
IM18	30.54	31.54	30.99	36.6	36.54	**39.47**	39.11
IM19	31.74	32.77	34.35	40.76	37.23	42.58	**42.93**
IM20	34.58	35.27	35.85	40.14	40.56	**43.2**	42.09
IM21	31.54	39.06	37.15	38.38	37.57	**41.53**	40.65
IM22	33.34	38.69	38.54	39.87	38.64	39.79	**42.03**
IM23	37.98	42.14	42.54	43.49	43.84	43.38	**45.81**
IM24	29.38	36.06	33.14	35.32	35.54	37.27	**37.67**

TABLE II
COMPUTATIONAL OPERATIONS COMPARISON

Method	Computational Operations				
	Addition	Bit shift	Absolute	Comparison	Multiplication
Edge directed	4MN	2MN	1MN	0.5MN	0
Adaptive color plane	14MN	6MN	4MN	1MN	0
ECI[7]	10MN	4MN	0	0	0
EAP[17]	391.5MN	3.5MN	2MN	0.5MN	384MN
Proposed	19.5MN	4MN	6MN	3.5MN	0

V. CONCLUSION

In digital static camera systems, color filter array interpolation, besides the de-noise step, belongs to the forefront of image processing. If the design effect of the image color interpolation algorithm is good, then the following stages of image processing can use correct image information. But if the image interpolation procedure is not proper, it will directly affect the results of the following image processing steps, and thus the final image quality.

Our method has simple steps and good effects. It focuses on green plane interpolation steps, besides the second-stage correction filter proceeding of the image color information determination that this paper uses, and also can equally use new and better effects of green interpolation to establish an image green plane for use in the following green interpolation technique development. It also continually performs our oblique edge correction to obtain the best effect. The method provides an optimized solution between computation operations and performance for color filter array demosaicking sensor.

ACKNOWLEDGMENT

The authors gratefully acknowledge the Chip Implementation Center (CIC), for supplying the technology models used in IC design.

REFERENCES

[1] R. Lukac and K. N. Plataniotis, "Color filter arrays: Design and performance analysis," *IEEE Trans. Consum. Electron.*, vol. 51, no. 4, pp. 1260–1267, Nov. 2005.

[2] B. E. Bayer, "Color imaging array," U.S. Patent 3 971 065, May 12, 1976.

[3] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demosaicking: Color filter array interpolation," *IEEE Signal Process. Mag.*, vol. 22, no. 1, pp. 44–54, Jan. 2005.

[4] M. M. Hadhoud, M. Fouad, and A. A. Hamdi, "Performance study for color filter array demosaicking methods," in *Proc. Nat. Radio Sci. Conf.*, Mar. 2007, pp. 1–10.

[5] R. Ramanath, W. E. Snyder, G. L. Bilbro, and W. A. Sander, "Demosaicking methods for Bayer color arrays," *J. Electron. Imag.*, vol. 11, no. 3, pp. 306–315, Jul. 2002.

[6] 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, May 5, 1986.

[7] S.-C. Pei and I.-K. Tam, "Effective color interpolation in CCD color filter array using signal correlation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 6, pp. 503–512, Jun. 2003.

[8] K.-L. Chung, W.-J. Yang, W.-M. Yan, and C.-C. Wang, "Demosaicing of color filter array captured images using gradient edge detection mask and adaptive heterogeneity," *IEEE Trans. Image Process.*, vol. 17, no. 12, pp. 2356–2367, Dec. 2008.

[9] D. Menon and G. Calvagno, "Regularization approaches to demosaicking," *IEEE Trans. Image Process.*, vol. 18, no. 10, pp. 2209–2220, Oct. 2009.

[10] 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, Dec. 23, 1994.

[11] J. E. Adams, "Design of practical color filter array interpolation algorithms for digital cameras, part 2," in *Proc. IEEE Int. Conf. Image Process.*, vol. 1. Oct. 1998, pp. 488–492.

[12] W. Lee, S. Lee, and J. Kim, "Cost-effective color filter array demosaicing using spatial correlation," *IEEE Trans. Consum. Electron.*, vol. 52, no. 2, pp. 547–554, May 2006.

[13] K.-L. Chung, W.-J. Yang, P.-Y. Chen, W.-M. Yan, and C.-S. Fuh, "New joint demosaicing and zooming algorithm for color filter array," *IEEE Trans. Consum. Electron.*, vol. 55, no. 3, pp. 1477–1486, Aug. 2009.

[14] X. Wu and X. Zhang, "Joint color decrosstalk and demosaicking for CFA cameras," *IEEE Trans. Image Process.*, vol. 19, no. 12, pp. 3181–3189, Dec. 2010.

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

[16] B. K. Gunturk, Y. Altunbasak, and R. M. Mersereau, "Color plane interpolation using alternating projections," *IEEE Trans. Image Process.*, vol. 11, no. 9, pp. 997–1013, Sep. 2002.

[17] L. Chang and Y.-P. Tan, "Effective use of spatial and spectral correlations for color filter array demosaicking," *IEEE Trans. Consum. Electron.*, vol. 50, no. 1, pp. 355–365, Feb. 2004.

[18] *Kodak Lossless True Color Image Suite*. (1999, Nov.) [Online]. Available: http://r0k.us/graphics/kodak/

[19] Y.-C. Fan, A. Chiang, and Y.-T. Hsieh, "Edge-oriented constant-hue scheme for color filter array demosaicking," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, May 2010, pp. 1501–1504.

Yu-Cheng Fan (S'00–M'05) was born in Hsinchu, Taiwan, in 1975. He received the B.S. and M.S. degrees from National Cheng Kung University, Tainan, Taiwan, in 1997 and 1999, respectively, and the Ph.D. degree from National Taiwan University, Taipei, Taiwan, in 2005, all in electrical engineering.

He was an IC Design Engineer with the Computer and Communications Research Laboratory, Industrial Technology Research Institute, Hsinchu, from 1999 to 2000. From 2000 to 2005, he was with the Integrated System Laboratory, National Taiwan University. In 2006, he joined the Department of Electronic Engineering, National Taipei University of Technology, Taipei. Currently, he is an Associate Professor. His current research interests include consumer electronics, digital watermarking, image and video coding system, digital television broadcasting, and VLSI/SoC design.

Dr. Fan received the 13th Long-Term (Acer) Paper Awards in 1999. In 2002, he received Honors of 1st Electronics Innovative Design Award at National Taiwan University. In 2003, he received the Best Paper Award (Best Poster) of the 2003 IEEE International Conference on Consumer Electronics. He was an elected Chairman of the IEEE NTU Student Branch in 2003. In 2005, he received the IEEE Award for Outstanding Leadership and service to the IEEE NTU Student Branch. He received the Best Paper Award of the 2005 IEEE International Conference on Information Technology. He received the Honors of 2nd Taiwan Information Storage Association Ph.D. Dissertation Award in 2005. He received the Honors of 19th Long-Term (Acer) Paper Awards at the same year. In 2010, he received the Research Advancement Award at the College of Electrical Engineering and Computer Science, National Taipei University of Technology. He is a Scholastic Honor member of Phi Tau Phi.

Yi-Feng Chiang was born in Taipei, Taiwan, in 1975. He is currently pursuing the doctorate degree at the Department of Electronic Engineering, Graduate Institute of Computer and Communication Engineering, National Taipei University of Technology, Taipei.

His current research interests include image and video processing, digital static camera, and associated VLSI architectures design.

Yin-Te Hsieh was born in Taipei, Taiwan, in 1980. He received the M.S. degree from the Department of Electronic Engineering, Graduate Institute of Computer and Communication Engineering, National Taipei University of Technology, Taipei Taiwan.

He is a Senior Engineer with ELITEGROUP (ECS) Technology, a professional design company dedicated to providing computer systems. His current research interests include digital image processing and static camera.