Edge Strength based Fuzzification of Color Demosaicking Algorithms

Major Project submitted in partial fulfilment of the requirements for the award of degree of

Master of Technology

In

Information Systems

Submitted By:

Deepak Aneja

(2K11/ISY/04)

Under the Guidance of:

Ms. Seba Susan

(Assistant Professor)

(Department of Information Technology)

Department of Information Technology

Delhi Technological University

(2011-2013)

CERTIFICATE

This is to certify that **Deepak Aneja (2K11/ISY/04)** has carried out the major project titled **"Edge Strength based Fuzzification of Color Demosaicking Algorithms"** as a partial requirement for the award of Master of Technology degree in Information Systems by **Delhi Technological University**.

The major project is a bonafide piece of work carried out and completed under my supervision and guidance during the academic session **2011-2013**. The matter contained in this report has not been submitted elsewhere for the award of any other degree.

(Project Guide)

Ms. Seba Susan

Assistant Professor

Department of Information Technology

Delhi Technological University

Bawana Road, Delhi-110042

Abstract

Most digital camera use only a single photo sensor overlaid with a color filter array (CFA) to capture image data. This allows only one of the required color samples to be available at each pixel location and other two color components need to be interpolated. This process of reconstructing the full color image from the incomplete color components at each pixel output from the image sensor is known as demosaicking or color filter array interpolation. Over the past years, many demosaicking algorithms have been introduced in order to optimize the subjective and objective interpolation quality, it becomes difficult to implement them in digital cameras due to their limited computing capacity, available processing time, and hardware size. An edge strength based fuzzification of demosaicking algorithms is proposed in this thesis in which edge strength information from the raw image data is fuzzified and effectively utilized to improve the interpolation quality of current demosaicking algorithms. We have used five image datasets including the kodak lossless true color image suit to test our approach over three performance measures, MSE, PSNR and computation complexity. Experimental results confirm the effectiveness of our approach when compared to other algorithms.

Keywords: Color Filter Array(CFA) Interpolation, Demosaicking, Color Correlation, Digital Cameras, Fuzzy Membership, Edge Strength Filter.

ACKNOWLEDGEMENT

I express my gratitude to my major project guide **Ms. Seba Susan, Assistant Professor, Department of Information Technology** for the valuable support and guidance she provided in making this major project. It is my pleasure to record my sincere thanks to my respected guide for her constructive criticism and insight without which the project would not have shaped as it has.

I humbly extend my words of gratitude to other faculty members and my friends for providing their valuable help and time whenever it was required.

Deepak Aneja

Roll No. 2K11/ISY/04

M.Tech. (Information Systems)

E-mail: deepakaneja.cs@gmail.com

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Defining Demosaicking and its Need

A digital image is composed of red (R), green (G) and blue (B) color samples at each pixel position. In order to construct a true color image a digital camera would require three separate photo sensors, each responsible for capturing color information lying in the wavelength range of red, blue and green color respectively. In a three chip digital camera, light entering the camera is split and projected onto each color sensor. These sensors have to be registered precisely because outputs from these three sensors are concatenated to construct a true color image. These additional requirements and the cost of three different color sensors make the system costly. Thus, most digital camera use only a single sensor covered with a color filter array (CFA) which allows only one color to be measured at each pixel. This means the other two color values must be estimated at each pixel. This process of estimation is known as demosaicking or color filter array interpolation.

1.2 Bayer CFA

Although many different CFA pattern have been proposed. The most common CFA pattern is Bayer CFA [1] which is 50% Green, 25% Red and 25% Blue as shown in figure 1.1. The Green color is sampled at a rate twice that of Blue and Green color because the human visual system is more sensitive in the medium wavelengths, corresponding to the green color. Other patterns are also used, e.g., the Nikon Coolpix 990 uses a cyan, magenta, yellow and green (CMYG) grid where each of the colors is sampled at the same rate.

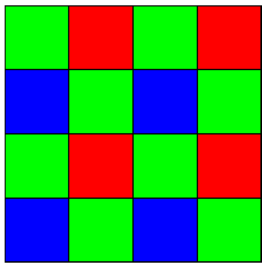


Figure 1.1: Bayer CFA

1.3 Motivation

The output from color filter array is the incomplete color samples (raw image data) which need to be processed using a demosaicking algorithm to construct the complete color image. Demosaicking method makes exploits the color correlation between pixels within an image to estimate the missing color component. Spatial correlation is defined as the tandency of pixels to assume similar color values within a small homogenous region of an image and spectral correlation is the dependency between the pixel values of different color planes in a small image region. The basic assumption is that color ratio or color difference is constant within a small homogenous region. This assumption tends to fail across edges, hence many demosaicking algorithms utilize the edge information adaptively. A good demosaicking algorithm should have the following traits.

- Avoidance of introduction of false color artifacts, such as chromatic aliases, zippering (abrupt unnatural changes of intensity over a number of neighbouring pixels).
- Maximum preservation of image resolution.
- Low computational complexity for fast processing.
- Amenability to analysis for accurate noise reduction.

1.4 Demosaicking: A literature survey

Demosaicking is a major step in image processing of digital cameras and has been an area of research in both academics and industry. A large number of demosaicking approaches have been proposed in past years. Some methods employ simple numerical formulas [2] such as nearest neighbour, median filtering [64], bilinear and bicubic interpolation. Bicubic interpolation requires high order computation [3] than bilinear but can product high-frequency components of images. A large circuit size is required for high-order computation which is not feasible for a small size digital imaging device. Therefore a small circuit is employed in DSC [4]-[6] so that low-order computation can be performed with a technique to improve the interpolation accuracy by using an exponential calculation [4]-[7]. These spatially invariant interpolation methods treat color channels separately and interpolate missing pixels in each channel. This approach works well in uniform areas but it produces color artifacts in areas with textures and edges.

In order to obtain better demosaicking performance, correlation between color channels is exploited. Constant color difference rule or constant color ratio rule [7], [8], [9], [10], [66], [71] exploits the spectral correlation which assumes that color ratio or color difference is constant in a homogenous region. This assumption fails across edges, therefore many demosaicking algorithms use the edge information adaptively during color interpolation. Apart from this some methods perform statistical analysis of images, for example multivariate analysis, Bayesian estimation, learning by training sample etc. [11]-[14]. Applications of the pattern matching [15] have also been employed. Green channel suffers less from aliasing because in Bayer CFA pattern green channel samples are twice that of red and blue ones. Therefore green channel is the starting point of the CFA interpolation process. Glotzbach et al. in [16], focuses on improving red and blue channel by adding into them the high frequency components obtained from green channel. Gunturk et al. in [17] used on

alternating projections scheme based on inter-channel color correlation in high frequency subbands to improve the red and blue channels successively. Many observations regarding color channel frequencies made in a method [18] and suggest that instead of filtering the CFA image as individual color channels, it should be filtered as a whole preserve high frequency information. The method uses a fixed 5×5 filter for green channel interpolation and an adaptive filter for blue and red channel interpolation. The fully interpolated green channel is then used to interpolate the chrominance information.

Various edge direction based decision rules for the green channel interpolation [9], [10], [19], [20], [62], [65], [69] has been proposed early. The method in [9] uses derivatives of red and blue samples in initial green channel interpolation. An approach using variance of color differences as a decision rule is been proposed by authors of [21]. Zhang et al. [22], tried to improve the interpolation performance of the original method [9] by making a soft decision. In this method, color differences along horizontal and vertical directions are treated as noise and they are combined in optimal manner using the linear minimum mean square estimation (LMMSE) framework. This approach is further improved by Paliy et al. [23] by introducing scale adaptive filtering based on linear polynomial interpolation (LPA). Hirakawa et al. [24] proposed performing interpolation in both horizontal and vertical directions by comparing local homogeneity of horizontal and vertical interpolation resuls and Menon et al. [25] used color gradients over a local window to make the direction decision.

A subset of methods involves image restoration techniques [66]. Techniques such as pseudo-inverse filter [26], a super resolution technique [27]-[29], a technique based on the interpolation of projection onto convex sets (POCS) [17], [30], and use of Discrete Cosine Transform (DCT) and Wavelet conversion [33]-[34]. Since these techniques requires a significant increase in hardware size, they are not suitable for single chip digital cameras.

Chapter 2

Classification of Demosaicking Methods

Many demosaicking methods have been proposed over the past years. This field of research and development is getting more and more attention because of the emerging market of electronic consumer devices. These demosaicking methods exploit the spatial and spectral color correlation within an image to interpolate the missing color values at each pixel position. The reconstructed image is generally accurate in constant color difference areas, but has a loss of resolution (detail and sharpness) and has edge artifacts. Demosaicking methods can be broadly categorized into non-adaptive algorithms and adaptive algorithms, as described below.

2.1 Non-adaptive algorithms: In non-adaptive demosaicking algorithms, a fixed pattern of computation is performed on every pixel location in the raw image data (mosaic pattern) in order to estimate the two missing color components. These types of algorithms are easy to implement with low cost in terms of computational requirements.

2.2 Adaptive algorithms: In adaptive demosaicking algorithms, intelligent processing is performed on every pixel location based on the characteristics of the image in order to estimate the missing color components. These types of algorithms yield better results in terms of quality as compared with the non-adaptive algorithms. However, effective algorithms in this category are computationally more complex.

2.3 Algorithms exploiting the correlation between color planes in an image: In this set of algorithms, we have explained two algorithms which utilizes the correlations between different color planes in an image to estimate the missing color components. Both algorithms are adaptive in nature and makes use of the edge information.

We review some algorithms from all three categories in order to introduce some flavour and characteristics of the demosaicking methods proposed in the literature. In this report, we have only reviewed some basic methods that we have compared with our proposed algorithm.

2.1 Non-adaptive algorithms

2.1.1 Nearest Neighbour Replication: In this simple color interpolation method [35], [36], each missing color component in a pixel replicates the value of the nearest pixel of the same color component in the input image. The nearest neighbour can be any one of the upper, lower, left and right pixel. An example is illustrated below in figure 2.1 for a 3x3 block in green plane. Here we assume the left neighbouring pixel value is used to fill the missing ones.

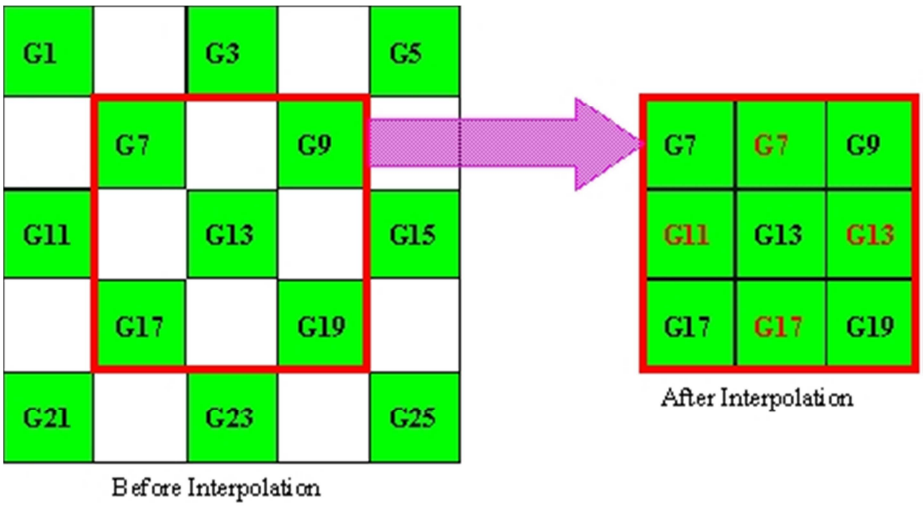


Figure 1.1: Illustration of Nearest Neighbour Replication

As discussed by James E. Adams [35], the only advantage of this approach is that computational requirement is very less and well suited for applications where speed is very crucial. However, the significant color errors make it unsuitable for still imaging system, such as high-resolution digital cameras.

2.1.2 Bilinear Interpolation: Instead of replicating the nearest neighbours, bilinear interpolation [35] estimate the missing color component by taking the linear average of the adjacent pixels with same color component. For example, the pixel B8 at location (2, 3) in figure 2.2 contains blue component only. Hence the missing green component at red/blue pixels is estimated by taking average of the left, right, top and bottom green pixel values. Interpolation of a red/blue pixel at a green position is performed by taking the average of two adjacent pixel values in corresponding color. The missing red/blue component at corresponding blue/red pixels can be estimated as linear average of the four diagonally adjacent corner neighbours containing red/blue pixels. This is illustrated by an example below using figure 2.2 for a 5 x 5 block.

Interpolation of green pixel at red or blue pixel position: Here we are estimating green value at blue pixel B8. Similarly green value can be estimated at red pixels.

$$g8 = \frac{G3 + G7 + G9 + G13}{4} \tag{2.1}$$

Interpolation of red/blue pixel at green position: Here we are estimating blue and red value at green pixel G7. Similarly red and blue value can be estimated at pixel location G13.

$$b7 = \frac{B6 + B8}{2};\ \ r7 = \frac{R2 + R12}{2} \tag{2.2}$$

Interpolation of red/blue pixel at blue/red position: Here we are estimating red and blue value at blue pixel B8 and red pixel R12 respectively.

$$r8 = \frac{R2 + R4 + R12 + R14}{4};\ \ b12 = \frac{B6 + B8 + B16 + B18}{4} \tag{2.3}$$

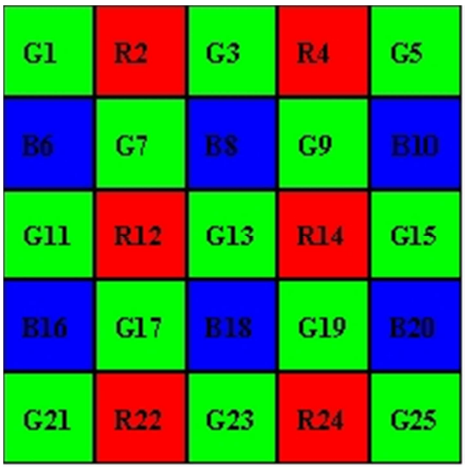


Figure 2.2: 5×5 Bayer CFA pattern (GRBG)

This method is very simple and can be easily implemented. However, experimental results show that zipper effect is introduced in the neighbourhood of the interpolated pixels in the interpolated full color image. This artifact may be acceptable in a video stream because the artifact may not be visible by the human eye due to effect of motion blur between video frames, but these artifacts are not acceptable for still images.

2.1.3 Smooth Hue Transition Interpolation: The key problem of the color artifacts in bilinear is that the hue values of adjacent pixels change abruptly across edges. The Bayer CFA pattern can be considered as combination of a luminance channel (green pixels) and two chrominance channels (red and blue pixels). The smooth hue transition interpolation method [37]-[42], [63] interpolates these channels independently. The luminance or green component can first be interpolated at red and blue pixel locations using bilinear interpolation as discussed before. Then the chrominance channel is interpolated by imposing a smooth transition in hue value from pixel to pixel. In order to do so, it defines "hue value" for blue as B/G, and "hue value" for red as R/G. For interpolation of the missing blue pixel values $b_{m,n}$, in pixel location (m, n) in the Bayer pattern, the following three different cases may arise.

Case 1: the pixel at location (m, n) contains green color component only and the adjacent left and right pixel locations contain blue color component only. For example, the pixel G7 at location (2, 2) in figure 2.2 contains green component only and its adjacent pixels in left and right contain only blue component. The blue component at pixel location (m, n) can be estimated as follows:

$$b_{m,n} = G_{m,n} * \left(\frac{B_{m,n-1}}{g_{m,n-1}} + \frac{B_{m,n+1}}{g_{m,n+1}}\right)/2 \tag{2.4}$$

Case 2: the pixel at location (m, n) contains green color component only and the adjacent top and bottom pixel locations contain blue color component only. The pixel G13 at location (3, 3) in figure 2.2 is such an example. The blue component at location (m, n) can be estimated as follows:

$$b_{m,n} = G_{m,n} * \left(\frac{B_{m-1,n}}{g_{m-1,n}} + \frac{B_{m+1,n}}{g_{m+1,n}}\right)/2 \tag{2.5}$$

Case 3: the pixel at location (m, n) contains red color component only and the four diagonally neighbouring corner pixels contain blue color component only. For example, the pixel R12 at location (3, 2) in figure 2.2 contains red color component only. The blue component at location (m, n) can be estimated as follows:

$$b_{m,n} = G_{m,n} * \left(\frac{B_{m-1,n-1}}{g_{m-1,n-1}} + \frac{B_{m-1,n+1}}{g_{m-1,n+1}} + \frac{B_{m+1,n-1}}{g_{m+1,n-1}} + \frac{B_{m+1,n+1}}{g_{m+1,n+1}}\right)/4 \tag{2.6}$$

The interpolation of missing red pixel values can be computed similarly at blue and green pixels.

The “hue value” changes depending on where the interpolation step happens in the image processing chain. For example, if the pixel value is transformed into logarithmic space from linear space before interpolation, then the “hue value” can be defined as B-G or R-G instead

of B/G or R/G. This is coming from the fact that log(X/Y) = log(X) – log(Y) = X' – Y'. Here X’ and Y’ are the logarithmic values of X and Y respectively. This helps reduce computational complexity for implementation because all the division for calculating the hue value is replaced by subtraction.

2.2 Adaptive algorithms

2.2.1 Pattern Matching based Demosaicking Algorithm: In the Bayer pattern, a blue or red pixel has four neighbouring green pixels. Wu, et al [43] proposed simple pattern matching algorithm for interpolating the missing color components based on the pixel contexts. The algorithm defines a green pattern for the pixel at location (m, n) containing a non-green color component as a four-dimensional integer-valued vector:

$$g(m,n) = \left(G_{m-1,n}, G_{m+1,n}, G_{m,n-1}, G_{m,n+1} \right) \tag{2.7}$$

The similarity (or difference) between two green patterns g1 and g2 is defined as the vector 1-norm:

$$\|g1 - g2\| = \sum_{0 \le i < 4} |g1_i - g2_i| \tag{2.8}$$

It is likely that the two pixel locations where the two green patterns are defined will have similar red and blue color components when the difference between two green patterns is small.

Missing color component is estimated by performing a weighted average proportional to the degree of similarity of the green patterns. For example, the missing blue color $b_{m,n}$, in pixel location (m, n) contains only red color component is estimated by comparing the green pattern $g(m,n)$ with the four neighbouring green patterns $g(m-1,n-1), g(m-1,n+$

$1), g(m+1, n-1)\ and\ g(m+1, n+1)$. If all the differences between $g(m, n)$ and other four green patterns are uniformly small, then a simple average is used to estimate the missing blue color component,

$$b_{m,n} = \frac{B_{m-1,n-1} + B_{m-1,n-1} + B_{m-1,n+1} + B_{m+1,n+1}}{4} \tag{2.9}$$

Otherwise, when the largest difference is above certain threshold, only the top two best-matched green patterns information are used. If $\|g(m, n) - g(m-1, n-1)\|$ and $\|g(m, n) - g(m+1, n-1)\|$ are the two smallest differences, then the missing blue color is estimated as follows.

$$b_{m,n} = \frac{\|g(m, n) - g(m-1, n-1)\| * B_{m-1,n-1} + \|g(m, n) - g(m+1, n-1)\| * B_{m+1,n-1}}{\|g(m, n) - g(m-1, n-1)\| + \|g(m, n) - g(m+1, n-1)\|} \tag{2.10}$$

Similarly the missing red color values can be computed.

This algorithm is simple and efficient. However, as pointed out by Wu, et al [43], the quality of reconstructed images is still undesirable.

2.2.2 Edge Sensing Interpolation: Different predictors are used for estimating the missing green values in the edge directed interpolation method depending on the luminance gradients [19]. First, two gradients, one in horizontal direction, the other in vertical direction are computed for each red or blue pixel location. For instance, consider the pixel B8 as shown in figure 2.2. The two gradients are defined as,

$$\Delta \mathrm{H} = |G7 - G9|\ ;\ \Delta \mathrm{V} = |G3 - G13| \tag{2.11}$$

Based on these gradient values and a certain threshold (T), the interpolation algorithm then can be described as follows.

$$
\begin{aligned}
&if\ \ \Delta\mathrm{H} < \mathrm{T} \text{ and } \Delta\mathrm{V} > \mathrm{T} \text{ then} \\
&\quad g8 = (G7 + G9)/2; \\
&else\ if\ \Delta\mathrm{H} > \mathrm{T} \text{ and } \Delta\mathrm{V} < \mathrm{T} \text{ then} \\
&\quad g8 = (G3 + G13)/2; \\
&else \\
&\quad g8 = (G3 + G7 + G9 + G13)/4; \\
&endif \\
&endif
\end{aligned}
\tag{2.12}
$$

A slightly different edge sensing interpolation algorithm is described in [10]. Instead of luminance gradients, chrominance gradients are used. The two gradients, refer to figure 2.3 below, are defined as:

$$\Delta\mathrm{H} = \left|B5 - \frac{B3+B7}{2}\right| \ ; \ \Delta\mathrm{V} = \left|B5 - \frac{B1+B9}{2}\right| \tag{2.13}$$

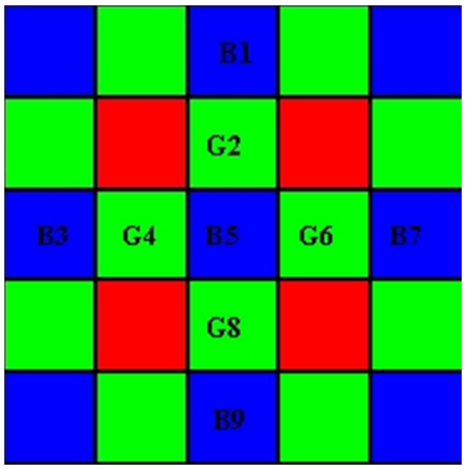


Figure 2.3: 5×5 Bayer CFA pattern (BGGR)

2.2.3 Linear Interpolation with Laplacian second-order Correction terms: This algorithm [9] focuses on improving the visual quality of the interpolated image when applied on images with sharp edge. Missing color components are interpolated using following steps.

The first step in this algorithm is to interpolate the missing green color components at the red and blue pixels. Consider the interpolation of green value at a blue pixel location (using figure 2.3) as an example. Interpolation of green value at red pixel location can be done in the similar fashion. Now interpolate the missing green component g5 at pixel location B5. We define horizontal and vertical gradients in this pixel location as follows:

$$\Delta\mathrm{H} = |G4 - G6| + |B5 - B3 + B5 - B7|\ ;\ \Delta\mathrm{V} = |G2 - G8| + B5 - B1 + B5| \qquad (2.14)$$

Intuitively, we can consider ΔH and ΔV above as combination of the luminance gradient and the chrominance gradient as described in edge sensing interpolation algorithm in the previous section. Using these two gradient values, the missing green component g5 at pixel location B5 is estimated as follows.

$if\ \ \Delta\mathrm{H} < \Delta\mathrm{V}$ then

$\quad g5 = (G4 + G6)/2 + (B5 - B3 + B5 - B7)/4;$

$else\ if\ \Delta\mathrm{H} > \Delta\mathrm{V}$ then

$\quad g5 = (G2 + G8)/2 + (B5 - B1 + B5 - B9)/4;$

$else$

$\quad g5 = (G2 + G4 + G6 + G8)/4 + (B5 - B1 + B5 - B3 + B5 - B7 + B5 - B9)/8;$

$endif$

$endif$ (2.14)

The interpolation step for g5 has two parts. The first part is the linear average of the neighbouring green values, and the second part can be considered as a second-order correction term based on the neighbouring blue (red) values.

The missing red (or blue) color components are estimated in every pixel location after estimation of the missing green components in every pixel location. Depending on the position, refer to figure 2.4, we have three cases:

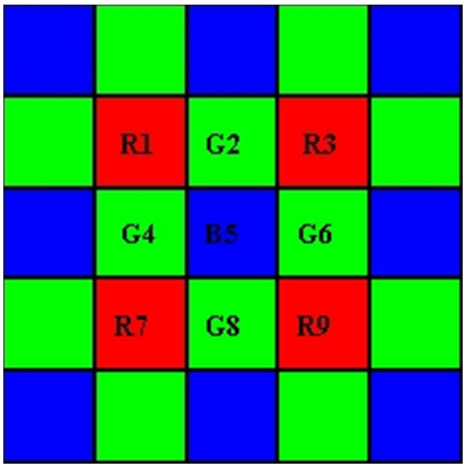


Figure 2.4: 5×5 Bayer CFA pattern (BGGR)

1. Estimate red (blue) color component at a green pixel where nearest neighbours of red (blue) pixels are in the same column, e.g. pixel location G4 as shown in figure 2.4 above. We estimate the red component r4 at pixel location G4 as follows.

$$r4 = (R1 + R7)/2 \ + (G4 - g1 + G4 - g7)/4 \tag{2.15}$$

2. Estimate red (blue) color component at a green pixel where nearest neighbours of red (blue) pixels are in the same row, e.g. pixel location G2 as shown in fig 2.4. We estimate the red component r2 at pixel location G2 as follows.

$$r2 = (R1 + R3)/2 + (G2 - g1 + G2 - g3)/4 \tag{2.16}$$

3. Estimate red (blue) color component at a blue (red) pixel. For instance, estimate red component r5 at pixel location B5 as shown in figure 2.4. Here we first define two diagonal gradients as follows:

$$\Delta \mathrm{N} = |R1.R9| + |g5 - g1 + g5 - g9| \; ;$$

$$\Delta \mathrm{P} = |R3.R7| + |g5 - g3 + g5 - g7| \tag{2.17}$$

Using these diagonal gradients, the algorithm for estimating the missing color components is described as:

$if \;\; \Delta \mathrm{N} < \; \Delta \mathrm{P}$ then

$\quad r5 = (R1 + R9)/2 + (g5 - g1 + g5 - g9)/2;$

$else\ if\ \Delta \mathrm{N} > \Delta \mathrm{P}$ then

$\quad r5 = (R3 + R7)/2 + (g5 - g3 + g5 - g7)/2;$

$else$

$\quad r5 = (R1 + R3 + R7 + R9)/4 + (g5 - g1 + g5 - g3 + \; g5 - g7 + g5 - g9)/4;$

$endif$

$endif$ (2.18)

This method provides much better visual quality of the reconstructed image containing a lot of sharp edges. However, the second-order derivative for calculating the gradients makes the algorithm quite sensitive to noise. Since only the color information in the same direction (vertical, horizontal, or one of the diagonal directions based on the gradient information) is used for interpolation, we believe that it is still possible to further improve the visual quality of the reconstructed image.

2.2.4 Interpolation using a Threshold-based variable number of gradients: This algorithm is described in [44]. A set of gradients is determined from the color values in the 5x5 neighbourhood centred at the pixel under consideration. Each gradient corresponds to a different direction. For each set of gradients, a threshold value is determined and the threshold is used to select a subset of gradients. Low-valued gradients indicate pixels having similar color values whereas high-valued gradients would be expected in regions of the image

where there are many fine details or sharp edges. The subset of gradients is used to locate regions of pixels that are most like the pixel under consideration. The pixels in these regions are then weighted and summed to determine the average difference between the color of the actual measured center pixel value and the missing color. A similar approach using weighted gradients is given in [67]. The algorithm is illustrated by an example using a 5 x 5 block as shown in figure 2.5.

1. Interpolation of the green, red/blue value at the blue/red pixel: consider figure 2.5 below, we want to estimate g13 and b13 at R13.

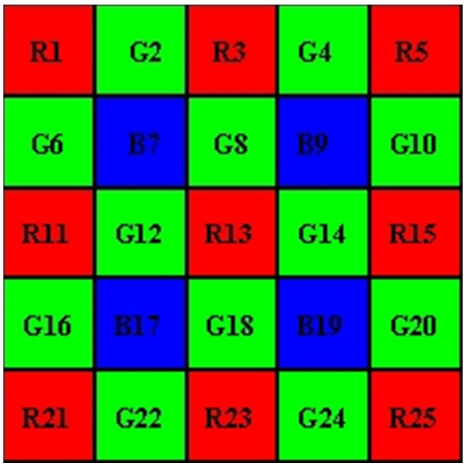


Figure 2.5: 5×5 Bayer CFA pattern (RGGB)

Form eight gradients as follows :

$$Gradient\ N = |G8 - G18| + |R3 - R13| + |B7 - B17|/2 + |B9 - B19|/2 + |G2 - G12|/2 + |G4 - G14|/2;$$

$$Gradient\ E = |G14 - G12| + |R15 - R13| + |B9 - B7|/2 + |B19 - B17|/2 + |G10 - G8|/2 + |G20 - G18|/2;$$

$$Gradient\ S = |G18 - G8| + |R23 - R13| + |B19 - B9|/2 + |B17 - B7|/2 + |G24 - G14|/2 + |G22 - G12|/2;$$

$$Gradient\ W = |G12 - G14| + |R11 - R13| + |B17 - B19|/2 + |B7 - B9|/2 + |G16 - G18|/2 + |G6 - G8|/2;$$

$$Gradient\ NE = |B9 - B17| + |R5 - R13| + |G8 - G12|/2 + |G14 - G18|/2 + |G4 - G8|/2 + |G10 - G14|/2;$$

$$Gradient\ SE = |B19 - B7| + |R25 - R13| + |G14 - G8|/2 + |G18 - G12|/2 + |G20 - G14|/2 + |G24 - G18|/2;$$

$$Gradient\ NW = |B7 - B19| + |R1 - R13| + |G12 - G18|/2 + |G8 - G14|/2 + |G6 - G12|/2 + |G2 - G8|/2;$$

$$Gradient\ SW = |B17 - B9| + |R21 - R13| + |G18 - G14|/2 + |G12 - G8|/2 + |G22 - G18|/2 + |G16 - G12|/2; \qquad (2.19)$$

Determine a threshold and select a subset of gradients: the threshold is determined by $T = k1 * Min + k2 * (Max - Min)$, where Min is the minimum gradient value and Max is the maximum gradient value. $k1$ and $k2$ are determined experimentally as 1.5 and 0.5, respectively. Here $k1 * Min$ accounts for the case in which the gradients are all very similar, so that we wish to include all of them by setting a threshold that exceeds them. $k2 * (Max - Min)$ accounts for the case in which there is a significant difference between the maximum and minimum gradient values.

Now, locate the pixels in the regions corresponding to the subset of gradients and to use those pixels to determine a color difference between the center pixel color and the color to be recovered. Determine the average green, blue and red values in the gradient subset regions: Form the average color values in the gradient subset regions to get G_{sum}, R_{sum} and B_{sum}

Find the normalized color difference by dividing the difference of two sums by the number of gradients in the threshold subset, and add this normalized color difference to the pixel value under consideration to form the other two missing color components.

2. Interpolation of the blue/red value at the green pixel: consider figure 2.2, we want to estimate r13 and b13 at G13.

Form eight gradients as follows

$$Gradient\ N = |G3 - G13| + |B8 - B18| + |G7 - G17|/2 + |G9 - G19|/2 + |R2 - R12|/2 + |R4 - R14|/2;$$

$$Gradient\ E = |R14 - R12| + |G15 - G13| + |G9 - G7|/2 + |G19 - G17|/2 + |B10 - B8|/2 + |B20 - B18|/2;$$

$$Gradient\ S = |B18 - B8| + |G23 - G13| + |G19 - G9|/2 + |G17 - G7|/2 + |R24 - R14|/2 + |R22 - R12|/2;$$

$$Gradient\ W = |R12 - R14| + |G11 - G13| + |G17 - G19|/2 + |G7 - G9|/2 + |B16 - B18|/2 + |B6 - B8|/2;$$

$$Gradient\ NE = |G9 - G17| + |G5 - G13| + |R4 - R12| + |B10 - G18|;$$

$$Gradient\ SE = |G19 - G7| + |G25 - G13| + |B20 - B8| + |R24 - R12|;$$

$$Gradient\ NW = |G7 - G19| + |G1 - G13| + |B6 - B18|/2 + |R2 - R14|;$$

$$Gradient\ SW = |G17 - G9| + |G21 - G13| + |R22 - R14| + |B16 - B8|;$$

(2.20)

As described earlier, determine a threshold and select a subset of gradients. Again, $T = k1 * Min + k2 * (Max - Min)$, where $k_1 = 1.5$ and $k_2 = 0.5$. Again, Locate pixels in the selected regions and use those pixels to determine a color difference between the center pixel color

and the color to be recovered, finally add this color difference to produce an estimate for the missing color value.

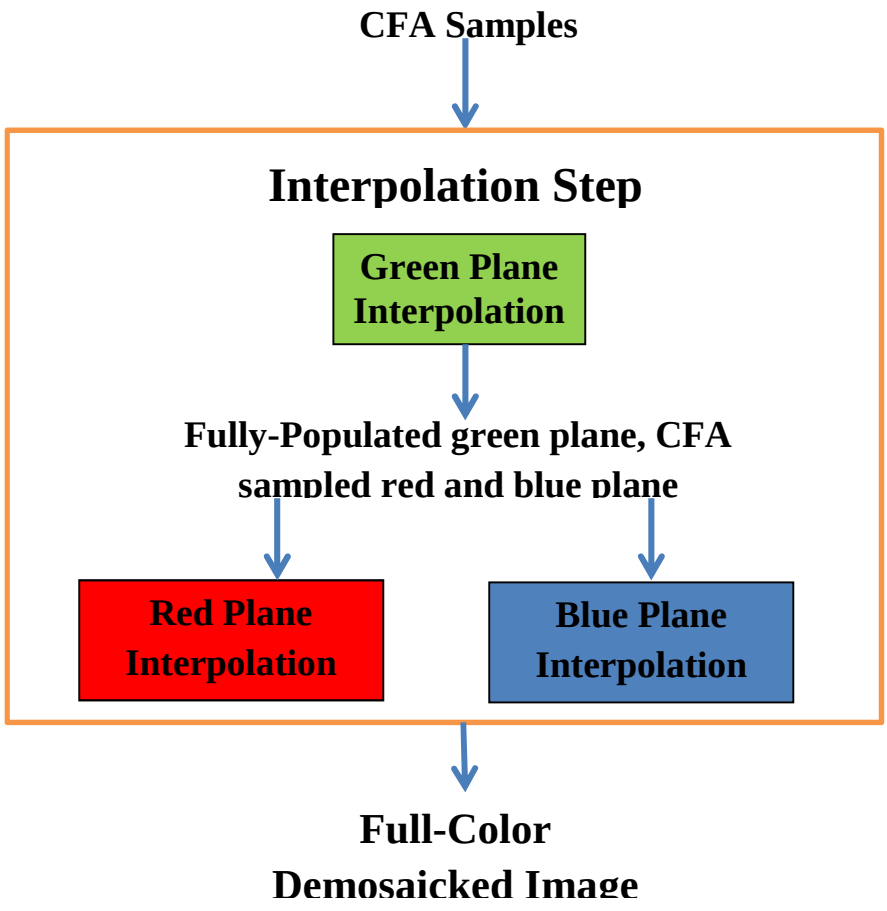
2.3 Algorithms exploiting the correlation between color planes in an image

This section explains two demosaicking algorithms,

2.3.1 Edge Strength Based Color Filter Array Interpolation [45].

2.3.2 Practical Color Filter Array Interpolation with Non-Linear Filter [46].

Algorithm 2.3.1 [45] proposes an edge strength filter that provides local, orientation-free intensity transition information. This algorithm utilizes this edge strength information combined with constant color difference assumption to interpolate the initial green channel while avoiding averaging across edge structures. The algorithm further uses this information to update the initially interpolated green channel. Algorithm 2.3.2 [46] proposes a filter that has a simple structure and is effective eliminating artifacts on the edge of color boundaries. This algorithm is the simpler and improved version of its previous algorithm [47] which uses a linear low pass filter and was more constrained. The algorithm makes use of the constant color difference correlation assumption combined with an edge detection technique which detects the direction in which the correlation is high and uses the signal of higher correlation to execute interpolation. Both algorithms works in three steps, the green channel is interpolated first followed by the interpolation of red and blue channel as shown in figure 2.6. These algorithms are explained in step by step as follows:



2.6: Three Step CFA Interpolation

2.3.1 Edge Strength Filter Based Color Filter Array Interpolation: The algorithm proposes an edge strength filter that provides local, orientation- free luminance transition information. The filter has a 3 by 3 block size as shown in figure 2.7. Given a grey scale input image, it could be formulated as

$$S_{P5} = \frac{|P_1 - P_9|}{2} + \frac{|P_3 - P_7|}{2} + |P_2 - P_8| + |P_4 - P_6| \qquad (2.21)$$

P1	P2	P3
P4	P5	P6
P7	P8	P9

Figure 2.7: 3x3 Edge Strength Filter Pattern

Where S_{P5} is the edge strength at pixel P5.

By applying the above filter to all available pixels, we obtain the edge strength map of the input image. Although the filter result for a single pixel does not provide any edge direction information, the relationship between neighbouring pixel is exploited that yields the edge orientation in that neighbourhood. The edge strength for green and blue pixels will be calculated in the same way. The edge strength map obtained from the raw image data will help us both in initial green channel interpolation stage and in subsequent green channel update.

Step 1: Green Channel Interpolation and Updation

Labelling each pixel as horizontal or vertical by comparing edge strength differences along each direction on a local window. For a window of 5 by 5, horizontal and vertical difference costs can be formulated as follows:

$$H_{i,j} = \sum_{m=-2}^{2}\left(\sum_{n=-2}^{1}\left(S_{i+m,\,j+n} - S_{i+m,\,j+n-1}\right)\right);$$

$$V_{i,j} = \sum_{m=-2}^{1}\left(\sum_{n=-2}^{2}\left(S_{i+m,\,j+n} - S_{i+m+1,\,j+n}\right)\right) \quad (2.22)$$

Where $S_{i,j}$ is the edge strength filter output at pixel location (i, j) and $H_{i,j}$ and $V_{i,j}$ represent the total horizontal and vertical costs. Now the target pixel will be labelled as horizontal if horizontal cost is less than vertical cost and vice versa. Based on the edge direction labels, green channel is interpolated as:

$$G'_{i,j} = \begin{cases} B_{i,j} + \dfrac{\bar{G}^{H}_{i,j} - B_{i,j}}{2} + \dfrac{G_{i,j-1} - \bar{B}^{H}_{i,j-1}}{4} + \dfrac{G_{i,j+1} - B^{H}_{i,j+1}}{4}, if\ horizontal \\ B_{i,j} + \dfrac{\bar{G}^{V}_{i,j} - B_{i,j}}{2} + \dfrac{G_{i-1,j} - \bar{B}^{V}_{i-1,j}}{4} + \dfrac{G_{i+1,j} - \bar{B}^{V}_{i+1,j}}{4}, if\ vertical \end{cases}$$

Where directional estimations are calculated by

$$\bar{G}_{i,j}^{H} = \frac{G_{i,j-1} + G_{i,j+1}}{2} + \frac{2 * B_{i,j} - B_{i,j-2} - B_{i,j+2}}{4}$$

$$\bar{G}_{i,j}^{V} = \frac{G_{i-1,j} + G_{i+1,j}}{2} + \frac{2 * B_{i,j} - B_{i-2,j} - B_{i+2,j}}{4}$$

$$\bar{B}_{i,j}^{H} = \frac{B_{i,j-1} + B_{i,j+1}}{2} + \frac{2 * G_{i,j} - G_{i,j-2} - G_{i,j+2}}{4}$$

$$\bar{B}_{i,j}^{V} = \frac{B_{i-1,j} + B_{i+1,j}}{2} + \frac{2 * G_{i,j} - G_{i-2,j} - G_{i+2,j}}{4} \quad (2.23)$$

Green channel estimation for red pixel locations is performed simply by replacing B's with R's in the equation above.

Now we will this initially interpolated green channel. For every green pixel to be updated, we consider the four neighbours with available color difference estimates. We have assigned a weight to each neighbour pixel which is inversely correlated with the total absolute edge strength difference in its direction.

$$D_1 = \left|S_{i,j} - S_{i-1,j}\right| + \left|S_{i-1,j} - S_{i-2,j}\right| + \left|S_{i-2,j} - S_{i-3,j}\right| + C_1$$

$$D_2 = \left|S_{i,j} - S_{i,j-1}\right| + \left|S_{i,j-1} - S_{i,j-2}\right| + \left|S_{i,j-2} - S_{i,j-3}\right| + C_1$$

$$D_3 = \left|S_{i,j} - S_{i,j+1}\right| + \left|S_{i,j+1} - S_{i,j+2}\right| + \left|S_{i,j-2} - S_{i,j+3}\right| + C_1$$

$$D_4 = \left|S_{i,j} - S_{i+1,j}\right| + \left|S_{i+1,j} - S_{i+2,j}\right| + \left|S_{i+2,j} - S_{i+3,j}\right| + C_1$$

$$M_1 = D_2 * D_3 * D_4$$

$$M_2 = D_1 * D_3 * D_4$$

$$M_3 = D_2 * D_1 * D_4$$

$$M_4 = D_2 * D_3 * D_1$$

$$M_{total} = M_1 + M_2 + M_3 + M_4$$

$$\bar{G}_{i,j} = B_{i,j} + W * \left(\bar{G}_{i,j} - B_{i,j}\right) + (1 - W) * \left[\frac{M_1}{M_{total}}\left(\bar{G}_{i-2,j} - B_{i-2,j}\right) + \frac{M_2}{M_{total}}\left(\bar{G}_{i,j-2} - B_{i,j-2}\right) + \frac{M_3}{M_{total}}\left(\bar{G}_{i,j+2} - B_{i,j+2}\right) + \frac{M_4}{M_{total}}\left(\bar{G}_{i+2,j} - B_{i+2,j}\right)\right] \quad (2.24)$$

Similarly for red pixel, replace $B_{i,j}$ by $R_{i,j}$. Here C_1 is a non- zero constant to avoid zero denominator.

Step 2: Red and Blue channel interpolation at Green pixels

For red and blue channel estimation at green pixels, we employ bilinear interpolation over color differences. Here, only the nearest two neighbours for which the original pixel value available are used.

$$\bar{B}_{2i,2j} = G_{2i,2j} - \frac{\left(\bar{G}_{2i-1,2j} - B_{2i-1,2j}\right) + \left(\bar{G}_{2i+1,2j} - B_{2i+1,2j}\right)}{2}$$

$$\bar{B}_{2i+1,2j+1} = G_{2i+1,2j+1} - \frac{\left(\bar{G}_{2i+1,2j+1} - B_{2i+1,2j}\right) + \left(\bar{G}_{2i+1,2j+2} - B_{2i+1,2j+2}\right)}{2} \quad (2.25)$$

Similarly Red channel can be estimated at green pixels.

Step 3: Red and Blue channel interpolation at Blue and Red pixel respectively

For red channel interpolation at blue pixels and blue channel interpolation at red pixels, diagonal neithbours are used adaptively based on green channel gradients in both directions.

$$M_1 = \left|\bar{G}_{i-2,j-2} - \bar{G}_{i,j}\right| + \left|\bar{G}_{i-1,j-1} - \bar{G}_{i+1,j+1}\right| + \left|\bar{G}_{i,j} - \bar{G}_{i+2,j-2}\right|$$

$$M_2 = \left|\bar{G}_{i-2,j+2} - \bar{G}_{i,j}\right| + \left|\bar{G}_{i-1,j+1} - \bar{G}_{i+1,j-1}\right| + \left|\bar{G}_{i,j} - \bar{G}_{i+2,j-2}\right|$$

$$\bar{B}_{i,j} = \bar{G}_{i,j} - \frac{M_2 * \left(\bar{G}_{i-1,j-1} - B_{i-1,j-1} + \bar{G}_{i+1,j+1} - B_{i+1,j+1}\right)}{2 * (M_1 + M_2)} + \frac{M_1 * \left(\bar{G}_{i-1,j-1} - B_{i-1,j-1} + \bar{G}_{i+1,j-1} - B_{i+1,j-1}\right)}{2 * (M_1 + M_2)} \quad (2.26)$$

2.3.2 Color Filter Array Interpolation with Non Linear Filter: The pixel interpolation procedure is described below. First we compute the signal gradient in horizontal and vertical direction at red and blue pixels. Now because the number of G pixels is larger than any other, we first interpolate the G pixels on the R and B planes (Step 1). It is important to perform G pixel interpolation as the first step so that interpolation errors will not be propagated in the R and B pixel interpolation. Next, we perform interpolation of R and B signals on the G plane (Step 2), and finally we perform interpolation of the R signal in the B plane and the B signal in the R plane, which have the lowest inter-channel correlation (Step 3). In the computing expression given below, the coordinates of each signal are the same as the pixel array shown in figure 2.8.

Step 0: Compute signal gradient at R_{32} pixel location.

Vertical Signal Gradient:

$$V = \frac{Rv + Gv + Bv}{4}$$

$$Rv = \frac{(|-R_{12} + 2R_{32} - R_{52}|)}{2}$$

$$Gv = (|G_{22} - G_{42}| + \frac{|-G_{11} + 2 * G_{31} - G_{51}|}{4} + \frac{|-G_{13} + 2G_{33} - G_{53}|}{4})$$

$$Bv = \frac{(|B_{21} - B_{41}| + |B_{23} - B_{43}|)}{2} \quad (2.27)$$

Horizontal Signal Gradient:

$$H = \frac{Rh + Gh + Bh}{4}$$

$$Rh = \frac{(|-R_{30} + 2R_{32} - R_{34}|)}{2}$$

$$Gh = (|G_{31} - G_{33}| + \frac{|-G_{20} + 2G_{22} - G_{24}|}{4} + \frac{|-G_{40} + 2G_{42} - G_{44}|}{4})$$

$$Bh = \frac{(|B_{21} - B_{23}| + |B_{41} - B_{43}|)}{2} \quad (2.28)$$

G_{00}	B_{01}	G_{02}	B_{03}	G_{04}	B_{05}	G_{06}
R_{10}	G_{11}	R_{12}	G_{13}	R_{14}	G_{15}	R_{16}
G_{20}	B_{21}	G_{22}	B_{23}	G_{24}	B_{25}	G_{26}
R_{30}	G_{31}	R_{32}	G_{33}	R_{34}	G_{35}	R_{36}
G_{40}	B_{41}	G_{42}	B_{43}	G_{44}	B_{45}	G_{46}
R_{50}	G_{51}	R_{52}	G_{53}	R_{54}	G_{55}	R_{55}
G_{60}	B_{61}	G_{62}	B_{63}	G_{64}	B_{65}	G_{66}

Figure 2.8: 7×7 Bayer CFA Pattern (GBRG) showing Pixel Coordinates

At the B pixel location, V and H is similarly computed as the red plane changed R to B.

Step 1: G plane interpolation on R and B plane.

If V<= 4 and H <= 4 or H == V

$$g_{32} = R_{32} + (G_{32e} - R_{32e})$$

Where,

$$G_{32e} = \sum_i \sum_j \frac{G_{i,j}}{4}, \qquad ij \in 22,31,33,42$$

$$R_{32e} = \frac{\{\sum_i \sum_j R_{ij} + 4R_{32}\}}{8}, ij \in 12,30,34,52 \tag{2.29}$$

Else if V > H

$$g_{32} = R_{32} + (G_{32he} - R_{32he})$$

Where,

$$G_{32he} = \sum_i \sum_j \frac{G_{ij}}{2}, \quad ij \in 31,33$$

$$R_{32he} = \frac{\{\sum_i \sum_j R_{ij} + 2R_{32}\}}{4}, ij \in 30,34 \tag{2.30}$$

Else if V < H

$$g_{32} = R_{32} + (G_{32ve} - R_{32ve})$$

Where,

$$G_{32ve} = \sum_i \sum_j \frac{G_{ij}}{2}, \quad ij \in 22,42$$

$$R_{32ve} = \frac{\{\sum_i \sum_j R_{ij} + 2R_{32}\}}{4}, ij \in 12,52 \tag{2.31}$$

Then the blue plane is similarly computed as the red plane.

Step 2: R and B plane interpolation on G plane

Interpolating R values at G pixels:

$$r_{22} = G_{22} + (R_{22ve} - g_{22ve})$$

$$r_{33} = G_{33} + (R_{33he} - g_{33he})$$

Where,

$$R_{22ve} = \frac{\sum_i \sum_j R_{i,j}}{2}, ij \in 12{,}32$$

$$R_{33he} = \frac{\sum_i \sum_j R_{i,j}}{2}, ij \in 32{,}34$$

$$g_{22ve} = \frac{\sum_i \sum_j g_{i,j}}{2}, ij \in 12{,}32$$

$$g_{33he} = \frac{\sum_i \sum_j g_{i,j}}{2}, ij \in 32{,}34 \quad (2.32)$$

Interpolating B values at G pixels:

$$b_{22} = G_{22} + (B_{22he} - g_{22he})$$

$$b_{33} = G_{33} + (B_{33ve} - g_{33ve})$$

Where,

$$B_{22he} = \frac{\sum_i \sum_j B_{i,j}}{2}, ij \in 21{,}23$$

$$B_{33ve} = \frac{\sum_i \sum_j B_{i,j}}{2}, ij \in 23{,}43$$

$$g_{33ve} = \frac{\sum_i \sum_j g_{i,j}}{2}, ij \in 23{,}43$$

$$g_{22he} = \frac{\sum_i \sum_j g_{i,j}}{2}, ij \in 21{,}23 \quad (2.33)$$

Step 3: R and B plane interpolation on B and R plane respectively.

If *V<= 4 and H <= 4 or H == V*

$$r_{23} = g_{23} + (R_{23e} - g_{33e}) \tag{2.34}$$

Else if *V>H*

$$r_{23} = g_{23} + (r_{23he} - G_{33he}) \tag{2.35}$$

Else if *V<H*

$$r_{23} = g_{23} + (r_{23ve} - G_{23ve}) \tag{2.36}$$

Then the blue plane is similarly computed as the red plane.

Chapter 3

Proposed Demosaicking Methods based on Edge Strength Fuzzification

We have proposed two demosaicking algorithms, one is the modified version of the standard bilinear interpolation and other is the modified version of the Algorithm 2.3.2 [46] explained in previous section. These algorithms are modified in three respects:

- Edge information is been used effectively to remove artifacts.
- Fuzzification of edges based on their strengths to reduce computational complexity involved in decision making.
- Execution time of the algorithm is reduced.

The modified algorithms are better than the conventional demosaicking algorithms in terms of both objective and subjective quality. Both algorithms make use of an edge strength filter as explained in [45].

3.1 **Terminology**

Universe of Discourse: The Universe of Discourse is the range of all possible values for an input to a fuzzy system.

Fuzzy Set: A fuzzy set is a pair (U, m) where U is the universe of discourse and $m: U \rightarrow [o, 1]$. For each $x \epsilon U$ the value $m(x)$ is called the membership of x in (U, m). For a finite set $U = \{x_1, \dots, x_n\}$ the fuzzy set (U, m) is often denoted by $\{m(x_1)/x_1, \dots, m(x_n)/x_n\}$.

Fuzzy Weighted Average: Let A be the fuzzy set such that $A = \{m(x_1)/x_1, \ldots, m(x_n)/x_n\}$ for each $x \epsilon U$, where U is the universe of discourse. The fuzzy weighted average Y_A for the fuzzy set A is defined as:

$$Y_A = \frac{x_1\mu_1 + x_2\mu_2 + \cdots + x_n\mu_n}{\mu_2 + \mu_2 + \cdots + \mu_n} = \frac{\sum_{i=0}^{n} x_i\mu_i}{\sum_{i=0}^{n} \mu_i}$$

3.2 Fuzzy Membership Assignment Strategy for Proposed Algorithm

The basis of the proposed algorithm is the constant color ratio/difference over a local distance in a homogenous region. This assumption is likely to fail across edges. The edge information can be used adaptively during interpolation to avoid considering non-correlated color differences, interpolation quality can be improved. The question at this point is how the edge information can be expressed meaningfully at the pixel level so that it is useful enough to improve interpolation quality. The answer to this is fuzzification of the edge information in an image.

A fuzzy set EDGE_STRENGTH is defined over each pixel of the image as the universe of discourse. The edge strength at each pixel is computed and treated as the membership value of each pixel. The higher is the intensity variation across edge, the higher will be its strength.

The membership value at each pixel is inversely proportional to its edge strength. The higher is the edge strength value at each pixel, the lower will be its membership in fuzzy set. The reason here is strong edges will contribute less to the interpolation process because missing color values are computed by taking the fuzzy weighted average of the similar neighbouring pixels.

3.3 Edge Strength based Fuzzification of Bilinear Interpolation

An edge strength filter that provides local, orientation-free luminance transition information is proposed in [45]. The edge strength is computed and fuzzified for each pixel such that membership value for each pixel in fuzzy set is inversely proportional to the edge strength. The green channel is interpolated first by computing the fuzzy weighted average of green pixel values from neighbourhood. Next we interpolate Red and Blue channel on the Green pixels, and finally we perform interpolation of the Red channel on Blue pixels and Blue channel on Red pixels, which have the lowest inter-channel correlation. Bayer CFA pattern which is being used for the proposed demosaicking algorithm is shown in figure 3.1.

G_{00}	B_{01}	G_{02}	B_{03}	G_{04}	B_{05}	G_{06}
R_{10}	G_{11}	R_{12}	G_{13}	R_{14}	G_{15}	R_{16}
G_{20}	B_{21}	G_{22}	B_{23}	G_{24}	B_{25}	G_{26}
R_{30}	G_{31}	R_{32}	G_{33}	R_{34}	G_{35}	R_{36}
G_{40}	B_{41}	G_{42}	B_{43}	G_{44}	B_{45}	G_{46}
R_{50}	G_{51}	R_{52}	G_{53}	R_{54}	G_{55}	R_{55}
G_{60}	B_{61}	G_{62}	B_{63}	G_{64}	B_{65}	G_{66}

Figure 3.1: 7×7 Bayer CFA Pattern (GBRG) showing Pixel Coordinates

Step 0: Edge strength at each pixel is computed using an edge strength filter [45] as explained in in previous section. The filter has 3 by 3 support size as shown in figure 2.7 and formula is

same as explained in equation 2.21. After computing the edge strength S at each pixel, we will now fuzzify this edge strength at each pixel using the min-max normalization.

$$\mu = \left|1 - \frac{S}{\max(S)}\right| \tag{3.1}$$

Step 1: Green channel Interpolation at red/blue pixel position, for example consider a red pixel R_{32} and a blue pixel $B_{23}3$, green value can be interpolated by taking fuzzy weighted average of four neighbour green pixel values as shown in equation 3.2 and 3.3. Please refer figure 3.1 for the pixel pattern and their positions.

$$g_{32} = \frac{\sum_i \sum_j \mu_{ij} G_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 22, 31, 33, 42 \tag{3.2}$$

$$g_{23} = \frac{\sum_i \sum_j \mu_{ij} G_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 13, 33, 22, 24 \tag{3.3}$$

Step 2: Red and Blue channel Interpolation on Green pixel. There can be two cases depending upon the position of red or blue pixels in four neighbourhood of green pixel. In first case, the red pixels are located at the above and below of the green pixel and blue pixels are located at the left and right of the green pixel. For example, consider a green pixel G_{22} on which we are interpolating the red and blue pixel value by taking fuzzy weighted average of the two same color pixels as shown in equations 3.4 and 3.7. In second case, the red pixels are located at the left and right of the green pixel and blue pixels are located at the above and below of the green pixel. For example, consider a green pixel G_{33} on which we are interpolating the red and blue pixel value by taking the fuzzy weighted average of the same color pixels as shown in equations 3.5 and 3.6. Please refer figure 3.1 for the pixel pattern and their positions.

$$r_{22} = \frac{\sum_i \sum_j \mu_{ij} R_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 12, 32 \tag{3.4}$$

$$r_{33} = \frac{\sum_i \sum_j \mu_{ij} R_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 32, 34 \tag{3.5}$$

$$b_{33} = \frac{\sum_i \sum_j \mu_{ij} B_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 23{,}43 \tag{3.6}$$

$$b_{22} = \frac{\sum_i \sum_j \mu_{ij} B_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 21, 23 \tag{3.7}$$

Step 3: Red/Blue channel interpolation on Blue/Red pixels. Consider a blue pixel B_{21} and a red pixel R_{12}. The red value at B_{21} and blue value at R_{12} are interpolated by taking the fuzzy weighted average of the four diagonal pixels of the same color as shown in equations 3.8 and 3.9. Please refer figure 3.1 for the pixel pattern and their positions.

$$r_{21} = \frac{\sum_i \sum_j \mu_{ij} R_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 10, 12, 30, 32 \tag{3.8}$$

$$b_{12} = \frac{\sum_i \sum_j \mu_{ij} B_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 01, 03, 21, 23 \tag{3.9}$$

3.4 Edge Strength based Fuzzification of Non-Linear CFA Interpolation

An edge strength filter that provides local, orientation-free luminance transition information is proposed in [45]. The edge strength is computed and fuzzified for each pixel such that membership value for each pixel in fuzzy set is inversely proportional to the edge strength. The green channel is interpolated first by adding the color difference of red/blue and green

colors at current pixel value. The difference is computed between fuzzified low frequency components of each color. Fuzzified low frequency component is the fuzzy weighted average of similar color pixel values from neighbourhood. Next we interpolate Red and Blue channel on the Green pixels, and finally we perform interpolation of the Red channel on Blue pixels and Blue channel on Red pixels, which have the lowest inter-channel correlation. Bayer CFA pattern which is being used for the proposed demosaicking algorithm is shown in figure 3.1.

Step 0: Edge strength at each pixel is computed using an edge strength filter [45] as explained in in previous section. The filter has 3 by 3 support size as shown in figure 2.7 and formula is same as explained in equation 2.21. After computing the edge strength *S* at each pixel, we will now fuzzify this edge strength at each pixel using the min-max normalization as shown in equation 3.1.

Step 1: Green channel interpolation on Red and Blue pixels. Consider a red pixel R_{32} and blue pixel B_{23}. The green value at R_{32} and B_{23} is interpolated by adding a color difference value at current pixel value as shown in equation 3.10 and 3.11. This color difference is computed by taking a fuzzy low frequency component for red/blue and green pixels at current position. At R_{32}, fuzzy low frequency component for red and green value is computed as shown in equations 3.12 and 3.13. At B_{23}, fuzzy low frequency component of the blue and green value is computed as shown in equations 3.14 and 3.15. Please refer figure 3.1 for the pixel pattern and their positions.

$$g_{32} = R_{32} + (G_{32e} - R_{32e}) \tag{3.10}$$

$$g_{23} = B_{32} + (G_{32e} - B_{32e}) \tag{3.11}$$

Where,

$$G_{32e} = \frac{\sum_i \sum_j \mu_{ij} G_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 22, 31, 33, 42 \quad (3.10)$$

$$R_{32e} = \frac{\sum_i \sum_j \mu_{ij} R_{ij} + 4R_{32}}{\sum_i \sum_j \mu_{ij} + 4\mu_{32}} + , ij\epsilon\ 12, 30, 34, 52 \quad (3.11)$$

$$G_{23e} = \frac{\sum_i \sum_j \mu_{ij} G_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 13, 33, 22, 24 \quad (3.10)$$

$$B_{23e} = \frac{\sum_i \sum_j \mu_{ij} B_{ij} + 4B_{23}}{\sum_i \sum_j \mu_{ij} + 4\mu_{32}} + , ij\epsilon\ 03, 43, 21, 25 \quad (3.11)$$

Step 2: Red and Blue channel interpolation on Green pixels. There can be two cases depending upon the position of red or blue pixels in four neighbourhood of green pixel. In first case, the red pixels are located at the above and below of the green pixel and blue pixels are located at the left and right of the green pixel. For example, consider a green pixel at G_{22} on which we are interpolating the red and blue pixel value by adding color difference to the current pixel value as shown in equations 3.12 and 3.14. In second case, the red pixels are located at the left and right of the green pixel and blue pixels are located at the above and below of the green pixel. For example, consider a green pixel G_{33} on which we are interpolating the red and blue pixel value by adding color difference to current pixel value as shown in equations 3.13 and 3.15. Please refer figure 3.1 for the pixel pattern and their positions. Please refer figure 2.8 for the pixel pattern and their positions.

$$r_{22} = G_{22} + (R_{22ve} - g_{22ve}) \quad (3.12)$$

$$r_{33} = G_{33} + (R_{33he} - g_{33he}) \quad (3.13)$$

$$b_{22} = G_{22} + (B_{22he} - g_{22he}) \tag{3.14}$$

$$b_{33} = G_{33} + (B_{33ve} - g_{33ve}) \tag{3.15}$$

Where,

$$R_{22ve} = \frac{\sum_i \sum_j \mu_{ij} R_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 12,32 \tag{3.16}$$

$$R_{33he} = \frac{\sum_i \sum_j \mu_{ij} R_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 32,34 \tag{3.17}$$

$$g_{33he} = \frac{\sum_i \sum_j \mu_{ij} g_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 33,35 \tag{3.18}$$

$$g_{22ve} = \frac{\sum_i \sum_j \mu_{ij} g_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 12,32 \tag{3.19}$$

$$B_{22he} = \frac{\sum_i \sum_j \mu_{ij} B_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 21,23 \tag{3.20}$$

$$B_{33ve} = \frac{\sum_i \sum_j \mu_{ij} B_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 23,43 \tag{3.21}$$

$$g_{33ve} = \frac{\sum_i \sum_j \mu_{ij} g_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 23,43 \tag{3.22}$$

$$g_{22he} = \frac{\sum_i \sum_j \mu_{ij} g_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 21,23 \tag{3.23}$$

Step 3: Red/Blue channel interpolation on Blue/Red pixel respectively. Consider a blue pixel B_{23} and a red pixel R_{32}. The red value at B_{23} and blue value at R_{32} are interpolated by adding the color difference of red/blue and green color to the current pixel value as shown in equations 3.24 and 3.25. The fuzzy low frequency component for the red/blue color at current

pixel is computed using four diagonal pixels of the same color as shown in equations 3.26, 3.27, 3.28 and 3.29. Please refer figure 3.1 for the pixel pattern and their positions.

$$r_{23} = g_{23} + (R_{23e} - g_{23e}) \tag{3.24}$$

$$b_{32} = g_{32} + (B_{32e} - g_{32e}) \tag{3.25}$$

Where,

$$R_{23e} = \frac{\sum_i \sum_j \mu_{ij} R_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 12, 14, 32, 34 \tag{3.26}$$

$$g_{23e} = \frac{\sum_i \sum_j \mu_{ij} g_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 13, 22, 24, 33 \tag{3.27}$$

$$B_{32e} = \frac{\sum_i \sum_j \mu_{ij} B_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 21, 23, 41, 34 \tag{3.28}$$

$$g_{32e} = \frac{\sum_i \sum_j \mu_{ij} g_{ij}}{\sum_i \sum_j \mu_{ij}}, ij\epsilon\ 22, 31, 33, 42 \tag{3.29}$$

3.5 Graphical Analysis of the Proposed Algorithms

We have analysed the improvement of our proposed algorithm over the algorithms explained in section 2.3. We have cropped two 9x9 block size regions from an image out of which one block belongs to the region of homogeneous intensity and second block belongs to a region where intensity is changing abruptly as we move across the region. The cropped regions are shown in figure 3.2. We have illustrated the errors that occurred during interpolation for both versions of the algorithm and compared those using graphs. These graphs illustrate the signal correlation existent between the original signal and the interpolated signal. The X-axis in the

graph indicates the pixel position and Y-axis in the graph indicates the intensity. There can be three types of correlation between two signals:

- Positive Correlation
- Non-Correlation
- Negative Correlation

Positive Correlation: The interpolated signal intensity increases or decreases with increase or decrease in the original signal intensity respectively.

Non-Correlation: The interpolated signal has no correlation with the original signal. This means the interpolated signal intensity remains constant with the increase or decrease in original signal intensity.

Negative Correlation: The interpolated signal intensity decreases with increase in original 9signal intensity and increases with decrease in original signal intensity.

Each graph shows values of 9 pixels for two algorithms and there are a total of 8 graphs, 4 each for an algorithm in four directions i.e., horizontal, vertical, left diagonal and the right diagonal. Same graphs are plotted for the homogenous region of the image. The pixel located at position 5 is the centre pixel. The gap between the points on same pixel position in a graph shows the interpolation error. The larger will be the gap, the more will be the interpolation error. From the graphs it can be clearly seen that the proposed versions of the algorithms shows less interpolation errors that the conventional algorithms. Graphs shown in figure 3.3 and figure 3.4 compare the bilinear interpolation with proposed edge strength based fuzzy bilinear interpolation algorithm explained in section 3.2.1. Figure 3.3 shows graphs for the homogenous region and figure 3.4 shows graphs for the edge region. Graphs shown in figure

3.5 and figure 3.6 compare the method explained in section 2.3.2 with proposed algorithm explained in section 3.2.1. Figure 3.5 shows graphs for the homogenous region and figure 3.6 shows graphs for the edge region.

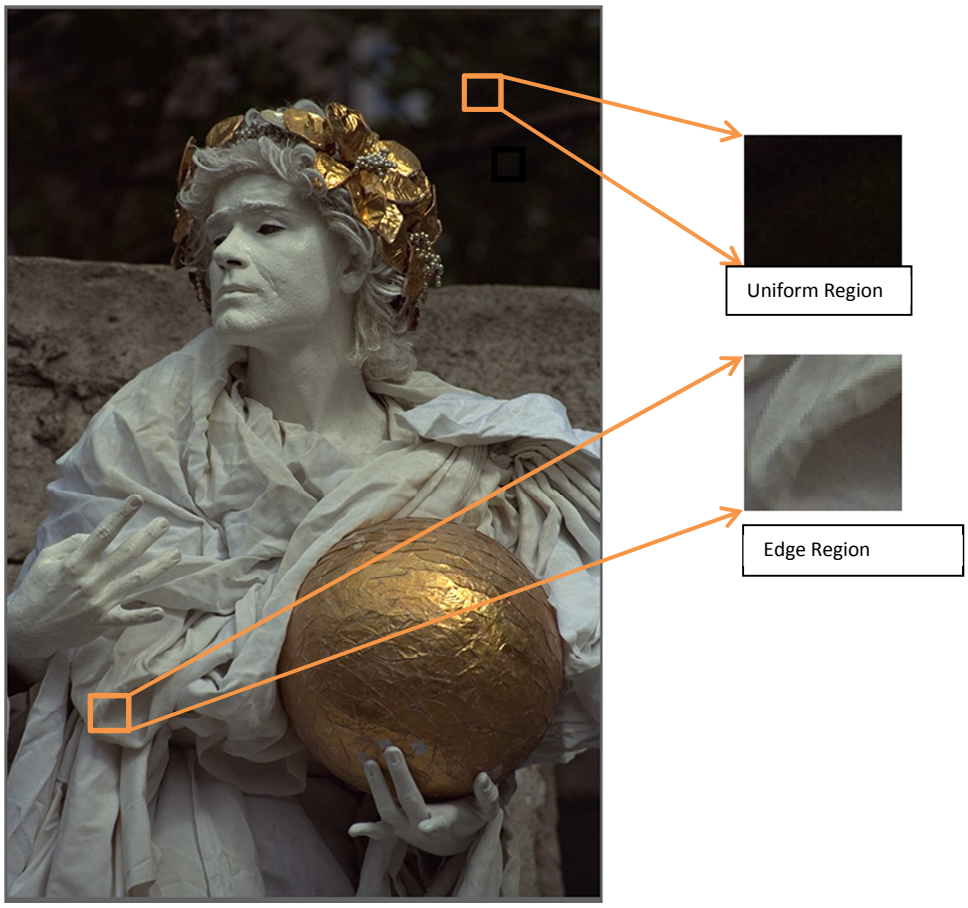
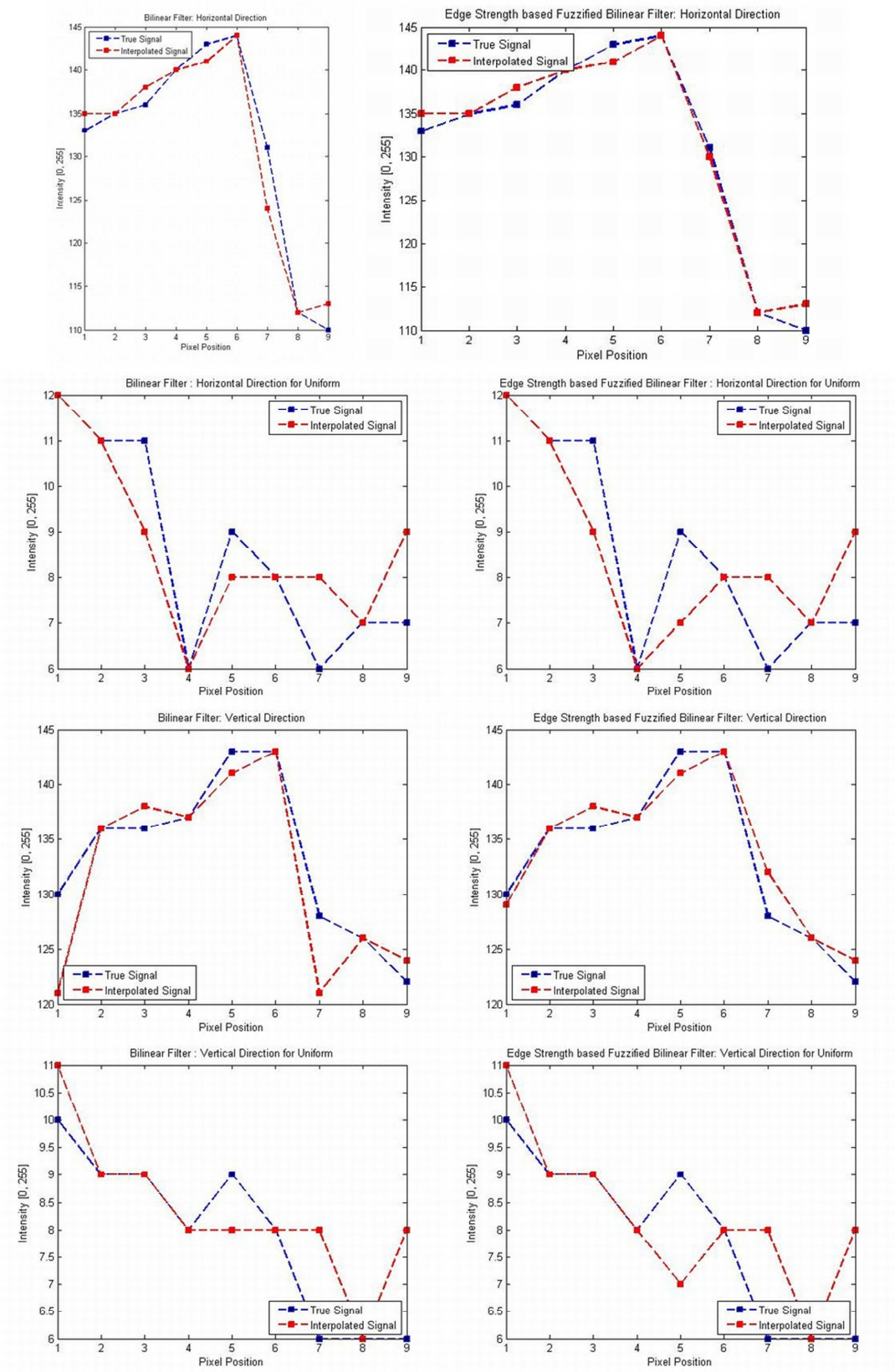
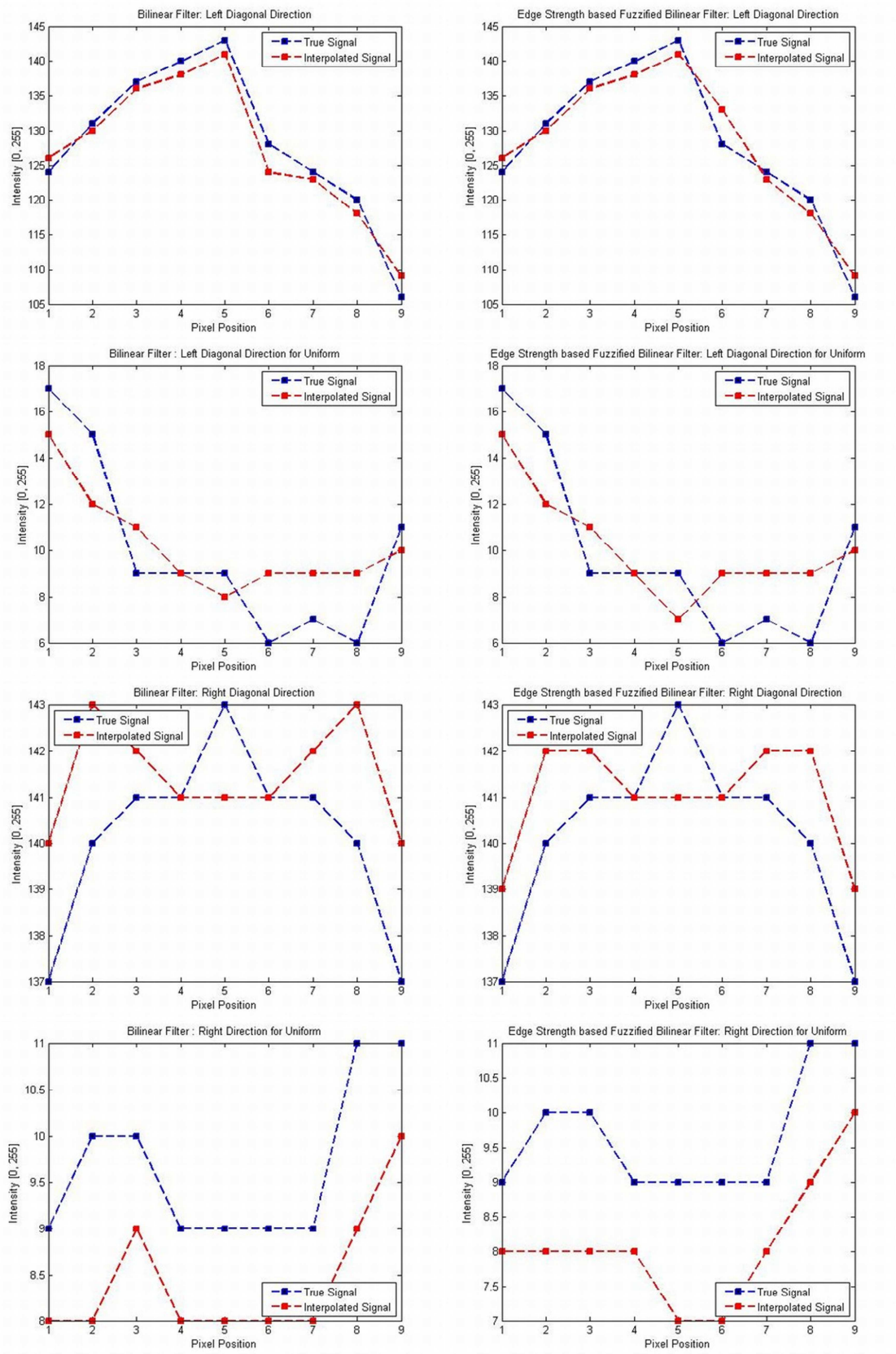


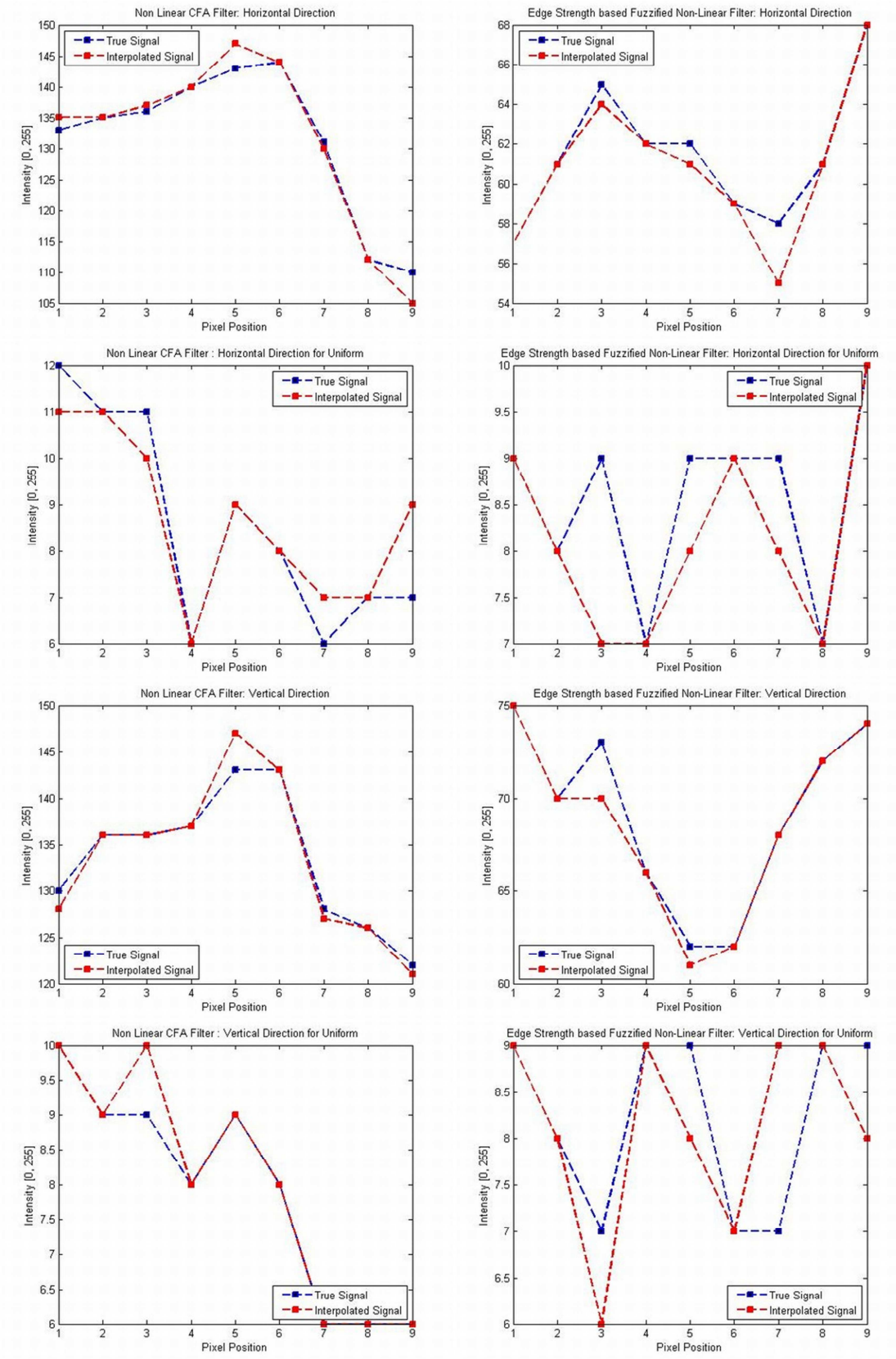
Figure 3.2: Cropped regions from the original image.



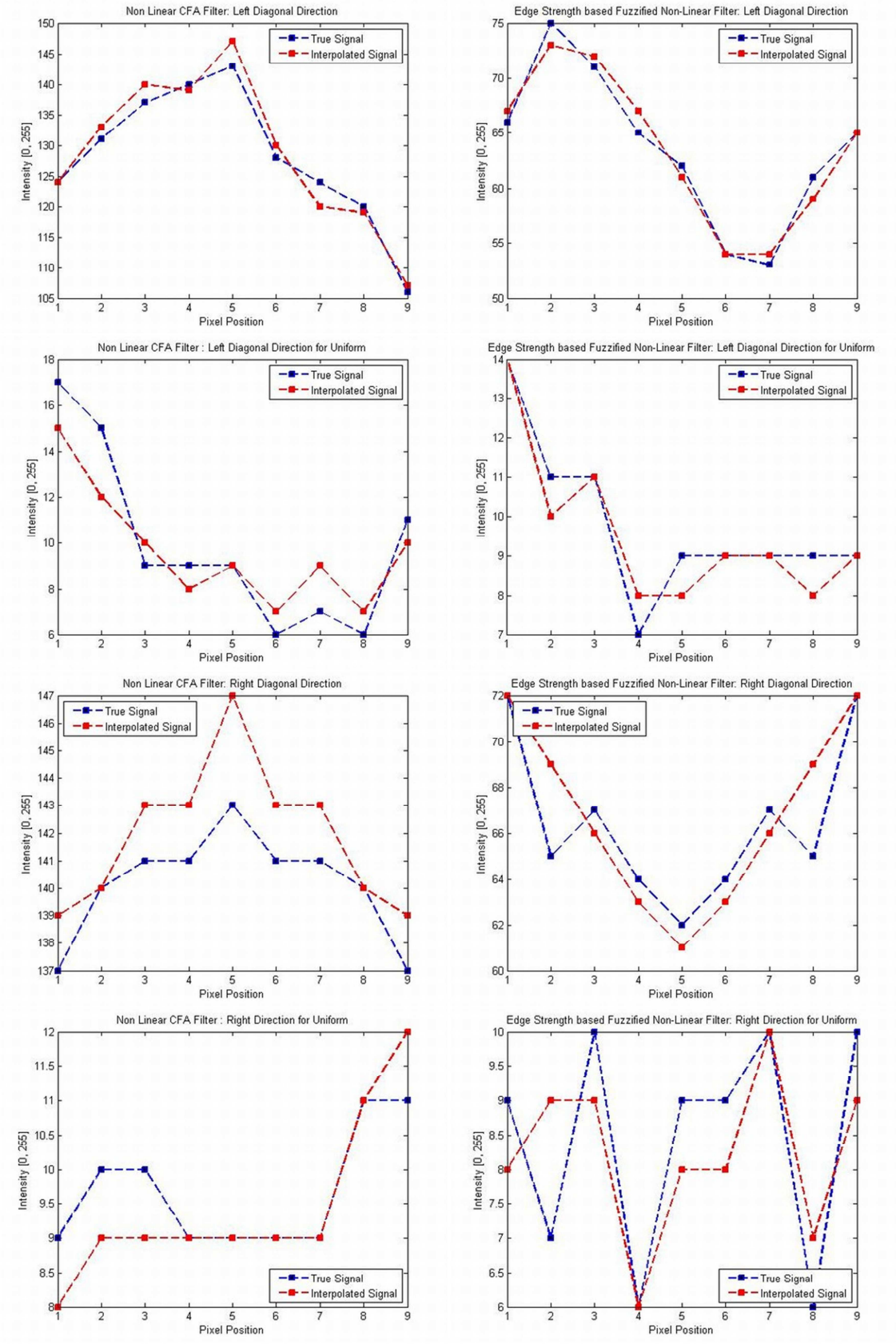
3.3: Bilinear Interpolation v/s Edge Strength based Fuzzified Bilinear Filter (Edge Region) with directions as labelled



3.4: Bilinear Interpolation v/s Edge Strength based Fuzzified Bilinear Filter (Homogenous 9Intensity Region) with directions as labelled.



3.5: Non-Linear Filter based CFA Interpolation v/s Edge Strength based Fuzzified Non-Linear CFA Interpolation (Edge Region) with directions as labelled.



3.6: Non-Linear Filter based CFA Interpolation v/s Edge Strength based Fuzzified Non-Linear CFA Interpolation (Edge Region) with directions as labelled.

Chapter 4

Experimental Results and Comparisons

4.1 Environmental Setup

The following system configuration has been used while conducting the experiments:

Hardware configuration

Processor:	AMD A10-4600M APU
Clock Speed:	2.3 GHz
Main Memory:	4 GB
Hard Disk Capacity:	1 TB

Software Configuration

Operating System:	Windows 8
Software Used:	MATLAB 7.9.0 (2009b)

We evaluate the performance of our proposed algorithm on five image datasets obtained from different domains namely, Nikon Microscopy Digital Images [49], Satellite Color Images [56], High Definition Color Images, Kodak Loss-Less True Color Images [57], Berkeley Segmentation Image Database [59]. The images are first synthetically sub-sampled in Bayer CFA pattern and then interpolated back to three channels using proposed algorithm. The details about each database with their experimental results are explained in further sections

4.2 Comparison with other methods

We have compared our algorithm with some of the algorithms in terms of objective measures and subjective quality measures. Objective measures are computed for each of the output images to determine the difference between the original image and the reconstructed image.

The following demosaicking methods are used for comparison, (a) Nearest Neighbour Replication, (b) Bilinear Interpolation, (c) Smooth Hue Transition Interpolation, (d) Pattern Matching Algorithm, (e) Edge Directed Interpolation, (f) Color Interpolation using Laplacian Second order color correction I, (g) Threshold based Variable Number of Gradients, (h) Gradient Corrected Linear Interpolation, (i) Edge Strength based CFA Interpolation, (j) Non-Linear Filter based CFA interpolation. All these algorithms are implemented in MATLAB. The MATLAB code for algorithms (a), (c), (d), (f) and (g) are obtained from [48]. Our proposed algorithm (k) Edge Strength based Fuzzified Bilinear Interpolation and (l) Edge Strength based Non-Linear Filter based CFA interpolation shows much better results than many of the other algorithms in terms of subjective quality and objective measures.

4.3 Evaluation Metrics

These objective measures and the detailed results for each database is explained in following sections.

4.3.1 Mean Squared Error (MSE)

The mean squared error (MSE) of an estimator is one of the ways to quantify the difference between values implied by an estimator and the true values of the quantity being estimated. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss. MSE measures the mean of the squares of the "errors." The error is the amount by which the value implied by the estimator differs from the true value of the quantity to be estimated. The difference occurs because of randomness or because the estimator doesn't account for information that could produce a more accurate estimate.

If $\hat{Y}_i$ is a dataset of n estimations, and Y_i is the dataset of the true values, then the (estimated) MSE of the estimator is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2 \quad (4.1)$$

4.3.2 Peak Signal-to-Noise Ratio (PSNR)

Peak signal-to-noise ratio is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. PSNR is usually expressed in terms of the logarithmic decibel scale because many signals have a very wide dynamic range. PSNR is most commonly used to measure the quality of reconstruction from sub-sample image data (e.g., for .image demosaicking). A higher PSNR generally indicates that the reconstruction is of higher quality.

PSNR can be easily defined via the mean squared error (*MSE*). Given a loss-less *m*×*n* monochrome image *I* with 255 as its maximum grey level, *PSNR* is defined as:

$$PSNR = 10.log\left[\frac{255^2}{MSE}\right] \quad (4.2)$$

Where MSE is the mean squared error as explained in the above section.

4.3.3 CPU Time

CPU time (or CPU usage, process time) is the amount of time for which a central processing unit (CPU) was utilized for processing instructions of a computer program.The CPU time is often measured in clock ticks or seconds. We have computed the CPU time in seconds for the running program by using the standard MATLAB commands.

4.4 **Experimental Results and Discussion**

4.4.1 Nikon Microscopy Digital Color Image suite

The test set consists of 24 images with 700×504 pixel resolution as shown in figure 4.1. This image set is being used for the first time for testing the quality of color interpolation. This image set is obtained from [49]. There are many other sources [50]-[55] that provide these digital microscopy images. The interpolated images are compared to the original images and results are reported for all three performance measures. The MSE results are summarized in table 4.1, PSNR results are summarized in table 4.2 and the CPU time results are summarized in table 4.3. The best result for each image is highlighted with bold text. An image region which is cropped from the original image is presented in figure 4.2 for the visual quality comparison. This image region is compared with other algorithm in figure 4.3.

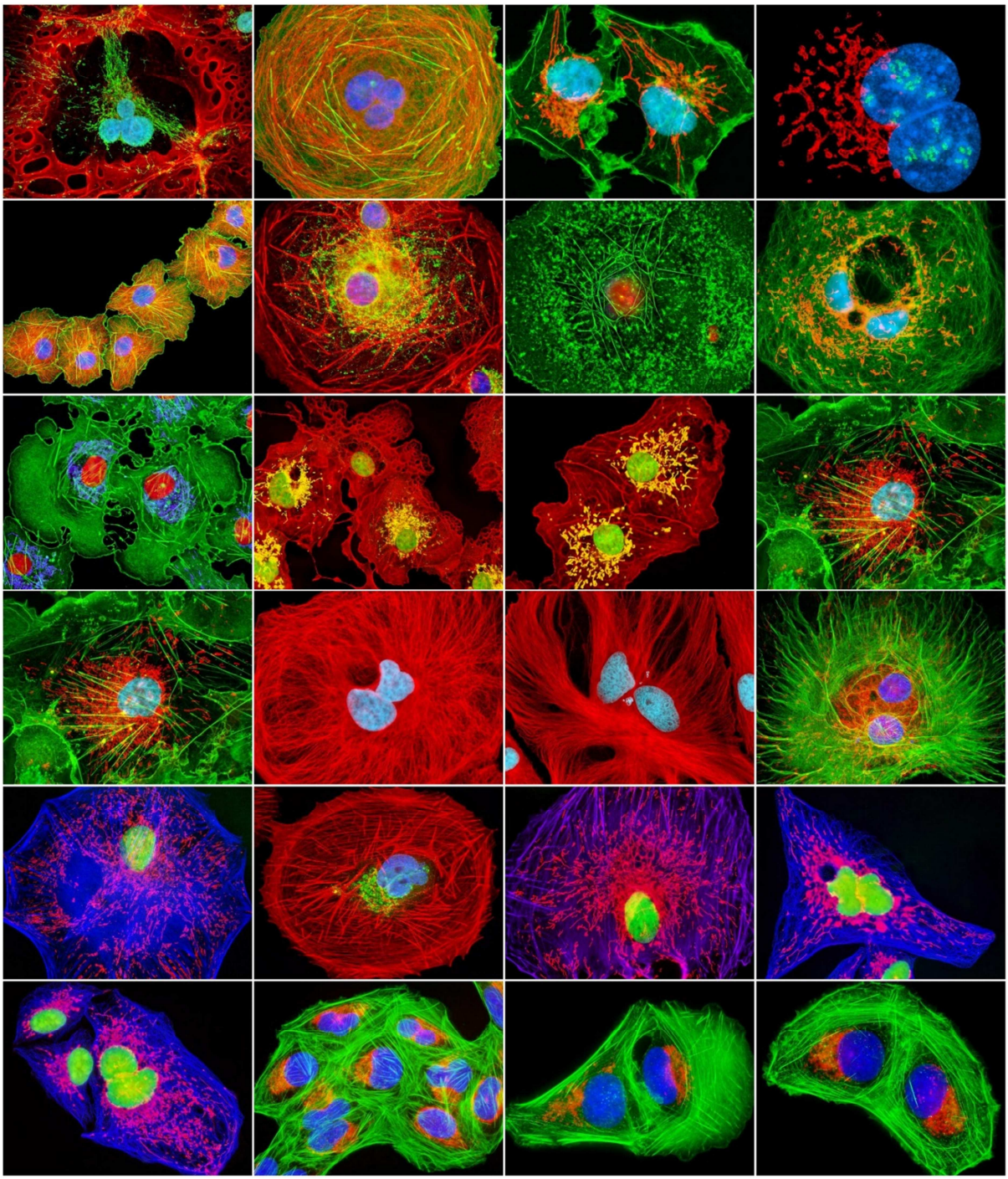


Figure 4.1: Nikon Digital Microscopy Test images: each image (700x504) is numbered in order of left-to- right and top-to-bottom, from 1 to 24..

The table 4.1 shows that average MSE over the set of images. The table shows that proposed fuzzy method (k) performs best on average in terms of MSE for all images.

Table 4.1: MSE Comparison (Nikon Digital Microscopy Images)

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
1.	7.32	5.05	7.83	10.43	4.63	9.34	11.90	5.33	6.49	6.40	**4.63**	6.02
2.	8.56	5.27	8.47	14.81	4.78	14.38	20.05	6.39	8.06	7.36	**4.81**	7.49
3.	3.27	1.76	4.60	4.16	1.62	5.97	7.72	2.19	2.75	2.35	**1.61**	2.43
4.	2.35	1.24	2.41	6.28	1.15	6.30	7.59	1.36	1.79	1.77	**1.17**	1.70
5.	5.55	4.42	5.44	8.57	4.05	8.78	9.65	4.79	5.28	5.04	**4.04**	4.97
6.	8.30	5.32	7.65	12.31	4.97	11.02	13.51	6.00	7.42	7.22	**4.89**	6.70
7.	6.35	4.01	7.81	3.88	3.81	5.02	8.58	3.76	4.70	4.09	**3.65**	3.73
8.	5.44	3.70	7.26	5.22	3.49	6.78	11.35	4.31	4.88	4.37	**3.43**	4.51
9.	6.77	4.41	8.99	5.71	4.05	7.35	11.90	4.92	5.90	5.26	**3.98**	5.07
10.	5.66	3.75	7.16	11.38	3.42	10.72	13.14	3.62	4.69	4.46	**3.40**	4.28
11.	3.54	2.20	4.05	6.55	1.97	5.88	7.50	2.08	2.74	2.57	**1.94**	2.48
12.	7.06	4.62	9.46	5.64	4.19	6.75	10.27	5.15	5.96	5.49	**4.19**	5.32
13.	7.06	4.62	9.46	5.64	4.19	6.75	10.27	5.15	5.96	5.49	**4.19**	5.32
14.	5.94	3.74	7.28	18.17	3.39	17.34	17.98	3.67	4.91	4.89	**3.44**	4.81
15.	6.22	4.14	7.62	15.02	3.77	13.71	15.79	4.02	5.02	4.87	**3.80**	4.88
16.	7.55	4.70	11.52	6.68	4.33	8.74	12.67	5.34	6.39	5.78	**4.30**	5.60
17.	7.77	5.27	5.47	14.01	4.85	12.29	14.98	7.21	8.08	7.56	**4.88**	7.86
18.	5.14	3.36	5.27	9.83	3.05	8.17	11.31	3.50	4.58	4.88	**3.10**	4.45
19.	7.74	5.18	11.40	11.15	4.69	8.35	12.78	5.87	6.84	6.27	**4.68**	6.67
20.	4.29	3.17	5.92	8.76	2.87	8.32	10.04	3.66	4.29	4.13	**2.92**	3.99
21.	5.47	3.98	6.26	10.03	3.63	9.58	10.70	4.82	5.39	5.27	**3.69**	5.11
22.	5.33	2.53	12.94	7.73	2.36	11.33	14.20	3.66	5.12	4.44	**2.31**	4.02
23.	2.19	1.07	2.26	2.54	1.00	5.15	7.22	1.18	1.53	1.32	**1.00**	1.21
24.	3.18	1.59	3.21	3.33	1.49	5.62	7.66	1.66	2.16	1.88	**1.46**	1.67

Table 4.2 reports the PSNR. The errors are reported for the same set of algorithms. These measures agree with the MSE comparison. The proposed fuzzy method (k) shows superior results to the other algorithms.

Table 4.2: PSNR Comparison (Nikon Digital Microscopy Images).

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
1.	40.18	40.00	40.32	39.87	42.04	39.70	38.97	41.63	41.01	41.29	**42.06**	41.37
2.	39.92	40.64	40.32	39.85	42.79	38.17	36.78	41.22	40.41	40.92	**42.81**	40.77
3.	43.16	44.17	42.61	42.90	46.39	40.47	39.59	44.91	44.00	44.78	**46.47**	44.60
4.	45.11	47.08	46.92	44.05	49.32	41.02	40.02	47.86	46.89	47.20	**49.42**	47.44
5.	41.09	40.50	41.21	40.39	42.58	39.52	39.25	41.87	41.46	41.75	**42.59**	41.76
6.	39.55	39.72	40.10	39.21	41.68	39.01	38.53	41.11	40.37	40.69	**41.79**	40.91
7.	40.49	40.56	39.36	42.33	42.55	42.24	41.29	42.51	41.63	42.13	**42.63**	42.47
8.	41.03	41.35	40.58	41.97	43.27	40.30	38.80	42.19	41.72	42.27	**43.44**	42.04
9.	39.89	40.09	39.12	40.80	42.15	39.67	38.28	41.23	40.45	40.98	**42.24**	41.14
10.	41.95	41.96	42.26	42.18	44.15	41.13	41.25	43.92	43.11	43.53	**44.21**	43.57
11.	44.08	44.00	44.26	44.34	46.38	43.21	43.19	46.31	45.30	45.91	**46.48**	45.95
12.	39.89	40.05	39.26	41.19	42.19	40.36	39.17	41.38	40.79	41.24	**42.20**	41.29
13.	39.89	40.05	39.26	41.19	42.19	40.36	39.17	41.38	40.79	41.24	**42.20**	41.29
14.	43.98	44.59	43.94	42.25	46.86	39.65	40.49	45.89	45.42	45.64	**46.77**	45.36
15.	42.67	43.21	42.82	41.46	45.47	39.47	39.21	44.99	44.62	45.00	**45.40**	44.66
16.	39.65	40.04	38.85	40.67	42.06	39.25	38.40	41.32	40.61	41.08	**42.11**	41.22
17.	40.45	40.92	42.69	40.11	42.79	39.54	38.22	39.80	39.41	39.86	**42.97**	39.54
18.	43.14	43.58	44.02	42.78	45.81	42.14	40.87	44.63	43.84	43.99	**45.82**	44.15
19.	41.78	41.94	41.56	41.45	44.13	40.70	39.26	42.60	42.20	42.64	**44.12**	42.32
20.	43.79	44.20	44.33	43.21	46.53	40.25	39.58	44.24	43.78	44.06	**46.48**	44.13
21.	42.61	43.08	43.62	42.27	45.22	39.26	38.85	42.85	42.48	42.73	**45.23**	42.79
22.	40.98	42.31	39.24	40.35	44.42	37.82	36.97	42.69	41.26	41.91	**44.52**	42.42
23.	44.94	45.85	45.02	44.87	48.15	41.95	40.88	47.47	46.37	46.95	**48.17**	47.34
24.	43.43	44.26	43.34	43.31	46.47	41.19	40.33	45.98	44.92	45.45	**46.52**	45.94

Table 4.3 shows the CPU time. The table shows that the (h) method is the fastest than all other algorithms. Since this method is implemented in MATLAB libraries, the method is supposed to be coded in optimal manner and is more close to the system platform. Since our proposed method is implemented at user level in a high level language and we have not performed any code optimizations, so the actual running time of the algorithm will be very

less. Our proposed fuzzy method is still faster than many of the other demosaicking algorithms as shown in table 4.3 and is also better than the method (h) in terms of MSE and PSNR as shown in table 4.2 and table 4.3.

Table 4.3: CPU Time Comparison in seconds (Nikon Digital Microscopy Images)

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
1.	0.02	0.12	3.79	10.64	1.00	2.32	26.99	0.01	295.30	4.41	2.09	5.65
2.	0.02	0.07	3.96	9.71	0.96	2.42	28.58	0.01	285.35	4.26	2.05	5.23
3.	0.02	0.06	3.79	9.11	1.02	1.98	24.39	0.01	290.98	4.26	1.98	5.07
4.	0.02	0.06	3.53	9.75	0.95	1.50	20.83	0.01	301.02	4.47	1.97	5.07
5.	0.02	0.07	3.54	10.22	0.90	1.54	20.60	0.01	337.59	4.53	1.96	5.07
6.	0.02	0.07	3.78	10.72	0.89	2.41	26.29	0.01	291.45	4.30	1.96	5.08
7.	0.02	0.07	3.67	9.98	0.91	2.12	26.13	0.01	277.54	4.34	1.97	5.07
8.	0.02	0.07	3.67	9.67	0.90	2.18	26.45	0.01	263.14	4.28	1.96	5.08
9.	0.02	0.07	3.72	9.56	0.92	2.16	25.87	0.01	269.06	4.30	1.96	5.05
10.	0.02	0.06	3.69	9.59	0.92	2.31	25.32	0.01	265.31	4.36	1.97	5.08
11.	0.02	0.07	3.54	9.16	0.89	1.67	21.37	0.01	265.16	4.39	1.97	5.07
12.	0.03	0.06	3.66	9.30	0.88	2.24	27.21	0.01	265.42	4.26	1.96	5.08
13.	0.02	0.06	3.69	9.63	0.89	2.19	26.90	0.01	266.48	4.29	1.96	5.08
14.	0.02	0.07	3.73	10.23	0.94	2.62	27.02	0.01	265.81	4.26	1.97	5.19
15.	0.02	0.07	3.74	9.98	0.90	2.54	27.06	0.01	266.83	4.26	1.96	5.08
16.	0.02	0.06	3.71	10.34	0.91	2.33	26.74	0.01	261.72	4.26	1.96	5.06
17.	0.02	0.07	3.75	10.19	0.88	2.23	27.18	0.01	268.00	4.29	1.98	5.18
18.	0.02	0.07	3.64	10.37	0.90	2.04	23.92	0.01	265.51	4.22	2.00	5.09
19.	0.02	0.07	3.80	10.68	0.89	2.20	25.38	0.01	266.98	4.31	1.98	5.10
20.	0.02	0.07	3.63	9.93	0.92	1.86	22.79	0.01	265.96	4.29	1.98	5.10
21.	0.02	0.07	3.62	9.32	0.95	1.83	23.57	0.01	265.09	4.27	1.95	5.06
22.	0.02	0.09	3.68	9.46	0.92	2.34	26.01	0.01	266.90	4.29	1.96	5.07
23.	0.02	0.07	3.59	9.70	0.89	1.65	21.55	0.01	266.34	4.30	1.98	5.07
24.	0.02	0.07	3.59	9.82	1.01	1.70	21.77	0.01	268.28	4.48	1.96	5.17

The numbers can only provide subset of the overall scenario. An important evaluation is the visual appearance of the output images. For this, an example image is presented. Figure 4.2 shows an image for which a small region is cropped and zoomed. This example includes a perspective that increases spatial frequency along the region. Aliasing is a prominent artifact in this image. The proposed interpolation algorithm (k) reconstructs this image very best. Very little aliasing is present in the output image. This is good example to show how the algorithm responds to features at various orientations. The algorithm (k) and the interpolation algorithm (l) show very few of the aliasing artifacts present in the other output images. This shows that these algorithms are fairly robust to the orientation of various features.

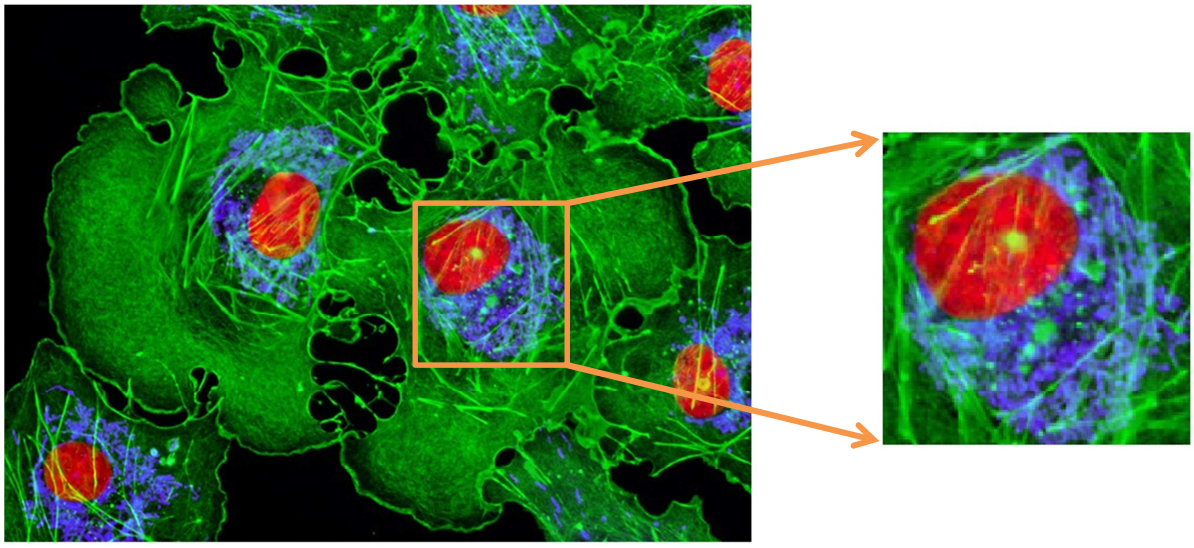


Figure 4.2: Cropped Region of the Original Image (Nikon Digital Microscopy Images).

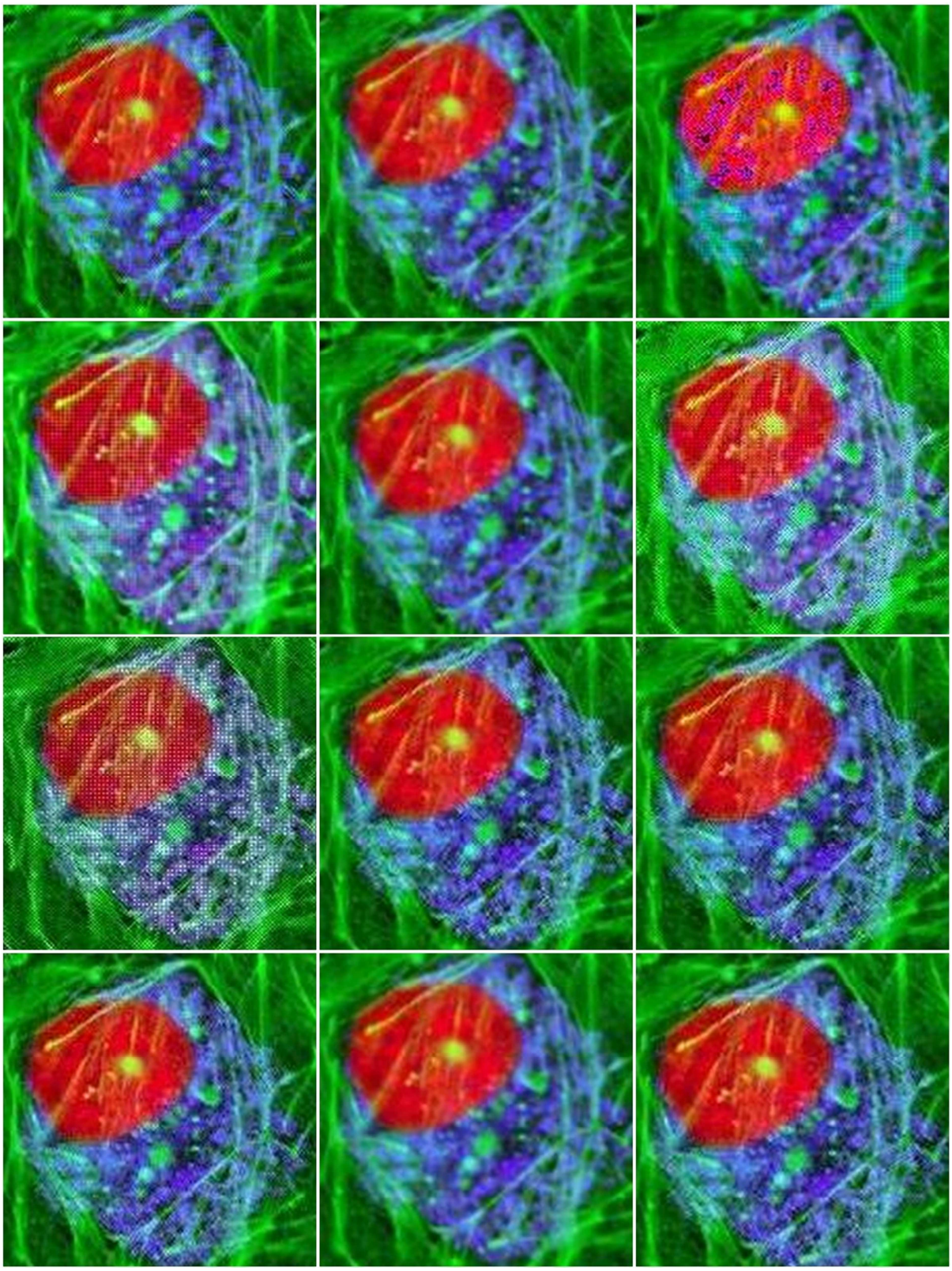


Figure 4.3: Visual Comparison (Nikon Digital Microscopy Images).

4.4.2 Satellite Color Images

The test set consists of 23 images with different pixel resolution (shown for each image in comparison table) as shown in figure 4.4. This image set has has been acquired from Landsat earth imaging [56]. The interpolated images are compared to the original images and results are reported for all three performance measures. The MSE results are summarized in table 4.4, PSNR results are summarized in table 4.5 and the CPU time results are summarized in table 4.6. The best result for each image is highlighted with bold text.

An image region which is cropped from the original image is presented in figure 4.5 for the original image visual quality comparison. This image region is compared with other algorithm in figure 4.6.

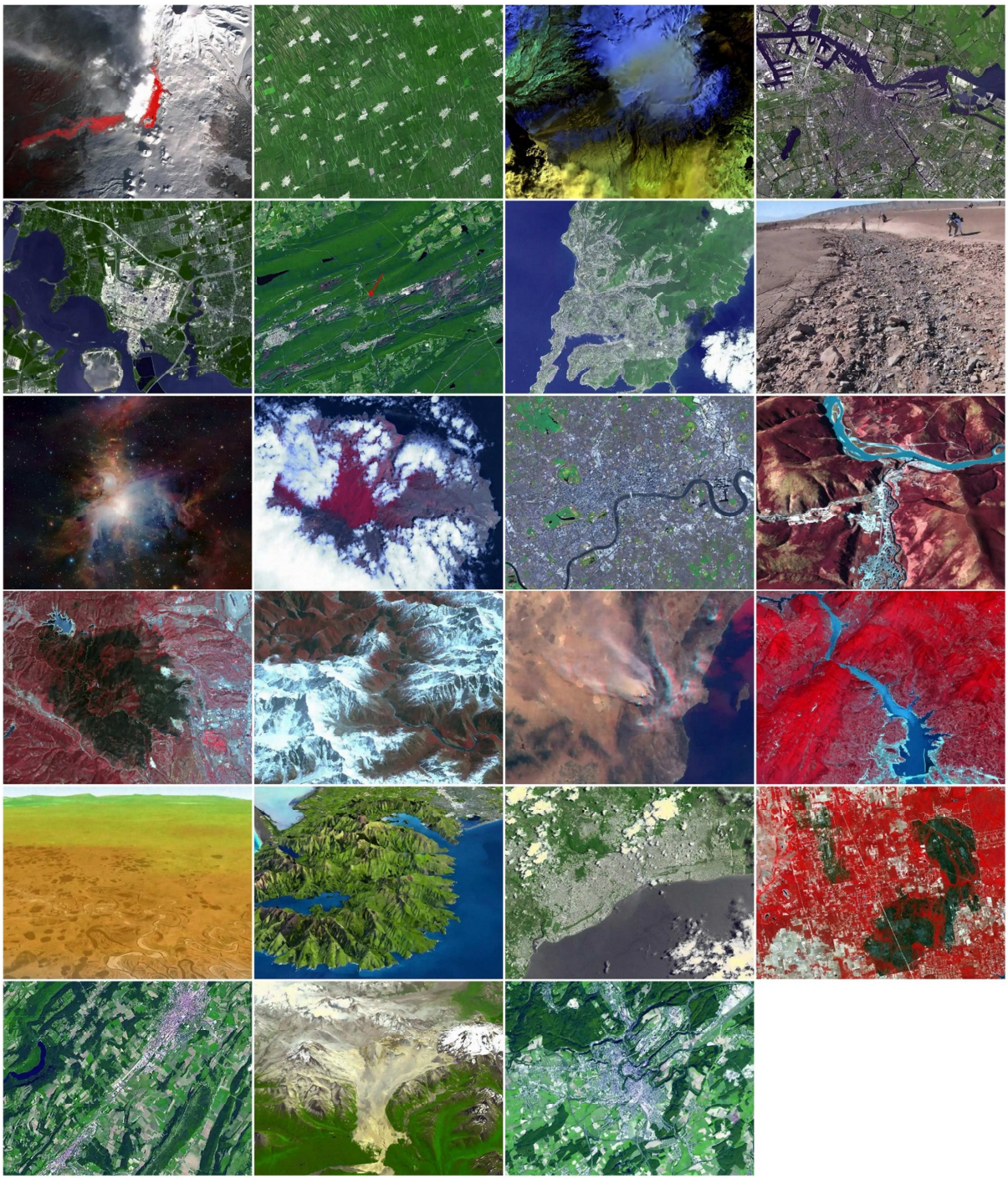


Figure 4.4: Satellite Color Images: each image is of different size and is numbered in order of left-to- right and top-to-bottom, from 1 to 23.

The table 4.4 shows that average MSE over the set of images. The table shows the method (j) performs best for 13 images and the proposed method (l) performs best on rest 10 images. However, the results of proposed algorithm are better than all other algorithms.

Table 4.4: MSE Comparison (Satellite Color Images)

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(j)	(k)	(l)
1.	4.86	2.55	4.78	15.48	2.40	14.50	14.74	1.00	**0.69**	2.43	0.86
2.	11.82	8.67	13.23	10.28	8.19	8.39	15.79	4.24	**3.16**	8.58	3.86
3.	5.60	2.82	9.85	11.95	2.71	10.81	14.03	1.41	**1.10**	2.73	1.35
4.	15.88	12.23	9.62	18.84	11.84	12.03	19.27	6.44	5.63	12.07	**5.60**
5.	9.99	6.51	5.83	11.62	6.18	8.53	11.21	2.84	**2.43**	6.25	2.70
6.	11.16	7.87	21.58	9.85	7.67	7.84	13.89	4.39	4.18	7.83	**4.05**
7.	8.11	6.22	6.35	16.49	6.00	12.81	18.18	3.35	**2.77**	6.08	2.86
8.	15.78	13.46	20.68	30.82	13.12	28.60	28.15	9.06	8.22	13.08	**7.69**
9.	2.74	1.77	5.87	4.37	1.81	2.94	4.06	1.47	1.20	1.84	**1.13**
10.	3.86	1.17	7.58	24.62	1.09	27.08	20.05	0.57	**0.48**	1.06	0.53
11.	16.44	12.79	17.25	25.39	12.62	17.77	28.06	7.45	7.11	12.77	**6.51**
12.	6.52	4.12	11.82	16.01	4.04	12.57	14.65	2.65	2.28	4.09	**2.27**
13.	9.16	5.09	10.88	15.87	4.87	10.43	15.25	2.41	**2.21**	4.99	2.37
14.	10.48	8.01	11.29	24.87	7.81	24.34	19.49	5.55	4.95	7.88	**4.64**
15.	1.90	0.70	17.19	19.87	0.64	15.25	22.82	0.39	0.34	0.57	**0.33**
16.	8.62	5.34	10.83	22.57	5.27	19.21	20.33	3.69	3.92	5.35	**3.51**
17.	4.14	1.54	32.41	23.30	1.39	31.45	26.75	0.55	0.58	1.40	**0.46**
18.	7.41	5.94	10.30	12.88	5.76	10.53	16.75	4.06	**3.43**	5.87	3.45
19.	15.15	12.36	8.30	24.65	11.94	21.48	22.31	7.28	**5.81**	12.30	6.23
20.	11.44	7.26	7.69	25.29	7.00	24.59	21.41	4.19	**3.82**	7.19	4.13
21.	15.16	11.62	13.76	18.34	11.32	16.12	18.51	6.50	**6.00**	11.37	6.09
22.	7.95	5.54	8.19	21.12	5.40	23.15	20.88	3.58	3.34	5.49	**2.99**
23.	13.82	9.57	12.69	20.18	9.22	20.96	18.80	4.86	**4.66**	9.20	4.67

Table 4.5 reports the PSNR. The errors are reported for the same set of algorithms. These measures agree with the MSE comparison. The method (j) and proposed algorithm (l) shows superior results to the other algorithms. Although the proposed method does not have the higher PSNR average for all images, its results are comparable to the latest demosaicking methods for the most part and it outperforms all other methods on a number of images.

Table 4.5: PSNR Comparison (Satellite Color Images).

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(j)	(k)	(l)
1.	41.33	44.29	42.51	38.66	44.82	36.54	36.45	48.35	**49.75**	44.53	49.18
2.	37.61	38.89	37.53	38.42	39.32	38.89	37.57	42.06	**43.18**	38.95	42.57
3.	40.79	43.87	41.56	39.32	44.25	37.84	36.72	46.89	**47.88**	44.06	47.24
4.	36.22	37.39	38.33	35.90	37.62	37.37	35.44	40.27	40.67	37.46	**40.93**
5.	38.25	40.16	40.67	38.31	40.59	38.86	37.72	43.81	**44.35**	40.37	44.26
6.	37.72	39.32	36.09	38.55	39.51	39.20	37.67	41.91	41.98	39.36	**42.39**
7.	39.14	40.34	41.96	37.24	40.58	37.09	36.01	43.11	43.75	40.45	**43.82**
8.	36.24	36.95	35.83	34.18	37.12	33.64	33.69	38.73	39.04	37.09	**39.40**
9.	43.87	45.88	41.98	42.56	45.75	43.54	42.07	46.75	47.55	45.72	**47.97**
10.	42.38	47.81	42.46	38.46	48.47	33.91	35.65	50.85	51.60	48.32	**51.61**
11.	36.07	37.20	36.98	34.82	37.29	35.70	33.82	39.62	39.68	37.22	**40.25**
12.	40.08	42.22	38.78	38.28	42.41	37.40	36.99	44.17	44.69	42.28	**44.92**
13.	38.61	41.27	38.73	37.62	41.62	38.01	36.34	44.63	**44.97**	41.39	44.81
14.	38.04	39.23	38.23	35.51	39.39	34.33	35.49	40.85	41.24	39.30	**41.62**
15.	45.61	49.94	40.99	40.28	50.91	36.63	34.72	52.51	53.37	51.02	**53.49**
16.	38.87	41.03	38.62	36.72	41.14	36.86	36.61	42.71	42.46	41.03	**42.99**
17.	41.99	46.59	37.28	40.37	47.90	34.93	36.83	51.04	50.65	47.16	**52.39**
18.	39.55	40.55	38.77	38.12	40.74	38.01	36.21	42.24	42.83	40.61	**42.99**
19.	36.47	37.34	39.01	35.03	37.59	34.83	34.75	39.73	**40.52**	37.37	40.41
20.	37.67	39.70	40.47	36.00	40.00	35.10	36.10	42.24	**42.63**	39.77	42.38
21.	36.42	37.61	37.27	36.04	37.79	36.06	35.78	40.16	40.40	37.72	**40.50**
22.	39.20	40.84	39.51	36.48	41.01	34.52	35.22	42.75	42.94	40.89	**43.55**
23.	36.81	38.48	37.66	36.02	38.74	34.96	35.92	41.46	41.55	38.67	**41.80**

Table 4.6 shows the CPU time. The table shows that the method (h) is the fastest than all other algorithms, the reason being the same as explained for Nikon Digital Microscopy Images. However, our proposed algorithms is still faster than many of the other demosaicking algorithms as shown in table 4.6.

Table 4.6: CPU Time Comparison in seconds (Satellite Color Images).

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(j)	(k)	(l)
1.	0.22	0.98	40.34	120.04	8.94	26.74	275.69	0.12	36.69	30.78	49.94
2.	0.09	0.56	19.79	71.63	5.28	13.42	157.63	0.07	24.99	17.84	29.96
3.	0.07	0.40	13.97	50.04	3.77	10.34	110.69	0.05	17.50	12.65	21.34
4.	0.04	0.20	7.42	25.94	1.91	5.26	58.04	0.03	8.86	6.52	11.05
5.	0.02	0.14	4.95	17.79	1.30	3.23	36.77	0.02	5.67	4.21	7.21
6.	0.06	0.38	13.12	47.28	3.38	8.42	99.95	0.05	15.52	11.40	19.75
7.	0.08	0.43	14.97	52.04	3.99	11.52	116.86	0.05	17.66	13.49	21.95
8.	0.03	0.21	7.34	24.64	1.88	7.53	61.09	0.03	8.06	6.26	10.57
9.	0.10	0.58	20.85	72.83	8.19	11.12	144.13	0.07	23.78	18.08	30.39
10.	0.09	0.53	19.36	68.29	7.85	18.96	157.84	0.07	22.75	17.29	29.41
11.	0.05	0.36	11.18	40.87	4.41	9.47	93.96	0.04	12.83	9.87	16.32
12.	0.12	0.67	23.31	89.79	9.39	17.66	186.39	0.08	27.66	21.48	33.95
13.	0.06	0.34	12.37	44.92	4.84	9.11	97.87	0.04	14.73	10.89	17.61
14.	0.07	0.45	15.59	56.82	6.10	14.54	139.45	0.06	18.64	13.59	22.23
15.	0.04	0.25	8.43	30.00	3.32	6.81	70.96	0.03	10.23	7.32	12.04
16.	0.11	0.55	19.09	67.09	7.58	15.63	164.46	0.07	23.11	17.19	27.56
17.	0.06	0.32	11.26	40.52	4.31	12.31	96.60	0.04	13.34	10.24	15.70
18.	0.06	0.35	11.96	44.59	4.60	7.79	90.95	0.07	15.28	12.06	17.86
19.	0.19	0.88	32.86	125.17	12.99	29.20	296.63	0.12	43.15	31.89	49.88
20.	0.11	0.60	22.04	77.61	8.47	19.11	198.44	0.08	28.38	21.01	32.14
21.	0.03	0.17	5.86	19.61	2.28	4.65	51.22	0.02	6.53	5.36	8.12
22.	0.10	0.60	23.17	77.17	8.94	20.27	199.77	0.09	25.35	21.06	32.35
23.	0.04	0.23	8.48	27.77	3.18	7.34	75.30	0.03	9.65	7.74	11.96

Now we will evaluate the visual appearance of the output images. For this, an image 7 is presented for which a small region is cropped and zoomed as shown in figure 4.5. This image includes a “island” from a perspective that increases spatial frequency along the region. Aliasing is a prominent artifect in this image. The proposed interpolation algorithm (l) reconstructs this image very best. Very little aliasing is present in the output image. This is good example to show how the algorithm respond to features at various orientations. The (l) algorithm and the (j) interpolation algorithm show very few of the aliasing artifacts present in the other output images. This shows that these algorithms are fairly robust to the orientation of various features.

Figure 4.5: Cropped Region of the Original Image (Satellite Images).

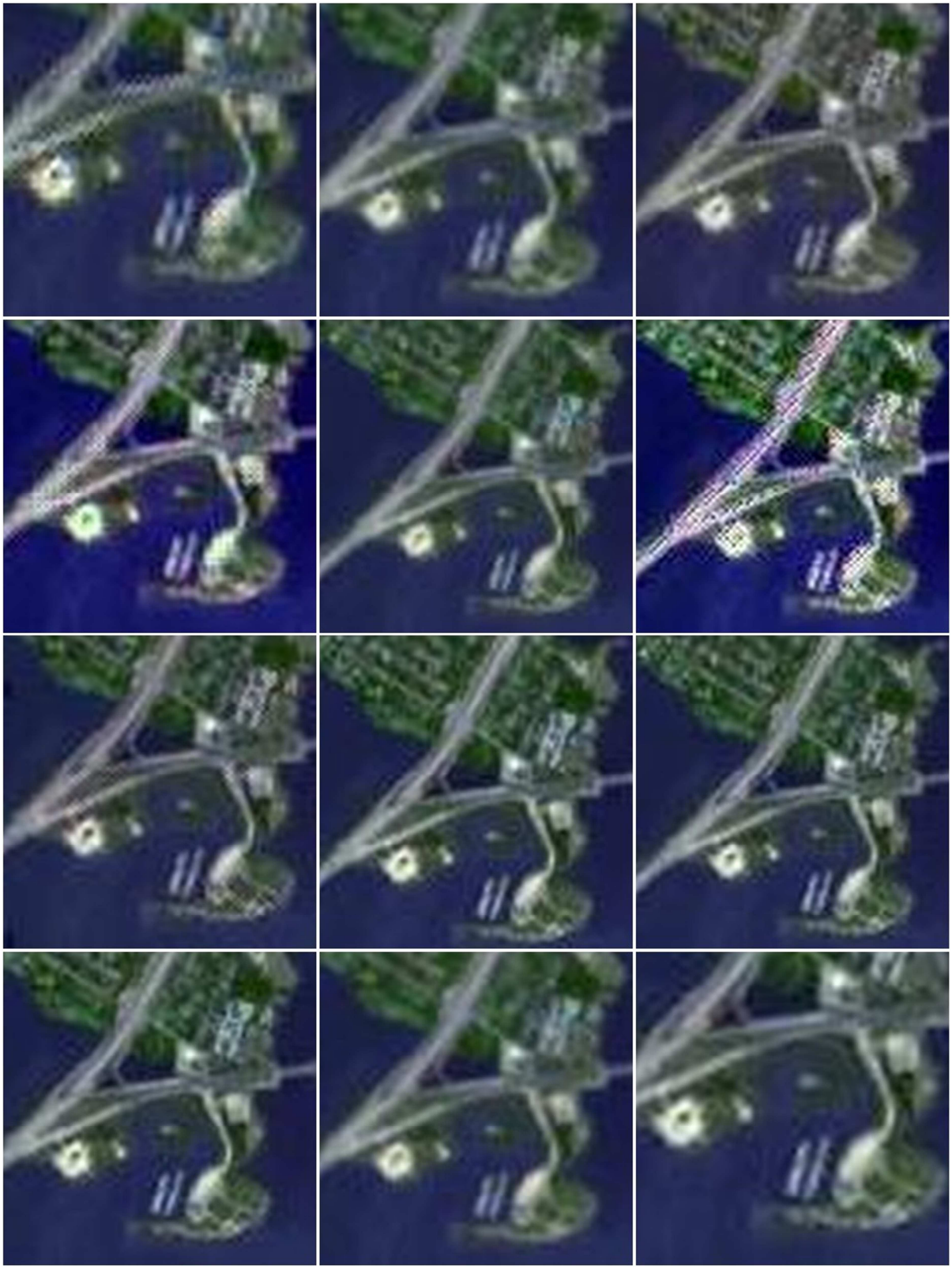


Figure 4.6: Zoomed Images for Visual Comparison (Satellite Color Images)

4.4.3 High Definition Color Images

The test set consists of 5 images with different pixel resolution (shown for each image in comparison table) as shown in figure 4.7. This image set contains random images that have been taken from internet. These images are captured using a high quality digital camera because resolution of these images is higher than a standard size image. The interpolated images are compared to the original images and results are reported for all three performance measures. The MSE results are summarized in table 4.7, PSNR results are summarized in table 4.8 and the CPU time results are summarized in table 4.9. The best result for each image is highlighted with bold text.

An image region which is cropped from the original image is presented in figure 4.8 for the visual quality comparison. This image region is compared with other algorithm in figure 4.9.

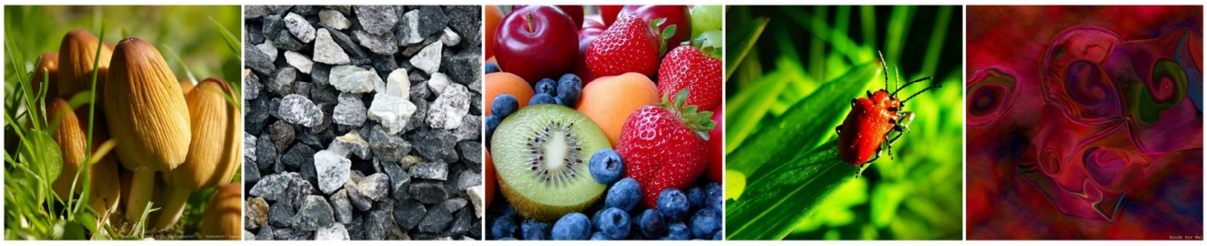


Figure 4.7: High Definition Color images: each image is of different size and is numbered in order of left-to- right and top-to-bottom, from 1 to 5.

The table 4.7 shows that average MSE over the set of images. The table shows the proposed method (l) performs best on average in terms of MSE except of one image i.e. 1 in which method (j) performs best.

Table 4.7: MSE Comparison (HD Color Images)

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(j)	(k)	(l)
1.	1.47	0.63	11.61	10.37	0.57	11.93	15.71	0.40	**0.28**	0.57	0.32
2.	1.15	0.58	3.17	6.58	0.57	3.43	6.58	0.78	0.96	0.52	**0.77**
3.	9.10	6.17	9.53	22.43	5.97	22.00	19.00	3.98	3.95	5.89	**3.65**
4.	16.97	13.62	9.93	24.39	13.35	21.81	18.75	8.07	7.43	13.26	**6.59**
5.	1.09	0.64	9.19	3.74	0.61	7.05	9.37	0.53	0.49	0.61	**0.44**

Table 4.8 reports the PSNR. The errors are reported for the same set of algorithms. These measures agree with the MSE comparison. The proposed method (l) and method (j) shows superior results to the other algorithms.

Table 4.8: PSNR Comparison (HD Color Images).

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(j)	(k)	(l)
1.	46.69	50.35	41.45	43.20	51.13	39.06	40.10	52.31	**54.02**	50.92	53.56
2.	48.02	51.60	46.48	47.31	51.96	49.56	45.17	49.60	49.36	51.93	**50.00**
3.	38.66	40.38	38.87	36.28	40.62	35.04	35.75	42.38	42.34	40.61	**42.79**
4.	35.92	36.91	38.27	34.90	37.04	34.81	35.41	39.23	39.46	37.04	**40.09**
5.	47.81	50.28	42.38	46.68	50.59	42.23	42.11	51.17	51.50	50.54	**52.05**

Table 4.9 shows the CPU time. The table shows that the (h) method is the fastest than all other algorithms, the reason being same as explained in previous sections. Our proposed algorithms is still faster than many of the other demosaicking algorithms as shown in table 4.9.

Table 4.9: CPU Time Comparison in seconds (HD Color Images).

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(j)	(k)	(l)
1.	0.23	0.77	43.18	124.77	12.48	34.29	351.95	0.22	58.12	26.82	65.73
2.	0.07	0.17	8.25	24.53	2.29	4.67	64.24	0.03	10.32	5.36	12.59
3.	0.03	0.04	1.97	5.46	0.52	1.78	17.53	0.01	2.41	1.12	2.91
4.	0.02	0.03	1.69	4.60	0.43	1.47	14.67	0.00	1.93	0.94	2.46
5.	0.09	0.40	18.38	52.61	5.13	13.36	145.94	0.05	23.25	11.21	29.70

Now we will evaluate the visual appearance of the output images. For this, an example image is presented. Figure 4.8 shows that “fruits” image out of which a region is cropped and zoomed to evaluate visual quality of interpolation algorithm. This cropped region includes a frequency transition from a perspective that increases spatial frequency along the region. Aliasing is a prominent artifect in this image. The proposed interpolation algorithm (l) reconstructs this image very best. Very little aliasing is present in the output image. The algorithm (l) and the (j) interpolation algorithm show very few of the aliasing artifacts present in the other output images. This shows that these algorithms are fairly robust to the orientation of various features.

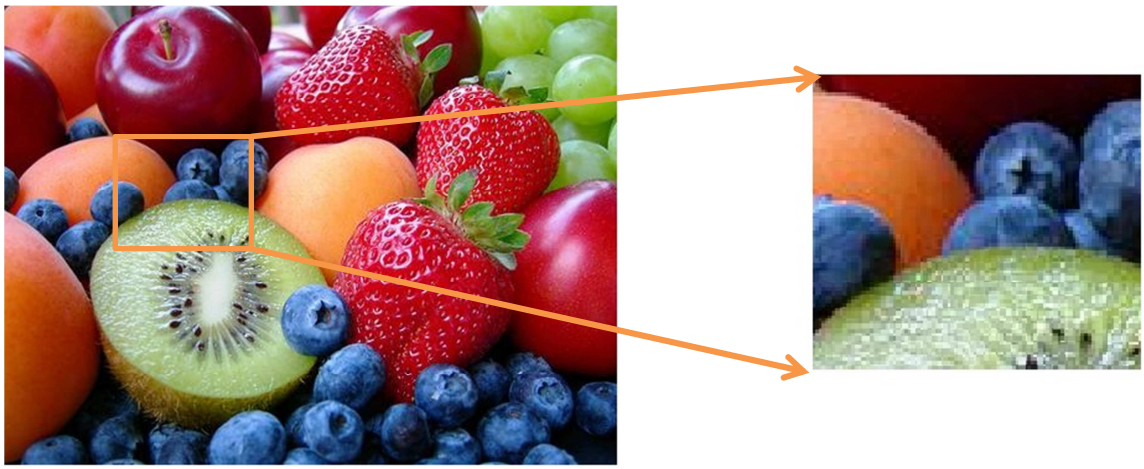


Figure 4.8: Cropped Region of the Original Image (HD Color Images).

Figure 4.9: Zoomed Images for Visual Comparison (HD Color Images)

4.4.4 Kodak Loss-Less Color Image suite

The test set consists of 24 images with 512x712 pixel resolution as shown in figure 4.10. This image set is released by Kodak [57] used for comparing the quality of color interpolation in recent survey paper [58]. The interpolated images are compared to the original images and results are reported for all three performance measures. The MSE results are summarized in table 4.10, PSNR results are summarized in table 4.11 and the CPU time results are summarized in table 4.12. The best result for each image is highlighted with bold text.

An image region which is cropped from the original image is presented in figure 4.11 for the original image visual quality comparison. This image region is compared with other algorithm in figure 4.12.

Figure 4.10: Test images: each image (768x512) is numbered in order of left-to- right and top-to-bottom, from 1 to 24.

The table 4.10 shows that average MSE over the set of images. The table shows the method (j) performs best on average in terms of MSE except of three images in which proposed method (l) performs best. However our proposed algorithm (l) is superior than all other algorithms except (j).

Table 4.10: MSE Comparison (Kodak Loss-Less True Color Images)

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
1.	12.09	10.86	6.41	20.86	8.99	15.84	21.69	5.49	4.52	**3.33**	9.37	4.25
2.	4.89	3.73	10.34	19.26	3.03	17.28	16.82	1.74	1.68	**1.42**	3.11	1.48
3.	3.80	2.96	10.40	15.18	2.26	11.80	18.19	1.21	1.03	0.95	2.27	**0.95**
4.	5.49	3.53	14.23	19.04	3.12	13.95	19.35	1.66	1.51	**1.33**	3.18	1.40
5.	10.61	8.79	10.39	14.14	7.06	9.55	12.72	3.85	2.94	**2.52**	7.28	3.10
6.	9.38	7.85	3.98	22.67	6.28	22.39	21.10	4.03	3.20	**2.46**	6.73	3.31
7.	5.18	3.36	6.27	16.73	2.38	12.69	21.32	1.29	1.01	**0.79**	2.35	1.00
8.	11.72	11.67	13.08	23.54	8.96	20.96	20.52	7.21	5.87	**3.51**	9.74	5.99
9.	4.73	3.42	3.15	25.26	2.90	22.58	27.83	1.86	1.42	**0.91**	3.00	1.43
10.	4.64	3.24	2.16	21.36	2.82	16.54	25.05	1.67	1.31	**0.90**	2.89	1.29
11.	7.34	5.92	4.98	13.85	4.76	7.87	18.04	2.87	2.49	**1.98**	5.00	2.34
12.	4.50	3.52	2.07	31.92	2.65	37.16	24.42	1.60	1.21	**0.98**	2.71	1.19
13.	14.44	12.04	9.48	20.33	10.89	15.78	18.69	6.96	5.84	5.80	11.12	**5.53**
14.	9.06	6.82	8.80	15.22	5.47	12.06	15.56	2.94	2.58	**2.21**	5.67	2.48
15.	4.56	3.75	6.91	18.61	2.96	20.23	13.98	1.92	1.82	**1.29**	2.99	1.59
16.	6.55	5.08	3.23	16.28	3.97	12.13	19.53	2.31	1.96	**1.14**	4.23	1.92
17.	5.14	3.43	4.72	11.85	3.04	8.31	12.60	1.75	1.30	**1.08**	3.03	1.30
18.	9.36	6.38	9.46	10.14	5.99	6.29	8.84	3.56	2.88	2.88	6.15	**2.87**
19.	7.37	6.27	10.81	19.72	5.25	15.41	22.42	3.61	2.85	**1.84**	5.61	2.85
20.	4.36	3.71	4.89	28.95	2.92	34.99	24.82	1.87	2.77	**1.23**	2.96	1.44
21.	7.33	6.21	11.01	23.46	5.13	19.44	23.73	3.16	2.64	**2.25**	5.29	2.50
22.	6.74	5.18	7.69	19.34	4.19	16.03	20.98	2.43	2.12	**1.85**	4.32	2.01
23.	2.77	2.15	8.87	14.88	1.54	14.96	16.91	0.92	0.76	**0.61**	1.49	0.80
24.	7.88	6.95	6.56	17.12	5.81	12.84	18.33	3.54	3.32	**2.73**	5.93	2.94

Table 4.11 reports the PSNR. The errors are reported for the same set of algorithms. These measures agree with the MSE comparison. The method (j) and proposed algorithm (l) shows superior results to the other algorithms. Although the proposed method does not have the

higher PSNR average, its results are comparable to the latest demosaicking methods for the most part and it outperforms all other methods on a number of images.

Table 4.11: PSNR Comparison (Kodak Loss-Less True Color Images).

Image Number	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
1.	37.38	37.95	41.82	36.15	38.95	36.25	35.19	41.03	41.58	42.95	38.57	42.11
2.	41.43	42.66	39.58	39.70	43.84	40.23	41.77	46.05	45.88	47.08	43.47	46.75
3.	42.42	43.63	39.69	39.01	45.08	37.76	35.90	47.60	48.02	48.48	44.77	**48.48**
4.	40.89	42.92	38.83	37.85	43.57	37.12	35.86	46.08	46.33	**46.98**	43.30	**46.89**
5.	37.98	38.86	38.63	37.53	40.00	38.39	37.38	42.53	43.44	44.19	39.65	43.47
6.	38.44	39.37	44.04	36.30	40.62	34.67	35.38	42.38	43.08	44.28	40.00	43.29
7.	41.04	43.13	41.06	38.76	45.02	37.51	35.38	47.29	48.09	49.24	44.65	48.39
8.	37.58	37.62	37.35	35.86	39.14	34.96	35.06	39.79	40.44	42.76	38.37	40.60
9.	41.47	43.03	43.38	37.17	44.10	34.61	33.72	45.71	46.60	48.62	43.54	46.94
10.	41.56	43.28	44.99	37.64	44.17	35.96	34.27	46.17	46.95	48.66	43.74	47.38
11.	39.55	40.62	42.00	38.23	41.86	39.19	35.74	43.85	44.17	45.28	41.35	44.77
12.	41.65	42.95	45.71	36.85	44.65	32.52	34.70	46.42	47.30	48.32	44.07	47.76
13.	36.62	37.46	38.58	35.72	37.95	36.18	35.74	39.91	40.46	40.53	37.80	**40.88**
14.	38.64	40.01	39.21	37.71	41.17	37.56	36.72	43.72	44.01	44.83	40.79	44.47
15.	41.74	42.59	41.17	37.86	43.81	35.20	36.84	45.53	45.53	47.22	43.56	46.37
16.	40.00	41.30	43.08	38.03	42.74	37.33	35.35	44.84	45.21	47.62	42.07	45.70
17.	41.13	42.99	43.43	39.08	43.70	38.95	37.24	45.91	47.00	47.85	43.48	47.31
18.	38.52	40.27	38.91	38.80	40.60	40.30	39.40	42.82	43.54	43.59	40.39	**43.77**
19.	39.59	40.34	38.78	36.97	41.39	36.33	34.76	42.78	43.58	45.55	40.76	43.78
20.	41.89	42.64	42.67	36.60	43.89	32.76	34.35	45.68	43.70	47.33	43.60	46.86
21.	39.56	40.37	38.48	36.39	41.35	35.35	34.45	43.39	43.91	44.67	41.05	44.42
22.	39.98	41.19	40.13	37.25	42.25	36.12	35.18	44.53	44.87	**45.56**	41.96	**45.40**
23.	43.92	45.03	40.56	39.70	46.79	36.60	36.30	48.75	49.32	50.51	46.62	49.36
24.	39.31	39.90	41.94	37.25	40.80	37.07	35.71	42.89	42.91	**43.84**	40.56	**43.72**

Table 4.12 shows the CPU time. The table shows that the method (h) is the fastest than all other algorithms, the reason being same as explained in previous sections. Our proposed algorithms is still faster than many of the other demosaicking algorithms as shown in table 4.12.

Table 4.12: CPU Time Comparison in seconds (Kodak Loss-Less True Color Images).

S.No.	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
1.	0.04	0.12	4.59	11.15	1.58	4.61	47.77	0.17	319.18	6.95	3.50	8.94
2.	0.04	0.13	4.54	11.27	1.65	3.83	45.94	0.02	316.10	6.95	3.36	8.77
3.	0.04	0.12	4.98	11.21	1.62	4.26	43.87	0.01	348.52	7.01	3.34	8.76
4.	0.04	0.12	5.20	10.97	1.66	4.15	45.67	0.01	340.19	7.20	3.34	8.72
5.	0.04	0.12	4.56	10.64	1.55	3.61	44.93	0.01	388.67	7.28	3.34	8.74
6.	0.04	0.17	4.72	10.42	1.56	5.16	47.97	0.01	4720.38	6.97	3.34	8.73
7.	0.04	0.14	4.55	11.30	1.75	4.71	46.13	0.01	342.99	7.28	3.35	8.75
8.	0.04	0.13	4.76	10.79	1.64	4.57	52.63	0.02	339.32	6.80	3.37	8.73
9.	0.04	0.13	4.79	10.67	1.54	5.81	50.04	0.01	368.97	6.98	3.82	8.73
10.	0.04	0.13	5.06	10.63	1.56	4.94	48.24	0.01	3731.79	7.23	3.34	8.75
11.	0.04	0.11	4.77	10.43	1.65	3.73	44.90	0.01	390.18	7.08	3.37	8.73
12.	0.04	0.12	4.84	10.49	1.62	6.62	50.38	0.02	3106.09	6.91	3.35	8.72
13.	0.05	0.13	4.40	10.46	1.55	4.10	46.91	0.01	390.23	6.84	3.37	8.73
14.	0.04	0.12	4.35	10.34	1.56	4.04	46.70	0.02	517.56	6.78	3.38	8.73
15.	0.04	0.14	4.33	10.64	1.61	4.49	44.35	0.01	476.25	6.94	3.42	8.74
16.	0.05	0.12	4.37	10.66	1.59	4.09	48.11	0.01	488.63	6.79	3.42	8.74
17.	0.04	0.12	4.53	10.10	1.56	3.74	44.21	0.01	494.59	6.78	3.43	8.73
18.	0.03	0.15	4.53	10.24	1.54	3.30	48.35	0.01	517.31	6.77	3.37	8.73
19.	0.03	0.12	4.54	10.13	1.55	4.34	50.47	0.02	437.75	6.86	3.36	8.92
20.	0.03	0.12	4.44	9.92	1.59	4.64	42.73	0.02	451.81	6.92	3.41	8.73
21.	0.04	0.12	4.37	10.33	1.60	4.49	53.77	0.02	418.04	6.80	3.41	8.95
22.	0.03	0.13	4.45	10.30	1.61	4.32	57.12	0.02	399.83	6.86	3.45	8.76
23.	0.03	0.15	4.41	10.37	1.83	4.25	54.34	0.02	413.73	6.93	3.40	8.85
24.	0.04	0.12	4.34	10.37	1.84	4.02	55.98	0.01	389.89	7.06	3.41	8.79

An important evaluation is the visual appearance of the output images. For this, an example image is presented. Figure 4.11 shows the “hills” image. This example includes a “zebra kind of pattern” from a perspective that increases spatial frequency along the region. Aliasing is a prominent artifect in this image. The (j) interpolation algorithm reconstructs this image very best. Very little aliasing is present in the output image. The image contains various lines at vaious angles across the image. This is good example to show how the algorithm respond to features at various orientations. The proposed algorithm (k) and (l) show very few of the aliasing artifacts present in the other output images. This shows that these algorithms are fairly robust to the orientation of various features.

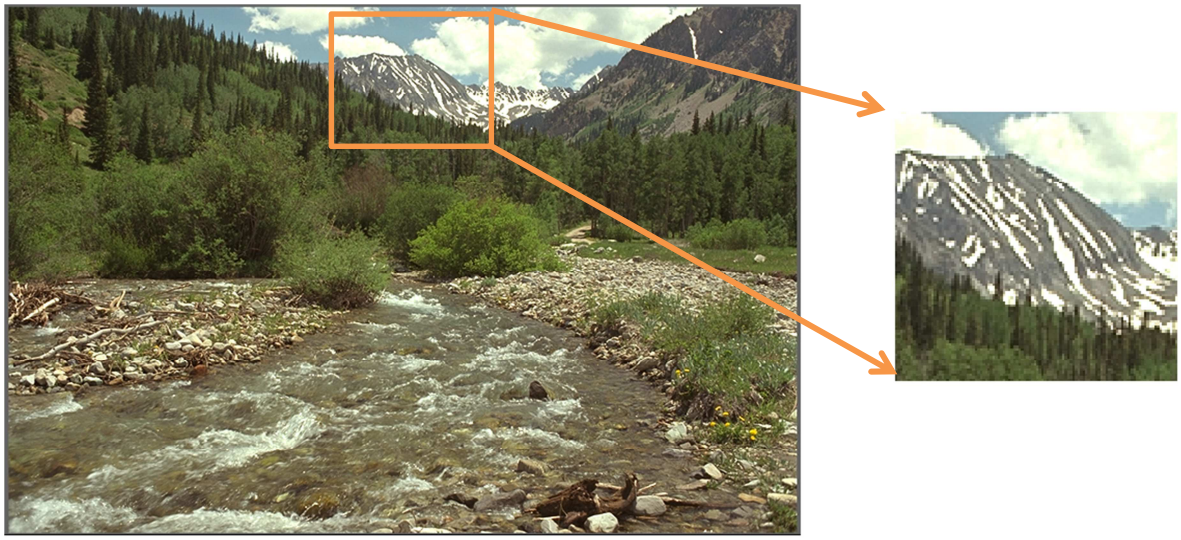


Figure 4.11: Cropped Region of the Original Image (Kodak Loss-Less True Color Images).

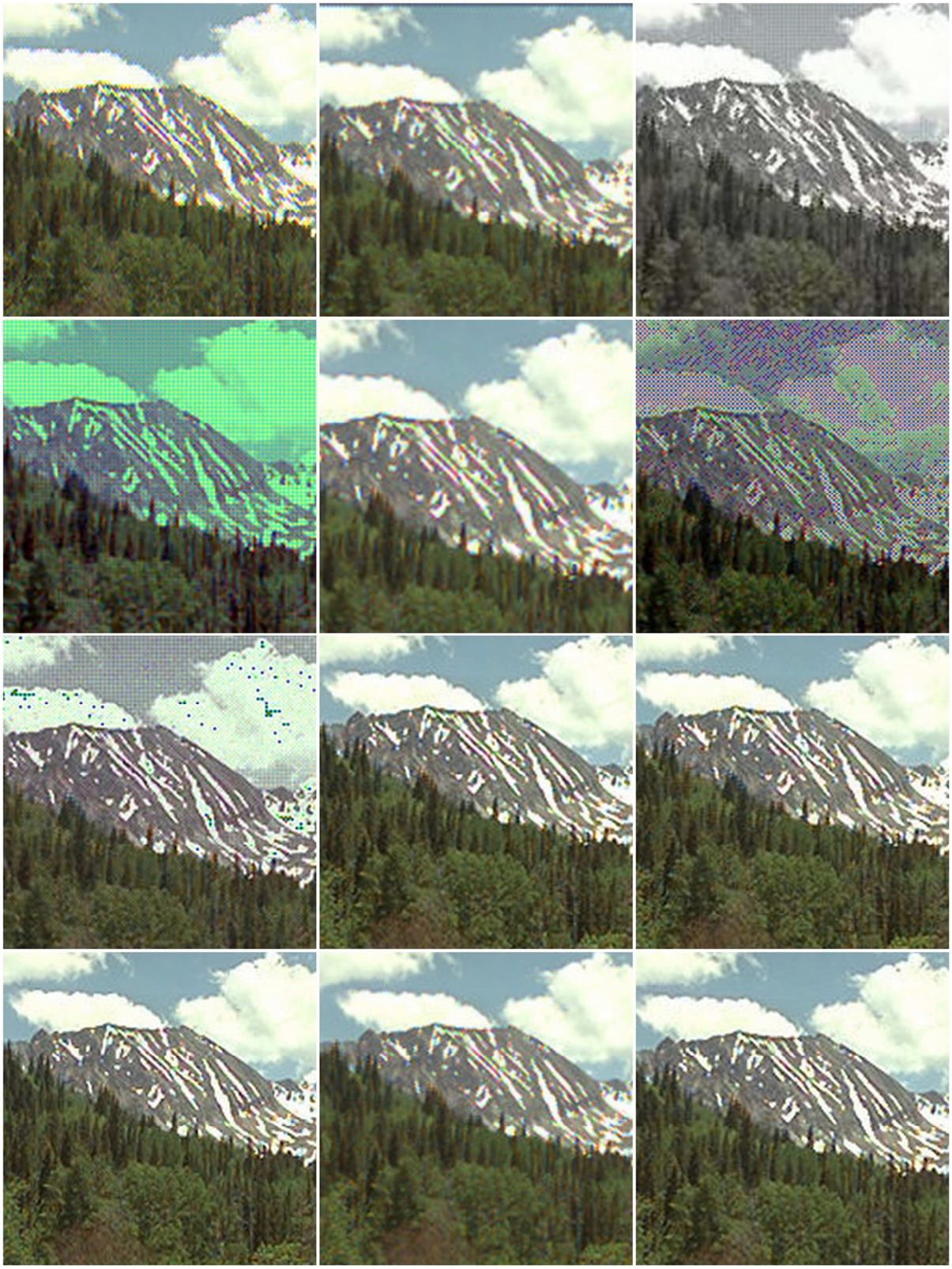


Figure 4.12: Zoomed Images for Visual Comparison (Kodak Loss-Less True Color Images)

4.4.5 Berkeley Image database

The test set consists of 100 images with 481x321 pixel resolution as shown in figure 4.13. This image set has been used for segmentation purposes [60]-[62]. This image database has been acquired from [59]. The interpolated images are compared to the original images and results are reported for all three performance measures. The MSE results are compared with other algorithms using a histogram. The X-axis of the histogram indicates the image number and Y-axis of the histogram shows the MSE value for the corresponding image. This is shown in figure 4.14. Similarly, PSNR results are shown in figure 4.15 and the CPU time results are shown in figure 4.16. The results for the proposed algorithms are highlighted with bold lines.

An image region which is cropped from the original image is presented in figure 4.17 for the original image visual quality comparison. This image region is compared with other algorithm in figure 4.18.

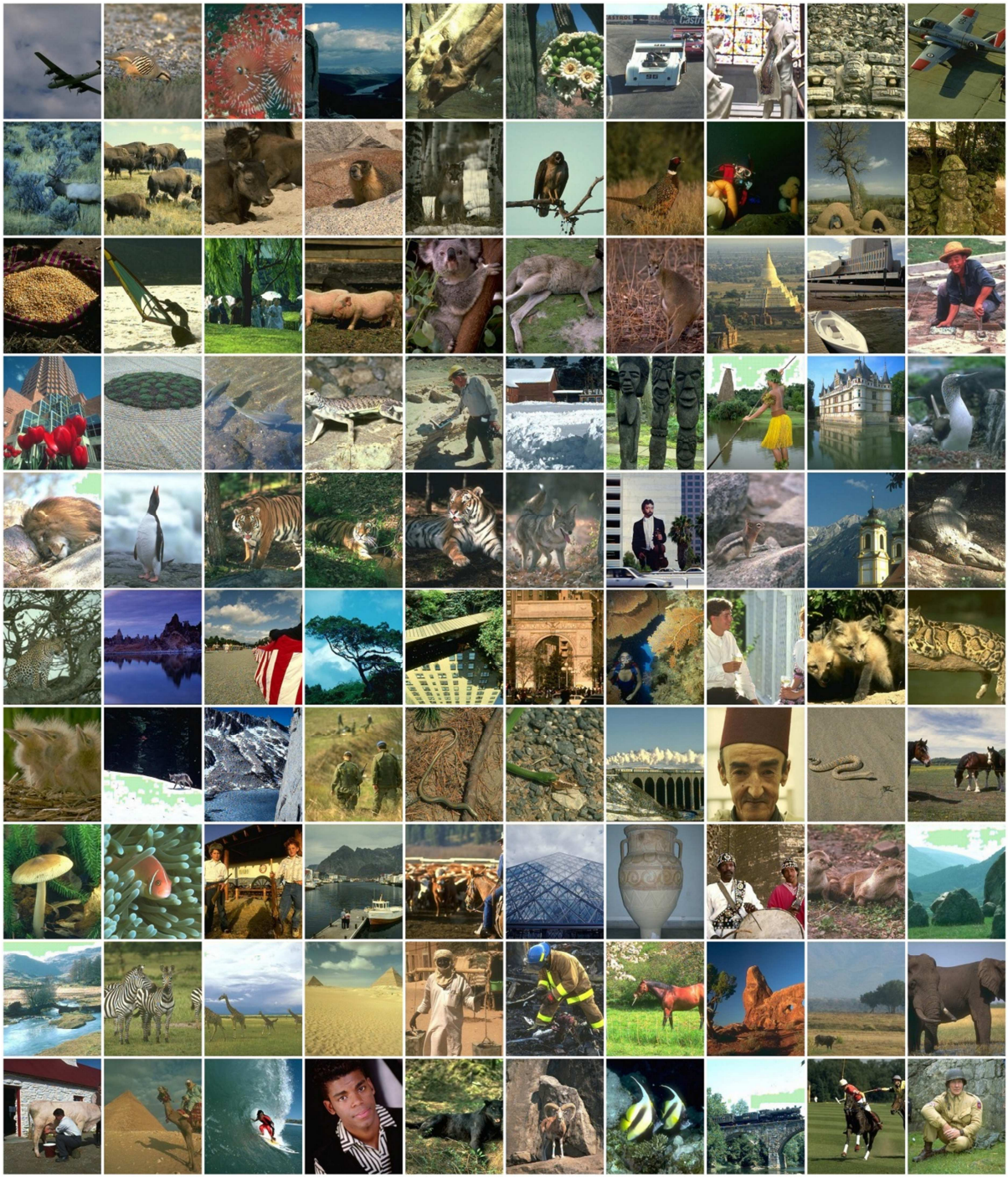


Figure 4.13: Berkeley Test images: each image (481x321) is numbered in order of left-to-right and top-to-bottom, from 1 to 100.

The histogram in figure 4.14 shows that average MSE over the set of images. The results of the proposed method are highlighted using bold lines. This histogram shows the proposed (l) method performs best on average in terms of MSE except of some of the images in which method (j) performs best.

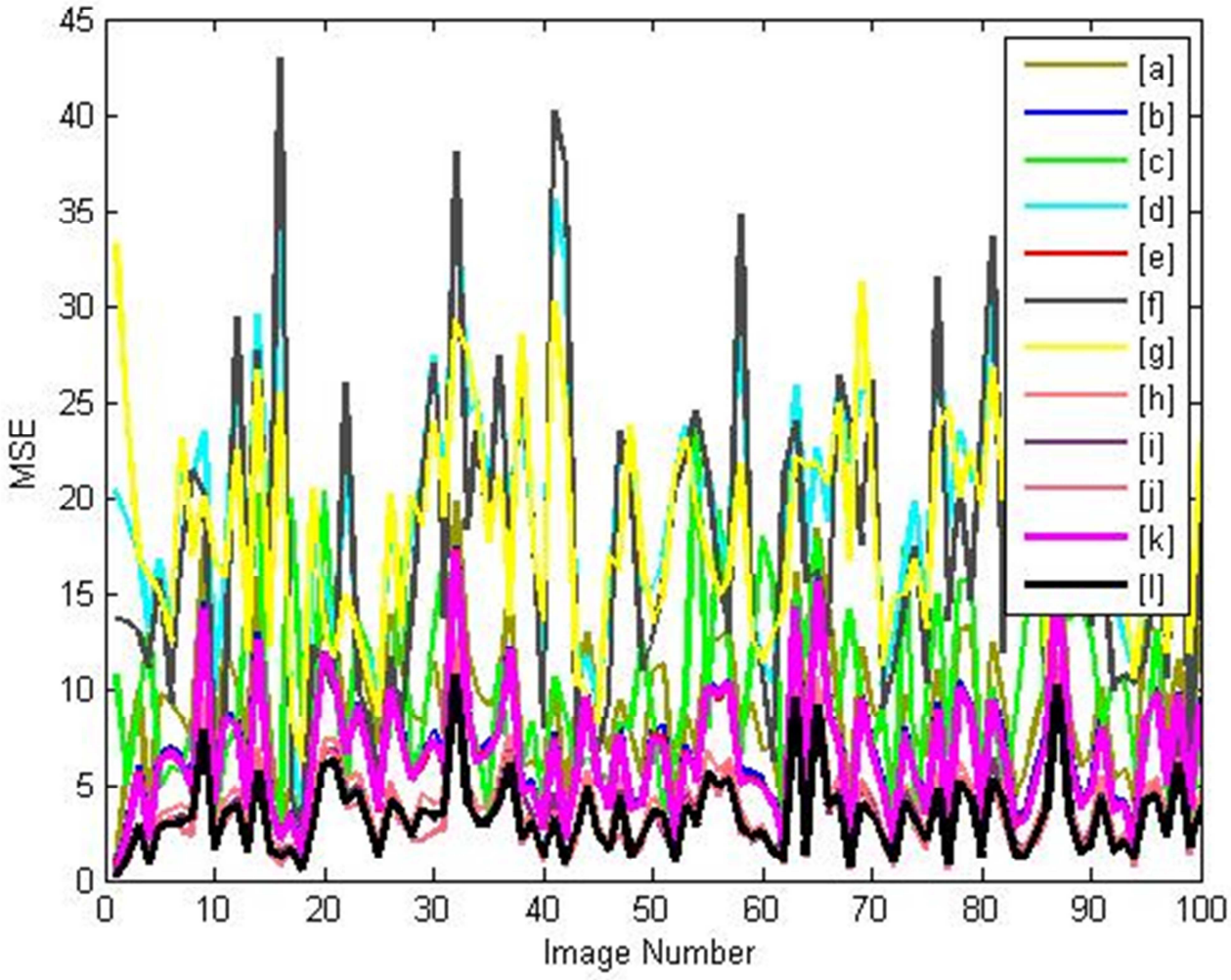


Figure 4.14: MSE Comparison (Berkeley Color Test Images)

Figure 4.15 shows the histogram for the PSNR comparison. The errors are reported for the same set of algorithms. These measures agree with the MSE comparison. The proposed method (l) and method (j) shows superior results to the other algorithms.

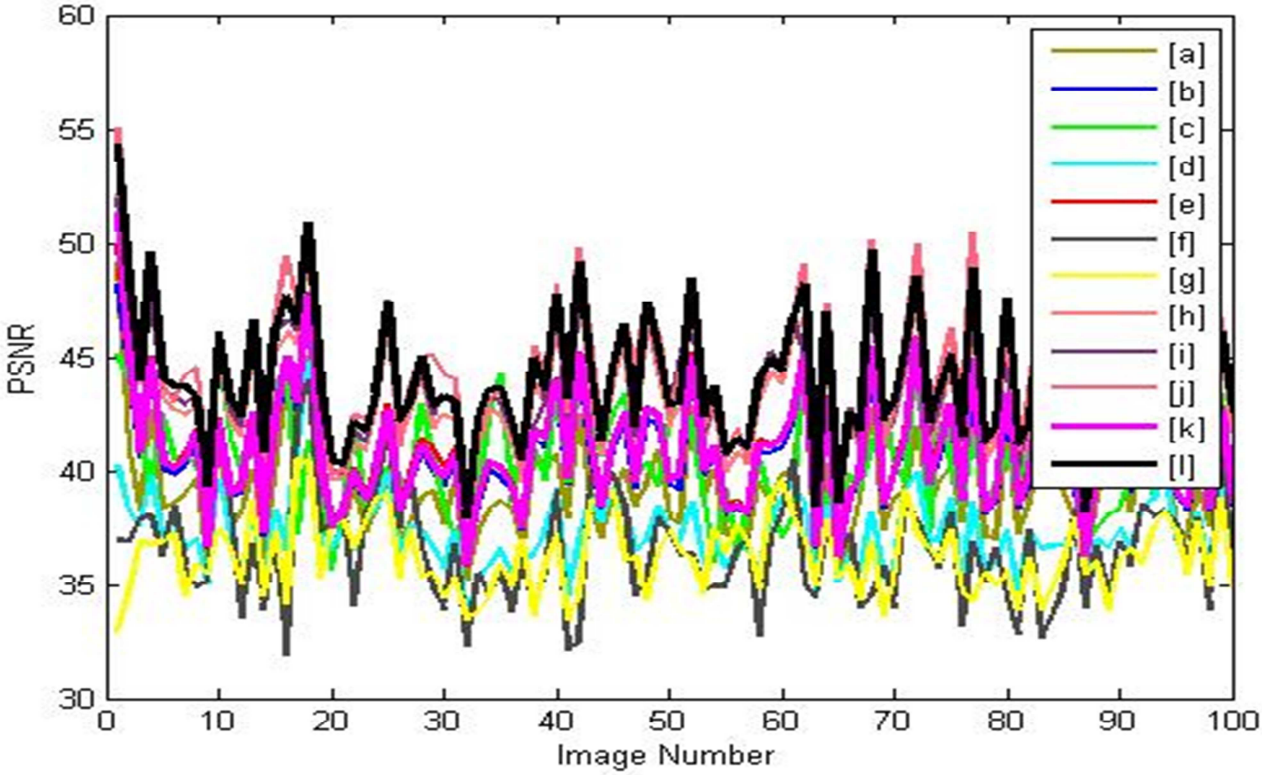


Figure 4.15: PSNR Comparison (Berkeley Color Test Images)

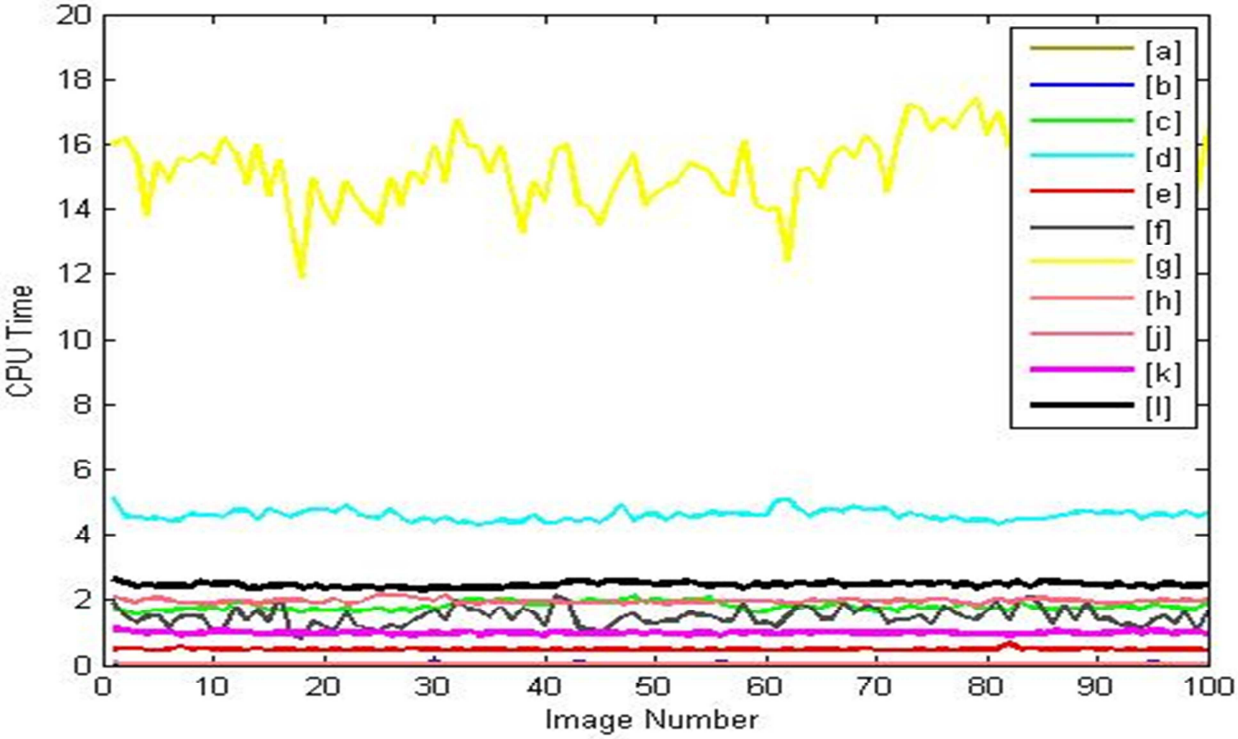


Figure 4.16: CPU Time Comparison (Berkeley Color Test Images)

Figure 4.16 shows the CPU time. The histogram shows that the method (h) is the fastest than all other algorithms, the reason being as explained in previous sections. Our proposed algorithms is still faster than many of the other demosaicking algorithms as shown in figure 4.16.

The numbers can only provide subset of the overall scenario. An important evaluation is the visual appearance of the output images. For this, an example image is presented. Figure 4.17 shows the image 29. This example includes a “building” from a perspective that increases spatial frequency along the region. Aliasing is a prominent artifect in this image. The proposed interpolation algorithm (l) reconstructs this image very best. Very little aliasing is present in the output image. This is good example to show how the algorithm respond to features at various orientations.

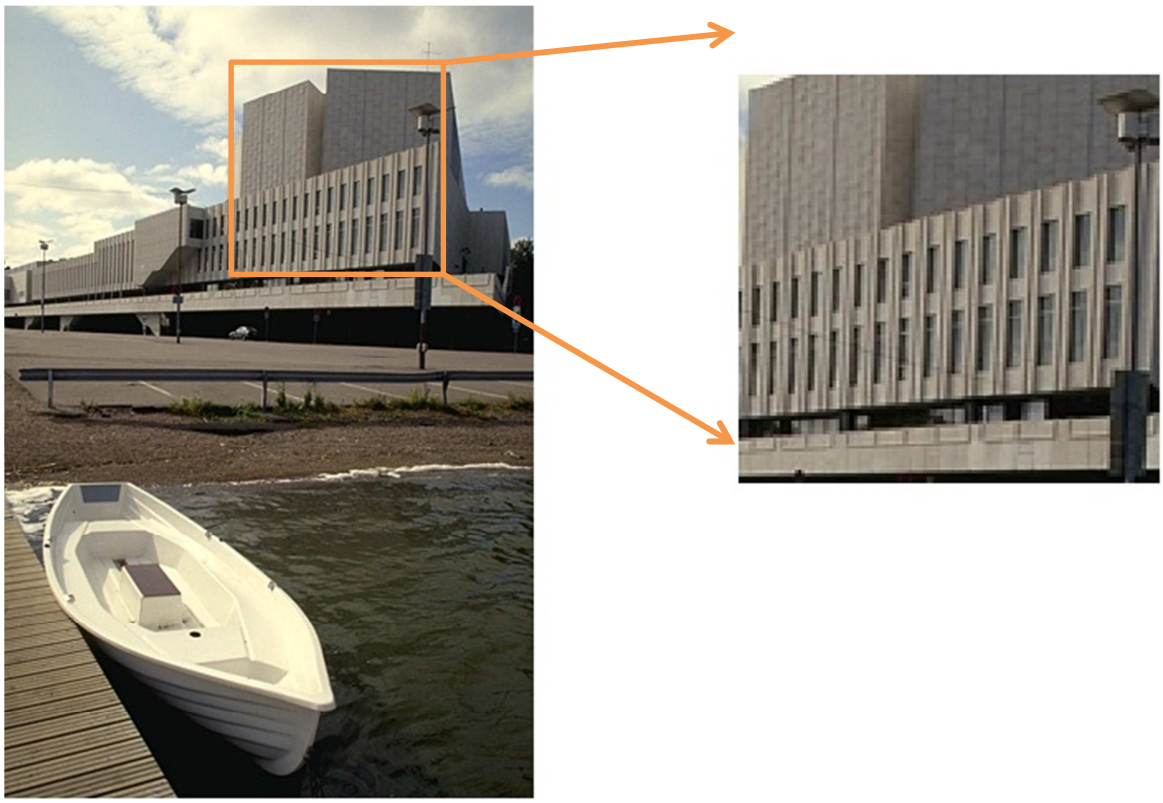


Figure 4.18: Cropped Region of the Original Image (Berkeley Color Test Images).

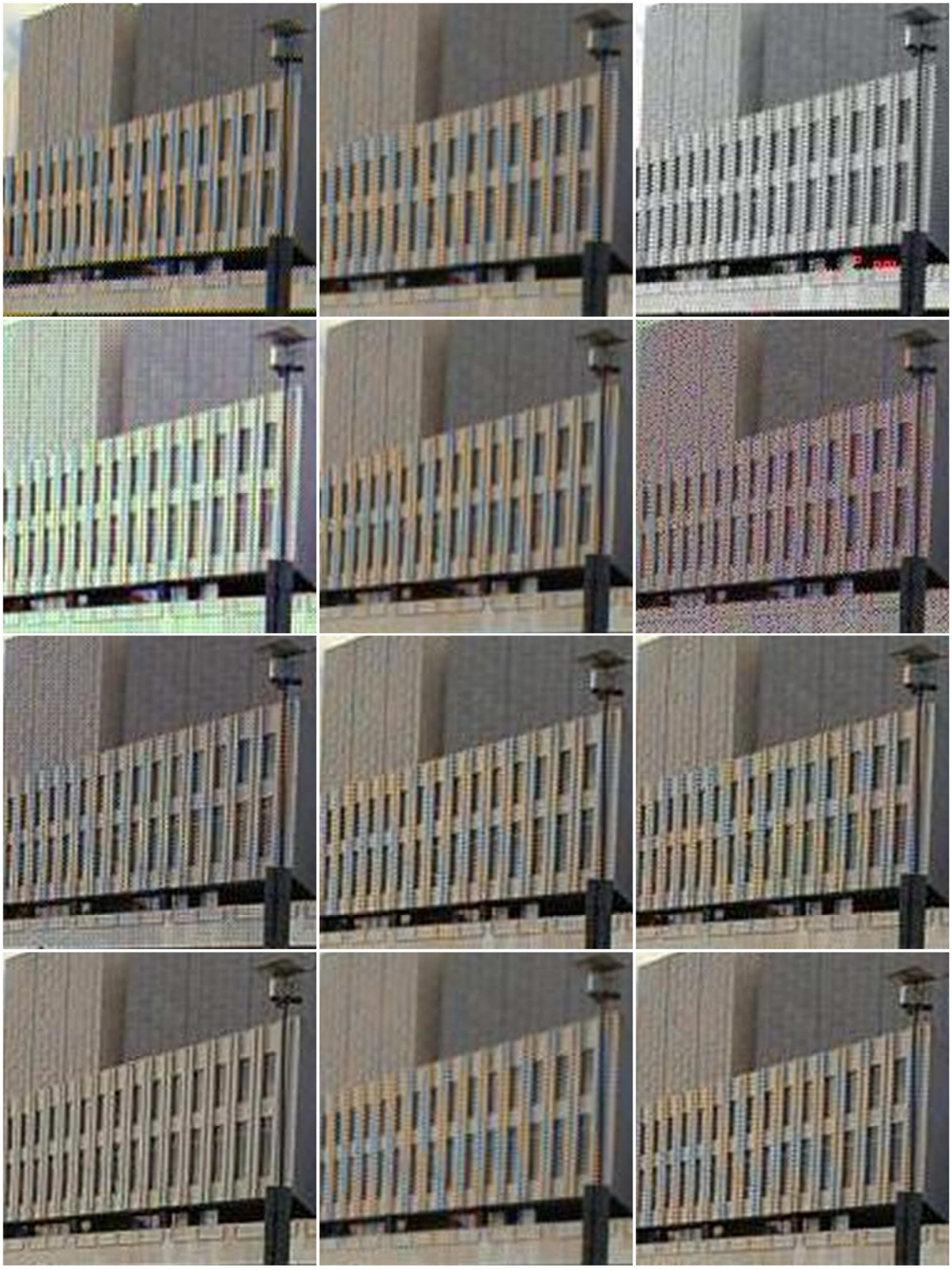


Figure 4.19: Zoomed Images for Visual Comparison (Berkeley Color Test Images)

Chapter 5

Conclusion and Future Directions

In this paper, we have presented a new color interpolation approach for Bayer pattern mosaic images. The proposed algorithm utilizes edge strength for the fuzzy membership assignment as a weighting factor for estimating the missing colors in each pixel. This algorithm significantly improves the overall visual quality of the interpolated color images. The experimental results prove that the algorithm preserves colors on the edges with minimal or no visual artifacts. We have also presented the objective quality metrics in terms of MSE and PSNR to show the performance of the algorithm with five images sets for color interpolation and we observe that PSNR is one of the highest among all comparison methods especially for the proposed fuzzy non-linear method. The CPU time of the proposed algorithm is also faster than those of many methods especially in the case of fuzzy bilinear interpolation, as only simple fuzzy weighted averaging is carried out for color interpolation.

Digital imaging devices such as digital cameras will continue to employ only a single electronic sensor for color interpolation due to the cost and packaging consideration. However demosaicking is still an important problem into research and has explored the imaging process and the correlation among three color planes. Artifact reduction is another research problem in color image interpolation. Temporal correlation in addition to spectral correlation should be exploited. For real time imaging system such as digital cameras, processing time is an important measure for algorithm implementation because a photographer may need to take pictures at a fast rate. Single-lens-reflex (SLR) cameras provide access to the raw image data which can later be processed on a digital computer. Here the processing time is not an issue. Therefore high performance algorithms that are computationally complex can still be implemented for off-line processing applications.

References

[1] Bryce E. Bayer, "Color imaging array," U.S. Patent 3,971,065, Eastman Kodak Company, 1976.

[2] P. The'venaz, T. Blu and M. Unser, "Interpolation revisited," *IEEE Trans. MI*, vol. 19, no. 7, pp. 739-758, 2000.

[3] M. Unser, A Aldroubi, M Eden, "B-spline Signal Processing*: Part1 Theory*," *IEEE Trans. Signal Processing*, vol. 41, pp. 821-833, 1993.

[4] M. Yoneyama, Y. Yamamoto, Y. Tanizoe, S. Sasaki, and H. Okayama, "Development of single chip digital camera with high picture quality," *Proc. ITE Annu. Conv*., vol. 8, no. 5, pp. 119-120, 1994.

[5] T. Kuno and H. Sugiura, "New Interpolation Method using discriminated color correlation for digital still cameras," *IEEE Trans. Consumer Electronics*, vol. 45, no. 1, pp. 259-267, Jan. 1999.

[6] S. Okada, Y Matsuda, T Yamada, A Kobayashi, "System on a chip for digital still camera," *IEEE Trans. Consumer Electronics*, vol. 45, no. 3, pp. 584-590, Aug. 1999.

[7] R. Kimmel, "Demosaicing; Image Reconstruction from Color CCD Sample," *IEEE. Trans. Image Processing*, vol. 8, pp. 1221-1228, Sept. 1999.

[8] R. Lukac and K. N. Plataniotis, "A normalized model for color-ratio based demosaicking schemes," *International Conf on Image Processing*, 2004, vol. 3, pp. 1657–1660.

[9] J. F. Hamilton and J. E. Adams, "Adaptive color plane interpolation in single sensor color electronic camera," U.S. Patent 5 629 734, Mar. 13, 1997.

[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. 13, 1994.

[11] Y. Hel-Or, "The Canonical Correlation of Color Image and their use for Demosaicing," *Tech. Report HPL-2003-164*, HP Labs (internal), Feb. 2004.

[12] B. E. Marino and R. L. Stevenson, "Improving the performance of single chip image capture devices," *IS&T Jour. Electronic Imaging*, vol. 12, no. 2, pp. 209-218, April 2003.

[13] X. L. Michael and T. Orchard, "New Edge-Directed Interpolation," *IEEE Trans. Image Processing*, vol. 10, no. 10, pp. 1521-1527, Oct. 2001.

[14] D. D. Muresan, and T. W. Parks, "Optimal Recovery Approach to Image Interpolation," *IEEE Proc. ICIP 2001*, vol. 3, pp. 7-10, 2001.

[15] D. R. Cok, "Signal Processing Method and Apparatus for Sampled Image Signals," U.S. patent 4, 630, 307, 1986.

[16] 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.*, 2001, vol. 1, pp. 141–144.

[17] 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.

[18] N.-X. Lian, L. Chang, Y.-P. Tan, and V. Zagorodnov, “Adaptive filtering for color filter array demosaicking,” *IEEE Trans. Image Process.*, vol. 16, no. 10, pp. 2515–2525, Oct. 2007.

[19] R. H. Hibbard, “Apparatus and method for adaptively interpolating a full color image utilizing luminance gradients,” U.S. Patent 5 382 976, Jan. 17, 1995.

[20] J. E. Adams and J. F. Hamilton, Jr., “Adaptive color plane interpolation in single sensor color electronic camera,” U.S. Patent 5 506 619, Apr. 9, 1996.

[21] K.-H. Chung and Y.-H. Chan, “Color demosaicing using variance of color differences,” *IEEE Trans. Image Process.*, vol. 15, no. 10, pp. 2944–2955, Oct. 2006.

[22] L. Zhang and X. Wu, “Color demosaicking via directional linear minimum mean square-error estimation,” *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2167–2178, Dec. 2005.

[23] D. Paliy, V. Katkovnik, R. Bilcu, S. Alenius, and K. Egiazarian, “Spatially adaptive color filter array interpolation for noiseless and noisy data,” *Int. J. Imag. Syst. Technol.*, vol. 17, no. 3, pp. 105–122, 2007.

[24] K. Hirakawa and T. W. Parks, “Adaptive homogeneity-directed demosaicing algorithm,” *IEEE Trans. Image Process.*, vol. 14, no. 3, pp. 360 369, Mar. 2005.

[25] D. Menon, S. Andriani, and G. Calvagno, “Demosaicing with directional filtering and a posteriori decision,” *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 132–141, Jan. 2007.

[26] I. Tsubaki and K. Aizawa, "A restoration and demosaicing method for a pixel mixture image," *Proc. SPIE/IS&T Electronic Imaging*, SPIE vol. 5301, pp. 346-355, 2004.

[27] T. Gotoh and M. Okutomi, "Color Super-Resolution from a Single CCD," CD-ROM *Proc. IEEE Workshop on color and photometric methods in computer vision*, 2003.

[28] P. Vandewalle , S. Susstrunk, and M. Vetterli, "Double resolution from a set of aliased images," *SPIE/IS&T Electronic Image Conf.*, 2004.

[29] S. Farsiu, M. Elad, and P. Milanfar, "Multi-Frame Demosaicing and Super-Resolution from Under-Sampled Color Images," *Proc. 2004 IS&T/SPIE 16th Annual Sympo. Electronic Imaging*, pp. 222-233, Jan. 2004.

[30] L. Chang and Y. Tan, "Effective use of Spatial and Spectral Correlations for Color Filter Array Demosaicing," *IEEE Trans. Consumer Electronics*, vol. 50, no. 1, pp. 355-365, Feb. 2004.

[31] H. J. Trussell and R. E. Hartwig, "Mathematics for Demosaicing," *IEEE Trans. Image Processing*, vol. 11, no. 4, pp. 485-492, April 2002.

[32] Bo Tao, I. Tastl, T. Cooper, M. Blasgen, and E. Edwards, "Demosaicing using Human Visual Properties and Wavelet Interpolation Filtering," *US Sony Proc. IS&T/SID 7th Color Image Conference (CIC'99)*, pp. 252- 256, 1999.

[33] Z. Baharav and R. Kakarala, "Compression aware demosaicing methods," *Proc.SPIE, Image Processing*, vol. 4667, pp. 149-156, 2002.

[34] Y. Hel-Or and D. Keren, "Demosaicing of Color Image Using Steerable Wavelets," *Tech. Report HPL-97-104*, HP Labs (internal), 1997.

[35] James E. Adams, Jr., "Interactions between color plane interpolation and other image processing functions in electronic photography," *Proceedings of the SPIE Electronic Imaging Conference*, Vol. 2416, pp:144-151, 1995.

[36] Tadashi Sakamoto, Chikako Nakanishi and Tomohiro Hase, "Software Pixel Interpolation for Digital Still Cameras Suitable for A 32-bit MCU," *IEEE Transactions on Consumer Electronics*, Vol. 44, No. 4, pp 1342-1352, Nov. 1998.

[37] David R Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal," U.S. Patent 4,642,678, Eastman Kodak Company, 1987.

[38] 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, 1986.

[39] W.T. Freeman, "Method and apparatus for reconstructing missing color samples," U.S. Patent 4 774 565, 1988.

[40] R. Kimmel, "Demosaicing: image reconstruction from CCD samples," *IEEE Trans. Image Processing*, vol. 8, no. 9, pp. 1221–1228, 1999.

[41] S.-C. Pei and I.-K. Tam, "Effective color interpolation in CCD color filter arrays using signal correlation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 6, pp. 503–513, June 2003.

[42] J.A. Weldy, "Optimized design for a single-sensor color electronic camera system," *Proc. SPIE*, vol. 1071, pp. 300–307, May 1988.

[43] X. Wu, W. K. Choi, and P. Bao, "Color Restoration from Digital Camera Data by Pattern Matching," *Proceedings of the SPIE's Electronic Imaging Conference Color Imaging: Device- Independent Color, Color Hardcopy, and Graphic Arts II*, Vol. 3018, pp. 12-17, 1997.

[44] Chang, Ed. et.al., "Color Filter Array Recovery Using a Threshold-based Variable Number of Gradients" *to be published in Proceedings of SPIE*, January, 1999.

[45] Ibrahim Pekkucuksen, *Student Member, IEEE*, and Yucel Altunbasak, *Senior Member, IEEE,* "Edge Strength Filter Based Color Filter Array Interpolation", *IEEE TRANSACTIONS ON IMAGE PROCESSING*, VOL. 21, NO. 1, JANUARY 2012.

[46] T. Kuno and H. Sugiura, "Practical Color Filter Array Interpolation Part 2 with Non-linear Filter " *IEEE Transactions on Consumer Electronics*, Vol. 52, No. 4, NOVEMBER 2006.

[47] Tetsuya Kuno and Hiroaki Sugiura, *Member*, IEEE, "Practical Color Filter Array Interpolation with Constrained Color Correlation", *IEEE Transactions on Consumer Electronics*, Vol. 52, No. 3, AUGUST 2006.

[48] http://scien.stanford.edu/pages/labsite/1999/psych221/projects/99/tingchen/main.htm.

[49] http://www.microscopyu.com/articles/fluorescence/index.html.

[50] http://www.olympusmicro.com/galleries/index.html.

[51] http://www.mvainc.com/category/image-gallery/.

[52] http://www.microimaging.ca/forum.htm.

[53] http://www.microscopy-uk.net/.

[54] http://micro.magnet.fsu.edu/micro/gallery.html.

[55] http://zeiss-campus.magnet.fsu.edu/galleries/apotome/index.html.

[56] http://www.landsat.org/.

[57] Rich Franzen, "Kodak Lossless True Color Image Suite", URL: http://r0k.us/graphics/kodak/.

[58] X. Li, B. Gunturk, and L. Zhang, "Image demosaicing: A systematic survey," *Proc. SPIE–Int. Soc. Opt. Eng.*, vol. 6822, p. 68221J-1-15, Jan. 2008.

[59] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

[60] Pablo Arbel´aez et al. "From Contours to Regions: An Empirical Evaluation", *ONR MURI* N00014-06-1-0734.

[61] Sungwoong Kim et al. "Higher-Order Correlation Clustering for Image Segmentation", *Machine Learning and Perception*, Microsoft Research Cambridge.

[62] Newlin, Dev R., and Elwin Chandra Monie. "EDGE SENSING DEMOSAICING USING ADAPTIVE WEIGHTED INTERPOLATION." *American Journal of Applied Sciences* 10.4 (2013): 418.

[63] Fan, Yu-Cheng, Yi-Feng Chiang, and Yin-Te Hsieh. "Constant-hue Based Color Filter Array Demosaicking Sensor for Digital Still Camera Implementation." (2013): 1-1.

[64] Wang, Xingbo, et al. "Median filtering in multispectral filter array demosaicking."*IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2013.

[65] Li, Hua. "SDRAM Based Hardware Design of Bayer-pattern Demosaicking."*Dianshi Jishu(Video Engineering)* 37.5 (2013).

[66] Wang, Guo-gang, Xiu-chang Zhu, and Zong-liang Gan. "Image demosaicing by non-local similarity and local correlation." *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*. Vol. 2. IEEE, 2012.

[67] Côté, Guy, and Jeffrey E. Frederiksen. "SYSTEM AND METHOD FOR DEMOSAICING IMAGE DATA USING WEIGHTED GRADIENTS." European Patent No. EP 2491721. 29 Aug. 2012.

[68] Guarnera, M., G. Messina, and V. Tomaselli. "Demosaicing and Aliasing Correction." *Red* 33.R53: R53.

[69] Hore, Alain, and Djemel Ziou. "An edge-sensing generic demosaicing algorithm with application to image resampling." *Image Processing, IEEE Transactions on* 20.11 (2011): 3136-3150.

[70] Kakarala, Ramakrishna, and Izhak Baharav. "Method and system for adaptive demosaicing." European Patent No. EP 1289310. 13 Jul. 2011.

[71] DiCarlo, Jeffrey, David D. Scott, and Wenyi Zhao. "METHODS AND APPARATUS FOR DEMOSAICING IMAGES WITH HIGHLY CORRELATED COLOR CHANNELS." U.S. Patent Application 13/027,071.