TRAINING-BASED DEMOSAICING
Hasib Siddiqui and Hau Hwang
Qualcomm Incorporated, San Diego, CA 92121, USA

ABSTRACT
Typical digital cameras use a single-chip image sensor covered with a mosaic of red, green, and blue color filters for capturing color information. At each pixel location, only one of the three color values is known. The interpolation of the two missing color values at each pixel in a color filter array image (CFA) is called demosaicing.

In this paper, we propose a novel training-based approach for computing the missing green pixels in a CFA. The algorithm works by extracting a multi-dimensional feature vector comprising derivatives of various orders computed in a spatial neighborhood of the pixel being interpolated. Using a statistical machine learning framework, the feature vector is then used to predict the optimal interpolation direction for estimating the missing green pixel. The parameters of the statistical model are learned in an offline training procedure using example training images. Once the green channel has been estimated, the red and blue pixels are estimated using bilinear interpolation of the difference (chrominance) channels.

Both subjective and objective evaluations show that the proposed demosaic algorithm yields a high output image quality. The algorithm is computationally and memory efficient, and its sequential architecture makes it easy to implement in an imaging system.

Index Terms—Color filter array, Bayer mosaic, interpolation, demosaic, bilateral filter

1. INTRODUCTION
A digital color image typically comprises three color samples, namely red (R), green (G), and blue (B), at each pixel location. However, using three separate color sensors for measuring the three R, G, and B color values at each pixel in a digital camera is expensive. Thus, most digital cameras employ a single-chip image sensor, where each pixel in the image sensor is covered with an R, G, or B color filter for capturing the color information. The mosaic of color filters covering the pixels in a single-chip image sensor is referred to as the color filter array (CFA). The most commonly used CFA is the Bayer mosaic formed by replication of two-by-two sub-mosaics, with each sub-mosaic containing two green, one blue, and one red filter. The process of reconstructing a complete RGB color image from the CFA samples captured by a digital camera is called demosaicing.

Several demosaicing techniques, with varying degrees of complexity and image-reconstruction quality, have been proposed in the literature. One important observation commonly used in demosaicing algorithms is that there exists a high degree of correlation among the R, G, and B components of a color image. Specifically, in the high frequency region of the Fourier spectrum, the three color channels are very similar. Consequently, differences (R − G or B − G) and ratios (R/G or B/G) of the original RGB color components, referred to respectively as chrominance and hue, show a rather gradual spatial variation and can be considered constant within an object in an image. The low-pass nature of the chrominance and hue channels makes their interpolation considerably simpler. Some important contributions in demosaicing based on the aforementioned constant-chrominance or constant-hue assumptions include [1, 3, 4].

The block diagram of a typical demosaicing algorithm based on the constant-chrominance assumption is shown in Fig. 1. The method begins with the interpolation of the missing G pixels. After the G channel has been completely interpolated, sparse chrominance channels, (R − G) and (B − G), are formed. The low-pass chrominance channels in our work are interpolated using bilinear interpolation. Finally, the interpolated G and chrominance channels are added to determine the missing R and B pixel values.

Many demosaicing algorithms [1, 3, 4] exploit edge directionality during estimation of missing G pixels. Interpolation along an object boundary is generally preferable to interpolation across an object boundary. Adams et al. [1] use the absolute sums of first- and second-order directional derivatives at a pixel to determine the interpolation direction. Hirakawa's algorithm [3] works by producing two sets of color interpolated images: a horizontally-interpolated image and a vertically-interpolated image. A homogeneity metric, which works by transforming the image data to CIE − L* a* b* color space, is then used to determine the extent of color artifacts in each of the interpolated images. The optimal interpolation direction at a pixel is selected as the direction that results in lower color artifacts in the spatial vicinity of the current pixel. Hirakawa's algorithm has been demonstrated to outperform many existing demosaic algorithms. However, the need to compute two separate color images and to compare color artifacts therein for final pixel estimation requires extensive memory, which makes the algorithm unfriendly for implementation in a low-cost imaging system.

In this paper, we propose a novel training-based approach for CFA interpolation. For estimating the missing G pixels, the algorithm works by extracting a d-dimensional spatial feature vector, \( f_s \), which comprises derivatives of various orders computed at pixels in a 3 × 3 spatial neighborhood of pixel \( s = (s_1, s_2) \). The use of a high-dimensional feature allows us to capture structural information in the local image region, which is then used to predict optimal direction for CFA interpolation. The relationship between spatial feature vector, \( f_s \), and interpolation direction, \( y_s \in \{-1, +1\} \), is learned in an offline statistical framework. In our setup, \( y_s = +1 \) indicates that optimal interpolation direction is horizontal, while \( y_s = -1 \) indicates optimal interpolation direction is vertical. The paper also describes generation of training data \( \{(y_s, f_s)\}_{s=1}^{S} \) for optimizing parameters of the statistical model. The interpolation directions estimated using our machine-learning framework provide better suppression of zipper artifacts than some well-known demosaic algorithms. Moreover, the new algorithm yields a high image quality using computation-as well as memory-efficient image processing, which makes the algorithm attractive for hardware implementation.

The statistical classification framework in our algorithm is based on the discrete adaboost algorithm [2]. The reason for selecting adaboost statistical model is its computational simplicity and its success demonstrated in a myriad of recent data classification problems.
2.1. Interpolation of Green Channel

The proposed algorithm works by first interpolating the green channel. The block diagram of the algorithm for interpolating the green channel is shown in Fig. 3. First, a spatial feature vector \( f_s \) is extracted at pixel \( s \) in the Bayer mosaic. The spatial feature vector is then input to an adaboost-based classifier which outputs a parameter \( \beta \) characterizing the strength and orientation of the edge in the local window. Finally, a directional interpolation filter is used for estimating the missing pixel value, \( G_s \).

2.1.1. Feature Vector

For local window classification, we use a 19-dimensional feature vector, which comprises differences of absolute values of first-, second-, and third-order directional derivatives at pixels in a \( 3 \times 3 \) spatial neighborhood of \( s \). Assuming the first three directional derivatives of \( X \) in the \( x \)-direction are denoted by \( \nabla_xX \), \( \nabla_{xx}X \), and \( \nabla_{xxx}X \) respectively, the feature vector \( f_s \) at pixel \( s \) can be expressed as

\[
f_s = \begin{pmatrix}
\|\nabla_xX\|_{s+(-1,-1)} - \|\nabla_yX\|_{s+(-1,-1)} \\
\|\nabla_{xx}X\|_{s+(-1,-1)} - \|\nabla_{yy}X\|_{s+(-1,-1)} \\
\|\nabla_{xxx}X\|_{s+(-1,0)} - \|\nabla_{yy}X\|_{s+(-1,1)} \\
\|\nabla_{xxx}X\|_{s+(1,0)} - \|\nabla_{yy}X\|_{s+(1,1)} \\
\|\nabla_{xxx}X\|_{s+(-1,1)} - \|\nabla_{yy}X\|_{s+(-1,0)} \\
\|\nabla_{xxx}X\|_{s+(1,1)} - \|\nabla_{yy}X\|_{s+(1,0)} \\
\|\nabla_{xxx}X\|_{s+(0,-1)} - \|\nabla_{yy}X\|_{s+(0,1)} \\
\|\nabla_{xxx}X\|_{s+(1,-1)} - \|\nabla_{yy}X\|_{s+(1,0)} \\
\|\nabla_{xxx}X\|_{s+(-1,0)} - \|\nabla_{yy}X\|_{s+(-1,1)}
\end{pmatrix}
\]

(1)

where \( s + r \) denotes a pixel location in a \( 3 \times 3 \) neighborhood of \( s \). We use the following 1-D kernels for estimation of, respectively, the first-, second-, and third-order directional derivatives in the neighborhood of pixel \( s \): \( p = (-1,0,1)^T \), \( q = (-1,0,2,0,-1)^T \), and \( r = (-1,0,2,0,-2,0,1)^T \).

From (1), we notice that while the first two derivatives are computed at \( s \) as well as at each of its 8 neighbors \( s + r \), the third-order
derivative is computed only at $s$. This is done to ensure that the computation of $f_s$ does not involve pixels outside a $7 \times 7$ local window centered at $s$.

2.1.2. Classifier Decision Rule

AdaBoost is an ensemble classifier that uses a group of $K$ weak classifiers to reach a decision. In our setting, each weak classifier, $\{h_k(f_s)\}_{k=1}^{K}$, compares a component of the multi-dimensional feature vector, $f_s$, against a threshold $T_k$ and decides the class label $y_s \in \{-1, 1\}$ for pixel $s$ according to the following rule:

$$h_k(f_s) = \text{sign}[b_k(f_s, l_k) - T_k],$$

where $b_k \in \{-1, 1\}$, $f_s$ denotes the $i^{th}$ component of $f_s$, and $\text{sign}(x) = \begin{cases} +1 & \text{if } x > 0, \\ -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0. \end{cases}$

The decisions of the weak learners are weighted according to their relevance weights, $w_k$, and summed.

$$\sigma_s = \sum_{k=1}^{K} w_k h_k(f_s).$$

Finally, the classifier decision rule, $\beta_s$, is computed as:

$$\beta_s = \frac{1}{2} \max( (\sigma_s + 1), 2),$$

where $\beta_s \in [0, 1]$. The value of $\beta_s$ signifies the presence of a predominantly vertical edge at $s$, i.e. $y_s = -1$, when $\beta_s < 0.5$ and horizontal edge at $s$, i.e. $y_s = +1$, when $\beta_s > 0.5$. The classification parameters $w_k$, $b_k$, $l_k$, and $T_k$ are computed in an offline training process using the boosting algorithm in Fig. 2. The generation of training data for adaboost parameter learning is described in the following sub-section.

2.1.3. Generation of Training Data

The training data for optimizing classification parameters $w_k$, $b_k$, $l_k$, and $T_k$ comprises examples of vectors $(y_s, f_s)$, where $y_s \in \{-1, +1\}$ is a binary variable indicating optimal direction for interpolation when the extracted feature vector is $f_s$.

Figure 4 illustrates the procedure for generating sample training vectors, $(y_s, f_s)$. The input training image is a high-quality, three-channel, RGB color image. The three-channel input image is used to simulate a Bayer CFA mosaic. The sparse green channel in the Bayer CFA mosaic is shown in the figure. For extracting training vectors, we select a subset $S$ of pixel locations in the training images. Each pixel $s \in S$ represents a location where the green pixel value, $G_s$, is unknown in the Bayer CFA image. The feature vector $f_s$ is extracted using pixels in a spatial neighborhood of $s$, as described in (1). To determine the class label, $y_s$, the value of $G_s$ is estimated using vertical and horizontal interpolations: $G^H_s = (X_{s+0.1})/2$ and $G^V_s = (X_{s+0.1})/2$.

The class label $y_s$ is then selected as:

$$y_s = \begin{cases} +1 & \text{if } |G^H_s - G_s| > |G^V_s - G_s|, \\ -1 & \text{if } |G^H_s - G_s| < |G^V_s - G_s|. \end{cases}$$

Notice that in the training process, we have access to full RGB color images, and hence the pixel values $G_s$ are known.

2.1.4. Interpolation Filter

The high frequency components in the $R$, $G$, and $B$ channels of an image are highly correlated. Thus, while estimating pixels in one

---

**Fig. 3.** Block diagram of proposed scheme used for interpolation of G channel.

**Fig. 4.** Generation of training data for optimizing classifier parameters using Adaboost.
of the three channels, it is customary to extract high-frequency information from the remaining two channels and use it for improving the estimation of the channel being interpolated [1,3,4].

For estimating $G_3$, we use a 1-D horizontal and a 1-D vertical interpolation kernel, each of length 7. The even samples of each kernel constitute a band-pass filter, while the odd samples constitute a low-pass filter. The value of the missing green pixel $G_3$ is estimated using the following convex average of the outputs of the two 1-D filters.

$$
\hat{G}_3 = (1 - \beta_s)G_3^V + \beta_sG_3^H,
$$

where $\beta_s$ denotes the classifier decision rule as in (4) and

$$
\hat{G}_3^V = (1 - a)X_{s_1, s_2} + a(X_{s_1-1, s_2} + X_{s_1+1, s_2}) + \frac{a - 1}{2}(X_{s_1-2, s_2} + X_{s_1+2, s_2}) + \frac{1 - 2a}{2}(X_{s_1-3, s_2} + X_{s_1+3, s_2}),
$$

where $a$ is empirically selected as 0.4. The value $\hat{G}_3^H$ is estimated using an interpolation kernel with same coefficients as in (6), but oriented in the horizontal direction.

### 3. EXPERIMENTAL RESULTS

A set of 6, 768 x 768 (or 512 x 768) Kodak color images is used to train the statistical model parameters of the proposed algorithm. Another set of 18 different Kodak color images is used to evaluate the performance of the algorithm. The color images are first used to simulate Bayer CFA mosaics, which are then interpolated using different demosaic algorithms.

For performance comparison, we use four different algorithms for interpolating the $G$ channel: (1) Bilinear interpolation; (2) Edge-directed interpolation [1]; (3) Homogeneity-directed interpolation [3]; and (4) Proposed training-based demosaic (TBD) algorithm.

Fig. 5 shows the demosaic results. Fig. 5(a) shows a zoomed-in view of a portion of an original Kodak image. Fig. 5(b) shows the demosaiced image using bilinear interpolation. We see that bilinear interpolation results in serious zipper and aliasing artifacts. The artifacts are greatly reduced in the demosaiced images of Figs. 5(c) and (d) which are generated using edge-directed [1] and homogeneity-directed [3] demosaicing algorithms. The demosaiced image with our TBD algorithm is shown in Fig. 5(e). We notice that the output of the proposed algorithm displays substantial improvements in image quality over the other considered demosaic solutions.

In Table 1 we compare the average performance of the various demosaic algorithms across our test set of 24 Kodak images using two different objective measures of image quality: peak-signal-to-noise-ratio (PSNR) and $YC_\alpha C_\beta / L^{a^*b^*} - \Delta E$ error [5].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR (dB)</td>
<td>37.25</td>
<td>37.79</td>
<td>37.81</td>
<td>38.8</td>
</tr>
<tr>
<td>$YC_\alpha C_\beta / L^{a^<em>b^</em>}$</td>
<td>2.05</td>
<td>1.72</td>
<td>1.72</td>
<td>1.65</td>
</tr>
</tbody>
</table>

#### Table 1. Quantitative performance comparison of demosaic algorithms.

# 4. CONCLUSIONS

The proposed training-based demosaic algorithm provides a new perspective for interpolation of color filter array image data. The algorithm makes use of training data vectors, comprising pairs of local feature vectors and their corresponding ground-truth class labels indicating optimal interpolation directions, for optimizing parameters of the statistical model. The paper also discusses a procedure for generating training data for parameter learning. The computationally-intensive process of learning algorithm parameters is performed offline. The online computations required for interpolating a color filter array test image are simple. A sequential architecture and efficient use of memory make the algorithm attractive for implementation in an imaging system. Using simulated Bayer CFA mosaics, we demonstrated that the proposed algorithm yields superior image quality compared to some well-known existing demosaic techniques.

# 5. REFERENCES


