

# EDGE DIRECTED RESOLUTION ENHANCEMENT AND DEMOSAICING

A Thesis  
Presented to  
The Academic Faculty

by

Ibrahim E. Pekkucuksen

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy in the  
School of Electrical and Computer Engineering

Georgia Institute of Technology  
December 2011

Copyright © 2011 by Ibrahim E. Pekkucuksen

# EDGE DIRECTED RESOLUTION ENHANCEMENT AND DEMOSAICING

Approved by:

Professor Yucel Altunbasak, Advisor  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Professor James McClellan  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Professor Xiaoli Ma  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Professor Ghassan AlRegib  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Professor Bahadir Gunturk  
Department of Electrical and  
Computer Engineering  
*Louisiana State University*

Date Approved: August 9, 2011

*To my parents,*

## ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Yucel Altunbasak for his invaluable support and guidance. I also would like to thank my friends at CSIP and MCCL, with whom I shared many great memories over the years. Mehmet Umut Demircin, Salih Dikbas, Zafer Aydin, Osman Sezer, Salman Aslam, Aytac Azgin, Tarik Arici, Toygar Akgun, Ali Cafer Gurbuz, Michael Santoro, Ali Cengiz Begen, and many others, thank you for countless technical and non-technical discussions and for your friendship.

I would like to thank the members of the Systems and Applications R&D Center at Texas Instruments, for giving me the opportunity to work in an exceptional industrial research environment. Special thanks go to Aziz Umit Batur, Buyue Zhang, and Wei Hong. I also thank Prof. James McClellan, Prof. Xiaoli Ma, Prof. Ghassan AlRegib, and Prof. Bahadir Gunturk for serving in my committee.

Finally, I would like to thank my family, without their support this work would not have been completed.

# TABLE OF CONTENTS

<b>DEDICATION</b> . . . . .	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b> . . . . .	<b>iv</b>
<b>LIST OF TABLES</b> . . . . .	<b>vii</b>
<b>LIST OF FIGURES</b> . . . . .	<b>viii</b>
<b>SUMMARY</b> . . . . .	<b>ix</b>
<b>I INTRODUCTION</b> . . . . .	<b>1</b>
<b>II ORIGIN AND HISTORY OF THE PROBLEM</b> . . . . .	<b>3</b>
2.1 Image Interpolation . . . . .	3
2.2 Demosaicing . . . . .	5
<b>III EDGE DIRECTED IMAGE INTERPOLATION</b> . . . . .	<b>9</b>
3.1 Background . . . . .	9
3.2 Algorithm Details . . . . .	9
3.3 Experimental Results . . . . .	19
<b>IV DEMOSAICING ON THE BAYER PATTERN</b> . . . . .	<b>23</b>
4.1 Edge Strength Filter Based Demosaicing . . . . .	23
4.1.1 A New Filter . . . . .	23
4.1.2 Initial Green Channel Interpolation . . . . .	25
4.1.3 Green Channel Update . . . . .	27
4.1.4 Red and Blue Channel Interpolation . . . . .	28
4.2 Color Difference Gradients Based Demosaicing . . . . .	30
4.2.1 Algorithm Background . . . . .	30
4.2.2 Green Channel Interpolation . . . . .	30
4.2.3 Red and Blue Channel Interpolation . . . . .	33
4.3 Multiscale Gradients Based Demosaicing . . . . .	34
4.3.1 Algorithm Background . . . . .	34

4.3.2	Initial Green Channel Interpolation . . . . .	38
4.3.3	Green Channel Update . . . . .	39
4.3.4	Red and Blue Channel Interpolation . . . . .	40
4.4	Experimental Results on the Bayer Pattern . . . . .	42
<b>V</b>	<b>DEMOSAICING ON THE LUKAC PATTERN . . . . .</b>	<b>45</b>
5.1	Motivation . . . . .	45
5.2	Application of the ESF Based Method to the Lukac Pattern . . . . .	45
5.2.1	Green Channel Interpolation . . . . .	46
5.2.2	Red and Blue Channel Interpolation . . . . .	49
5.3	Application of the MSG Based Method to the Lukac Pattern . . . . .	50
5.3.1	Green Channel Interpolation . . . . .	50
5.3.2	Red and Blue Channel Interpolation . . . . .	52
5.4	Experimental Results on the Lukac Pattern . . . . .	53
<b>VI</b>	<b>LOW SPECTRAL CORRELATION DEMOSAICING . . . . .</b>	<b>56</b>
6.1	Algorithm Background . . . . .	56
6.2	Green Channel Interpolation . . . . .	57
6.3	Red and Blue Channel Interpolation . . . . .	59
6.4	Experimentel Results . . . . .	61
<b>VII</b>	<b>A NEW FAMILY OF CFA PATTERNS . . . . .</b>	<b>64</b>
7.1	Introduction . . . . .	64
7.2	Pattern Design . . . . .	64
7.3	Experimental Results . . . . .	68
7.4	Extension of the Proposed Pattern . . . . .	70
<b>VIII</b>	<b>CONCLUSIONS AND FUTURE WORK . . . . .</b>	<b>74</b>
<b>REFERENCES</b>	<b>. . . . .</b>	<b>76</b>
<b>VITA</b>	<b>. . . . .</b>	<b>80</b>

## LIST OF TABLES

1	Comparison of PSNR values for different interpolation methods. . . .	20
2	Comparison of SSIM index results for different interpolation methods.	21
3	Comparison of PSNR values for different demosaicing methods. . . .	43
4	Comparison of CPSNR values for different demosaicing methods on the Lukac pattern. . . . .	54
5	Comparison of PSNR values for different demosaicing methods on the new McMaster dataset. . . . .	61
6	Comparison of PSNR values for different demosaicing methods on the new McMaster dataset (continued). . . . .	62
7	Spectral correlation comparison on the Kodak dataset . . . . .	66
8	Spectral correlation comparison on the McMaster dataset . . . . .	67
9	Comparison between Bayer and proposed pattern on the Kodak dataset.	69
10	Comparison between Bayer and proposed pattern on the McMaster dataset. . . . .	71

## LIST OF FIGURES

1	Bayer CFA pattern. . . . .	6
2	First interpolation step. . . . .	10
3	Second interpolation step. . . . .	11
4	Consistency checking. . . . .	12
5	Coefficient correlation. . . . .	14
6	Four coefficients. . . . .	16
7	Wall detail: original image (a), bilinear interpolation (b), NEDI (c), AQua2 (d), and proposed method (e). . . . .	18
8	Grayscale pixels. . . . .	23
9	Bayer CFA pixels. . . . .	24
10	Mosaicked lighthouse image and its edge strength filter output. . . . .	25
11	Horizontal and vertical color difference maps. . . . .	32
12	Relationship between the color difference gradients equation and the multiscale gradients equation. . . . .	37
13	24 image Kodak dataset. . . . .	42
14	Comparison on the Bayer pattern. Fence region from the lighthouse image. . . . .	44
15	Numbered Lukac pattern layout. . . . .	46
16	Mosaicked region and its edge strength filter output. . . . .	46
17	Comparison on the Lukac pattern. Fence region from the lighthouse image. . . . .	55
18	18 image McMaster dataset. . . . .	57
19	Proposed Bayer based pattern. . . . .	68
20	Image 17 interpolated with the low correlation method on the Bayer pattern and the proposed pattern. . . . .	70
21	Proposed Lukac based pattern. . . . .	72
22	Patterns generated by replacing channel locations in Bayer and Lukac based layouts. . . . .	73



## SUMMARY

The objective of the proposed research is to develop high performance, low computational complexity resolution enhancement and demosaicing algorithms. Our approach to both problems is to find creative ways to incorporate edge information into the algorithm design. However, in contrast with the usual edge directed approaches, we do not try to detect edge presence and orientation explicitly. For the image interpolation problem, we study the relationship between low resolution and high resolution pixels, and derive a general interpolation formula to be used on all pixels. This simple interpolation algorithm is able to generate sharp edges in any orientation. We also propose a simple 3 by 3 filter that quantifies local luminance transition and apply it to the demosaicing problem. Additionally, we propose a gradient based directional demosaicing method that does not require setting any thresholds. We show that the performance of this algorithm can be improved by using multiscale gradients. Finally, we address the low spectral correlation demosaicing problem by proposing a new family of hybrid color filter array (CFA) patterns and a local algorithm that is two orders of magnitude faster than a comparable non-local solution while offering the same level of performance.

# CHAPTER I

## INTRODUCTION

Digital images are comprised of data samples arranged in a two dimensional grid. These data samples are usually referred to as picture elements or pixels. The number of pixels in an image determines its resolution. The higher number of pixels an image has, the more information it could contain and the better it could represent the original data. In other words, all other things being equal, a high resolution image has better quality than a low resolution one.

Changing the resolution of an image is called image resampling. One may need to resample an image for a variety of reasons. For instance, if a display device has lower resolution than an image to be displayed, then the image needs to be downsampled so that it could fit to the display screen. Similarly, if an image takes up too much data storage space or takes too long to transmit, a possible solution (other than applying compression) is to downsample the image. On the other hand, a low resolution image can be upsampled to improve its visual quality. From a digital signal processing point of view, image downsampling is arguably simpler than upsampling because in the downsampling case all the information is already available and the only challenge is to represent it with a smaller number of pixels. However, for the upsampling case, one needs to create new information by interpolating the available input pixels. From this point on, we will refer to image upsampling when we talk about image interpolation or image resampling.

Natural images generally consist of various regions with different characteristics. While some regions/objects are smooth, others are structured or textured. Moreover, edges form wherever object boundaries meet, which leads to sharp luminance changes.

While most interpolation algorithms interpolate smooth regions successfully, they tend to fail in edge packed regions. Common interpolation failures are blurriness and staircase effect which refers to edge jagginess. In order to avoid such artifacts, some interpolation algorithms try to detect edge presence and orientation, and adapt the interpolation coefficients accordingly. However, edge detection can be error prone and costly, which results in degraded interpolation performance. We propose an edge preserving interpolation method that does not require explicit edge detection. The proposed method studies the relationship between low and high resolution pixels and it applies the same interpolation formula to all input pixels.

Demosaicing or Color Filter Array (CFA) interpolation is a special image interpolation problem. Here, the image size is fixed but only a subset of the color information is available at each pixel location. The missing information at every pixel need to be estimated to obtain the complete color image. While spatial correlation is the only estimation basis for regular image interpolation, spectral correlation between the color channels also comes into play for the demosaicing problem. Demosaicing algorithms need to exploit both of them to avoid false color artifacts that are closely associated with the demosaicing process.

Simple spatially invariant demosaicing methods work in smooth regions with subtle color changes, but they tend to fail around structures with saturated colors. Adaptive methods that take advantage of local directional information have been introduced to improve the interpolation quality. Our work on the demosaicing area have resulted in several algorithms and a new family of CFA patterns.

## CHAPTER II

### ORIGIN AND HISTORY OF THE PROBLEM

The image interpolation problem in general and the CFA interpolation problem as a special case of it have been studied for many years. The following sections provide a brief literature survey on both problems.

#### *2.1 Image Interpolation*

Different approaches to the spatial interpolation problem may be categorized as

- Linear spatially invariant interpolation
- Transform domain interpolation
- Statistical learning based interpolation
- Edge adaptive interpolation

Linear spatially invariant interpolation techniques such as bilinear and bicubic interpolation [21] have low computational complexity. However, they often fail to protect the integrity of edge structures and introduce blurring. Some adaptive techniques have been proposed to overcome such shortcomings [25]. Transform based algorithms try to extract high frequency information from the image and use it to improve interpolation quality. Although wavelet transform is the most common method of choice [6, 7], algorithms based on other transforms, such as fourier and discrete cosine, were proposed too [8, 11]. The main drawback of the transform based algorithms is their high computational cost. On top of the transform and inverse transform calculation costs, the iterative nature of these algorithms make them computationally expensive compared to linear interpolation methods.

Another approach to the interpolation problem is to use image statistics for training interpolation filters. The idea is to classify pixels using some kind of feature extraction and to train suitable filters for each pixel class. Resolution Synthesis [3] is an early example of classification based algorithms. Lenke et al. [26] proposed a classification based polynomial interpolation and applied their ideas to temporal interpolation for video sequences. Another method described in [19] uses neural networks to train content-adaptive filters.

Edge adaptive interpolation is yet another approach to the interpolation problem. It became a center of focus because the importance of edge preservation for improved interpolation quality has been recognized early on [41, 20].

The New Edge Directed Interpolation (NEDI) proposed by Li et al. [29] is a spatially adaptive interpolation technique that uses local covariance information to preserve the edge structure. Its basic assumption is that there is a significant correlation between low resolution and high resolution local covariances. Once the local covariance for the low resolution image is estimated, it can be used to adapt the interpolation coefficients for that neighborhood.

NEDI algorithm is powerful at maintaining well defined edges. It does not introduce any sign of staircase effect in most cases. However, the algorithm tends to perform poorly in textured areas especially where closely packed edge structures are present. It also introduces artifacts for perfectly horizontal or vertical edges. Another important disadvantage of NEDI is its high computational cost. It requires around 1300 multiplications per pixel for a local window size of 8 [29]. The number of computations can be reduced by excluding the smooth regions since they do not require edge directed interpolation. However, computational complexity still remains high for real time applications.

Muresan et al. [36] proposed selecting local quadratic signal class based on training data and using optimal recovery theory for interpolation. A simplified edge directed

interpolation algorithm based on the ideas in [36] is presented in [35]. This algorithm detects the presence of an edge and its direction, and applies some form of linear interpolation based on the edge direction decision.

Another algorithm proposed in [45] uses directional filtering and data fusion for edge directed interpolation. The algorithm starts with interpolating the image with a conventional method such as bicubic or bilinear. Then, it generates two orthogonal observation sets for each interpolated pixel. It treats these sets as noisy observations of the desired pixel value and uses local statistics to combine them adaptively. A simpler version of the algorithm which makes further assumptions to reduce the computational complexity is also presented.

The algorithm reduces ringing artifacts and its computational complexity is far less than NEDI. However, although satisfactory, its edge preservation is not as perfect as the NEDI algorithm. Also, its performance is dependent on the initial interpolation method used and the simplifying assumptions made. Furthermore, even though the method combines the orthogonal observations optimally in its own domain, its statistical data is limited to the directions of these observations rather than the whole neighborhood because of the directional nature of the algorithm.

## ***2.2 Demosaicing***

Color images require multiple data samples for each pixel as opposed to grayscale images for which a pixel is represented by only one data sample. For the RGB image format, these data samples represent red, green, and blue channels. A typical digital camera captures only one of these channels at each pixel location and the other two need to be estimated to generate the complete color information. This process is called Color Filter Array (CFA) interpolation or demosaicing. Although many CFA patterns have been proposed over the years, the most prevalent one is the Bayer pattern shown in Figure 1 [4]. Bayer pattern is an example of pure RGB based CFA

patterns. Some pattern designs are comprised of elements that are combinations of RGB colors such as the Hiraakawa pattern [18].



**Figure 1:** Bayer CFA pattern.

As an important step in image processing pipeline of digital cameras, demosaicing has been an area of interest both in academia and industry. The simplest approach to the demosaicing problem is to treat color channels separately and fill in missing pixels in each channel using a spatially invariant interpolation method such as bi-linear or bicubic interpolation. While such an approach works fine in homogenous areas, it leads to color artifacts and lower resolution in regions with texture and edge structures.

Obtaining better demosaicing performance is possible by exploiting the correlation between color channels. Spectral correlation can be modeled by either constant color ratio rule [22, 31] or constant color difference rule [14, 23]. The basic assumption is that color ratio/difference is constant over a local distance inside a given object. This assumption is likely to break apart across boundaries, hence many demosaicing algorithms try to utilize it adaptively in one way or another.

Since Bayer CFA pattern has twice as many green channel samples as red and blue ones, green channel suffers less from aliasing and it is the natural choice as the starting point for CFA interpolation process. In [12], Glotzbach et al. proposed improving red and blue channel interpolation by adding high frequency components extracted from green channel to red and blue channels. In another frequency domain approach, Gunturk et al. [13] used an alternating projections scheme based on strong inter-channel correlation in high frequency subbands. Although the main objective is to refine red and blue channels iteratively, the same approach can also improve green channel interpolation beforehand which in turn yields better red and blue channel results. A more recent method [30] makes several observations about color channel frequencies and suggests that filtering the CFA image as a whole instead of individual color channels should preserve high frequency information better. To estimate luminance, the method proposes a fixed 5 by 5 filter at green pixel locations and an adaptive filter for red and blue pixel locations. Estimated full resolution luminance is then used to complete the missing chrominance information.

Edge-directed green channel interpolation has been proposed early on with various direction decision rules [14, 23, 16, 2]. The method outlined in [14] is particularly noteworthy because it proposed using derivatives of chrominance samples in initial green channel interpolation. Several subsequent demosaicing algorithms made use of this idea. Authors of [9] proposed using variance of color differences as a decision rule while Zhang et al. [42] proposed making a soft decision to improve the interpolation performance of the original method. In this method [42], color differences along horizontal and vertical directions are treated as noisy observations of target pixel color difference and they are combined optimally using Linear Minimum Mean Square Error Estimation (LMMSE) framework. Paliy et al. [37] further improved directional filtering proposed in [42] by introducing scale adaptive filtering based on linear polynomial approximation (LPA).



Several methods proposed performing interpolation in both horizontal and vertical directions and making a posteriori decision based on some criteria. Hirakawa et al. [17] compared local homogeneity of horizontal and vertical interpolation results and Menon et al. [33] used color gradients over a local window to make the direction decision.

## CHAPTER III

### EDGE DIRECTED IMAGE INTERPOLATION

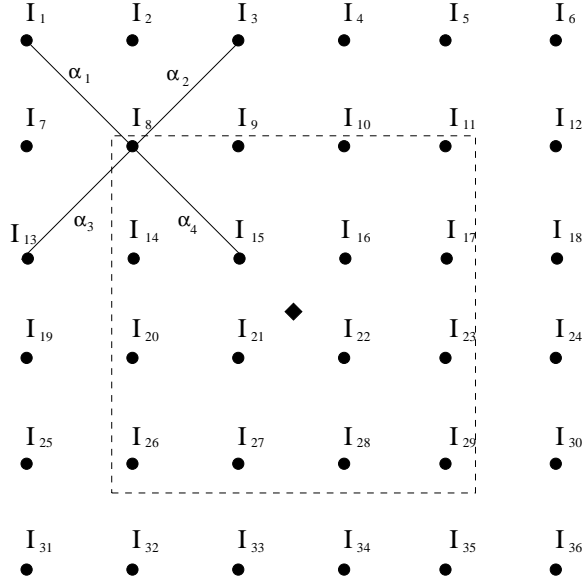
#### *3.1 Background*

When we started looking into the image interpolation problem, our aim was to develop a fast method that could avoid blurry output and jaggy edges. By studying the relationship between high resolution and low resolution pixels in the same neighborhood and employing Bayesian inference on the findings, we were able to come up with a successful edge directed method. Instead of trying to detect edge presence and orientation explicitly and performing interpolation based on that, the proposed method applies a simple yet powerful formula to all regions.

The proposed interpolation method is built upon the concept of geometric duality between low resolution and high resolution pixels. Namely, the relationship between the adjacent pixels of a low resolution image is correlated to that of the high resolution pixels in the same neighborhood. Thus, the interpolation performance can be improved by analyzing the interaction between low resolution pixels and formulating a cost function for the synthesis of high resolution pixels based on this analysis.

#### *3.2 Algorithm Details*

For the general case of scaling an image by a ratio of 2, the interpolation can be performed in two steps. The first step is to estimate the diagonal pixels by using the closest four neighbors all of which are available in the input image. The second step is to fill in the remaining pixels using both the pixels interpolated in the first step and the original ones. The second step is no different than the first one except that the pixel orientations are rotated by 45 degrees. The interpolation steps are illustrated



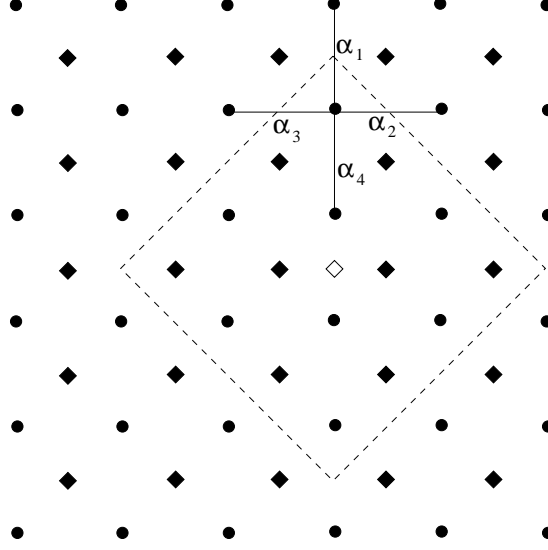
**Figure 2:** First interpolation step.

in Figure 2 and 3, respectively. The black circles in Figure 2 represent the already available low resolution pixels and the black diamond represents the diagonal pixel to be interpolated. Figure 3 demonstrates the second interpolation step with the white diamond representing the pixel to be interpolated while the black circles and black diamonds representing the already available pixels.

The coefficient set determines the weights given to the neighboring pixels for generating the interpolated pixel value. Any number of neighboring pixels can be chosen to be included in the coefficient set. Although a bigger coefficient set with more input pixels can achieve higher interpolation quality, it also leads to more computational complexity. The coefficient value for each pixel in the set is selected according to a cost function on the local training window in the low resolution image. The interpolated value of an already known input pixel in the training window is:

$$\hat{I}_8 = \alpha_1 \cdot I_1 + \alpha_2 \cdot I_3 + \alpha_3 \cdot I_{13} + \alpha_4 \cdot I_{15} \quad (1)$$

To measure the performance of the particular coefficient set, the actual and the interpolated pixel values are compared. The square or the absolute value of the error



**Figure 3:** Second interpolation step.

defines the cost for that particular pixel:

$$\text{Square Error} = (I_8 - \hat{I}_8)^2$$

$$\text{Absolute Error} = |(I_8 - \hat{I}_8)|. \quad (2)$$

The same calculation is performed for each pixel in the training window. Although it can be any positive integer, the size of the training window is 4 by 4 in Figure 2. Choosing the size an even number ensures symmetry with respect to the interpolated pixel. The final cost for the particular local neighborhood is calculated by adding all the individual costs of the training window pixels.

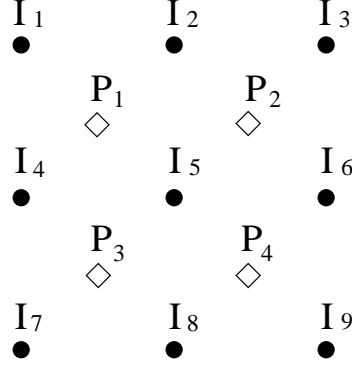
$$\text{Total Square Error} = \sum (I_i - \hat{I}_i)^2$$

$$\text{Total Absolute Error} = \sum |(I_i - \hat{I}_i)|. \quad (3)$$

The calculation of the final cost function can be altered by giving more weight to the closest neighbors of the interpolated pixel.

An alternative cost function scheme is to use all eight closest neighbors of a training window pixel. It is based on the idea to find the error when an already known pixel

is interpolated from its closest four interpolated neighbors as shown in Figure 4. It can be thought as a way of consistency checking for a coefficient set. Assuming that the same coefficient set is valid for interpolation on a small window of 3 by 3, which is a very reasonable assumption to make, the following formulas are extracted:



**Figure 4:** Consistency checking.

$$\begin{aligned}
\widehat{P}_1 &= \alpha_1 I_1 + \alpha_2 I_2 + \alpha_3 I_4 + \alpha_4 I_5 \\
\widehat{P}_2 &= \alpha_1 I_2 + \alpha_2 I_3 + \alpha_3 I_5 + \alpha_4 I_6 \\
\widehat{P}_3 &= \alpha_1 I_4 + \alpha_2 I_5 + \alpha_3 I_7 + \alpha_4 I_8 \\
\widehat{P}_4 &= \alpha_1 I_5 + \alpha_2 I_6 + \alpha_3 I_8 + \alpha_4 I_9 \\
\widehat{I}_5 &= \alpha_1 \widehat{P}_1 + \alpha_2 \widehat{P}_2 + \alpha_3 \widehat{P}_3 + \alpha_4 \widehat{P}_4.
\end{aligned} \tag{4}$$

Solving for  $\widehat{I}_5$  in terms of original  $I$  pixels gives the following equation:

$$\begin{aligned}
\widehat{I}_5 &= \alpha_1^2 I_1 + 2\alpha_1 \alpha_2 I_2 + \alpha_2^2 I_3 + 2\alpha_1 \alpha_3 I_4 + (2\alpha_1 \alpha_4 + 2\alpha_2 \alpha_3) I_5 \\
&\quad + 2\alpha_2 \alpha_4 I_6 + \alpha_3^2 I_7 + 2\alpha_3 \alpha_4 I_8 + \alpha_4^2 I_9.
\end{aligned} \tag{5}$$

Again, the error is calculated for all pixels in the training window and added together to find the final cost for the neighborhood. Note that all 8 first degree neighbor pixels contribute to the cost function in this case whereas only 4 diagonal neighbors were used in the first cost function.

With the cost function defined for a particular coefficient set, different sets can be compared to find the best one for each local neighborhood. The question at this point is, how the coefficients should be selected for the best interpolation quality. In NEDI's case, the coefficients are determined according to the local covariance matrix. However, this can lead to strange coefficients even with singular matrices controlled, which in turn leads to visual artifacts in the image.

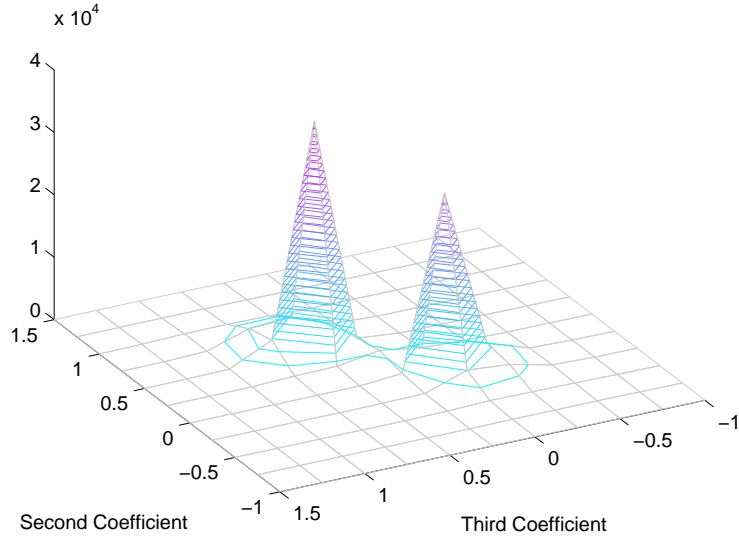
The first constraint for a better interpolation performance is to select the sum of the coefficients to be equal to one. This restriction prevents the signal from being amplified or attenuated.

With the sum restriction in place, experiments on natural images were performed to find out if the cost functions above are able to select appropriate coefficient sets without an additional constraint on the coefficients. Although the results were not impressive, they revealed some useful information about the coefficients.

An important observation was that coefficients much bigger than one or much smaller than zero are unnecessary and even harmful to the interpolation performance. That is why a maximum and minimum limit is set for every coefficient. -1 and 1.5 seemed reasonable choices for testing purposes (Note that their absolute distance to 0.25 is equal to each other). Additionally, a sampling grid of 0.25 is set to select the coefficient sets. It turns out that there are 891 coefficient sets such that the coefficients range from -1 to 1.5 and their sum is always 1. For each interpolated pixel, the best set among these choices is selected according to the first cost function defined above. The performance of this interpolation setting is moderate with some visual artifacts present.

Since it is very costly to search for the best coefficient set among hundreds of choices, another alternative is needed. For this reason, the best coefficient set for each interpolated pixel is saved and K-means clustering is performed both to decrease the number of possible choices and to increase the performance. Indeed, clustering

increased the interpolation quality. However, very interestingly, as the number of clusters is decreased, the interpolation quality kept improving with 16 clusters performing better than 32, and 8 performing better than 16. This observation implies that the additional clusters do not improve the interpolation but degrade it. This led to the conclusion that an additional constraint on the coefficients is needed.



**Figure 5:** Coefficient correlation.

To gain more insight on how the best coefficient sets are distributed and to find out if there is any dependency between them, the coefficients from the best sets are plotted against one another. This examination revealed that there is a correlation between the first and the fourth and between the second and the third coefficients. Coupled with the restriction that their total is one, this means that there is a negative correlation between adjacent coefficients (i.e. the first and the second one, the first and the third one, and so on). Figure 5 illustrates the correlation between the coefficients. X and Y axes denote the values of the second and third coefficients, respectively. Z axis denotes the number of times those coefficient values are selected as part of the best coefficient set. For more than 80 percent of the time, their values are equal to

each other in the best coefficient sets of the test image. The same observation is valid for the other cross coefficient pair, i.e. the first and the fourth. Hence, if we have information about a given coefficient, we can deduce the optimal value of its pair using Bayesian inference. The strong correlation between the cross coefficients tells us that their values should be close to each other with a high probability. We can simplify this relationship and apply it as another restriction by equating cross coefficients to each other:

$$\begin{aligned}\alpha_1 &= \alpha_4 \\ \alpha_2 &= \alpha_3.\end{aligned}\tag{6}$$

Combined with the sum requirement, the equations above imply the following results:

$$\begin{aligned}\alpha_1 &= 0.5 - \alpha_2 \\ \alpha_1 &= 0.5 - \alpha_3 \\ \alpha_4 &= 0.5 - \alpha_2 \\ \alpha_4 &= 0.5 - \alpha_3.\end{aligned}\tag{7}$$

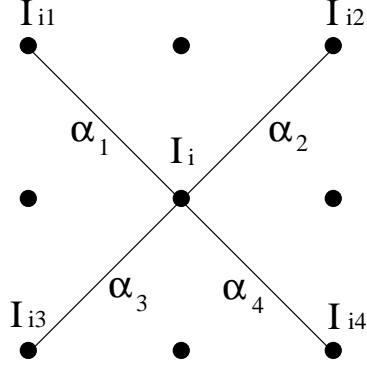
Applying the additional constraints above increases the interpolation quality significantly. Moreover, it is now possible to determine all coefficients when only one of them is given. This is very powerful because finding the best coefficient set according to least squares minimization reduces to a first degree problem. The formula to find the best coefficient set automatically can be derived as follows:

$$TSE = \sum [\alpha_1 I_{i1} + (0.5 - \alpha_1) I_{i2} + (0.5 - \alpha_1) I_{i3} + \alpha_1 I_{i4} - I_i]^2 \tag{8}$$

Rearranging:

$$TSE = \sum [\alpha_1 (I_{i1} + I_{i4} - I_{i2} - I_{i3}) + (0.5 I_{i2} + 0.5 I_{i3} - I_i)]^2$$





**Figure 6:** Four coefficients.

$$TSE = \sum [\alpha_1^2 (I_{i1} + I_{i4} - I_{i2} - I_{i3})^2 + 2\alpha_1 (I_{i1} + I_{i4} - I_{i2} - I_{i3})(0.5I_{i2} + 0.5I_{i3} - I_i) + (0.5I_{i2} + 0.5I_{i3} - I_i)^2]. \quad (9)$$

Taking the derivation of the equation above to find coefficient set that yields the minimum Total Square Error (TSE) gives:

$$\begin{aligned} \sum [2\alpha_1 (I_{i1} + I_{i4} - I_{i2} - I_{i3})^2 + 2(I_{i1} + I_{i4} - I_{i2} - I_{i3})(0.5I_{i2} + 0.5I_{i3} - I_i)] &= 0 \\ \alpha_1 &= \frac{-\sum 2(I_{i1} + I_{i4} - I_{i2} - I_{i3})(0.5I_{i2} + 0.5I_{i3} - I_i)}{\sum 2(I_{i1} + I_{i4} - I_{i2} - I_{i3})^2} \\ \alpha_1 &= \frac{\sum (I_{i1} + I_{i4} - I_{i2} - I_{i3})(I_i - 0.5I_{i2} - 0.5I_{i3})}{\sum (I_{i1} + I_{i4} - I_{i2} - I_{i3})^2}. \end{aligned} \quad (10)$$

The output of the final formula is the first coefficient. The fourth coefficient is equal to the first one and the second and third ones are equal to 0.5 minus the first coefficient.

There are two important points to check in the formula above. The first one is to make sure that the denominator is not zero. This condition is likely to happen in a perfectly smooth region. Hence, the coefficients are all set to 0.25 in this case and the method reduces to bilinear interpolation. The second point is to check the final output of the formula to see if the result is bigger or smaller than some limit. If it is, then it might be better to set the result to that limit. This restriction prevents any

visual artifacts from appearing in the output. The experiments suggest that  $[-1.0, -0.5]$  and  $[1.0, 1.5]$  are appropriate lower and upper limit ranges.

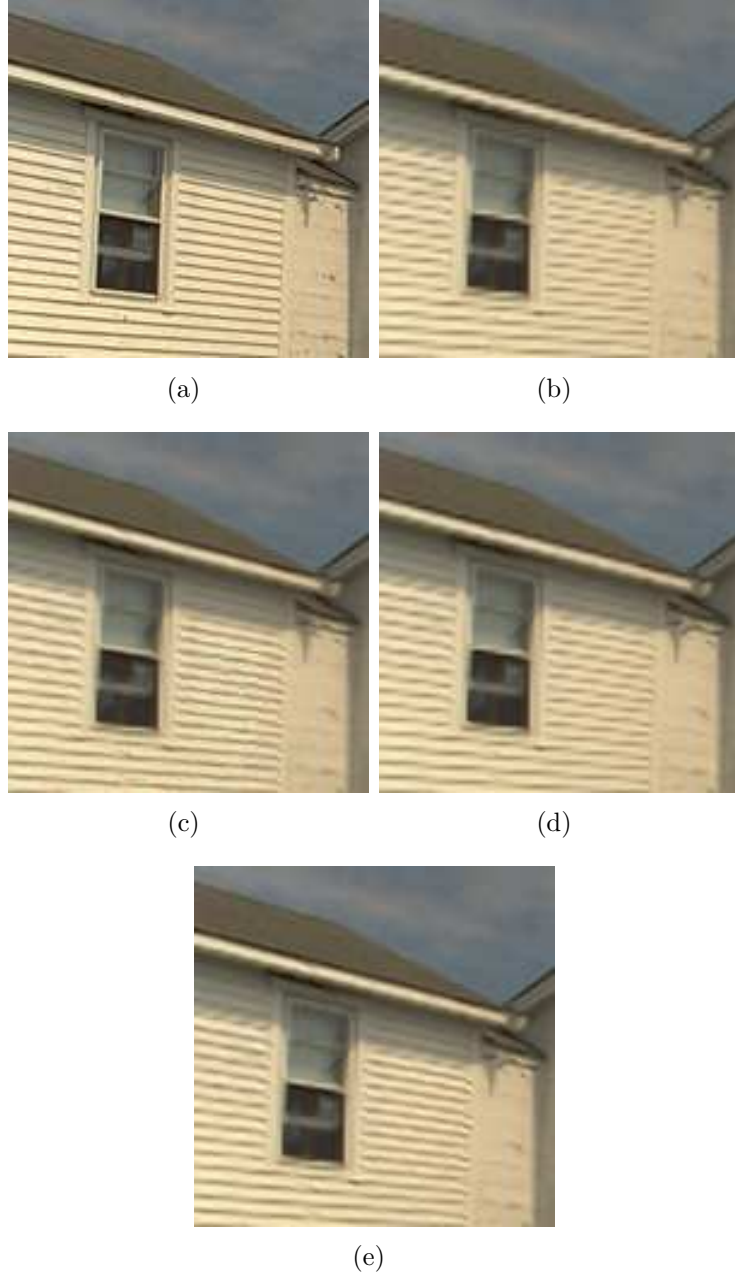
The least squares minimization for the second cost function remains as a third degree problem even with all the constraints. Hence, it is not easy to find the best coefficient set directly in this case. Still, it is possible to optimize the coefficient sets and perform the minimization among them. Comparable interpolation results can be obtained with only 8 or even 4 coefficient sets. The coefficient sets can be optimized by using K-means clustering on the data obtained from natural images.

Based on the chosen cost function, different versions of the proposed interpolation algorithm can be summarized as follows. If the first cost function (for which the pixel values are estimated with Equation (1)) with squared error is chosen, it is possible to find the interpolation coefficients automatically. On the other hand, if the first cost function with absolute error or the second cost function (for which the pixel values are estimated with Equation (5)) is chosen, the interpolation coefficients are found from a set of possible choices. The equality of the coefficients restriction can be relaxed to some degree in this case, since it does not lead to a closed form solution. Overall, the first cost function is preferable to the second one because finding coefficient values with a self-adaptive simple formula is much more convenient than training coefficient sets.

Low computational complexity is almost always desirable for image processing algorithms. However, it is a necessity rather than a convenience for real-time and low power applications. We believe the low computational requirement and high quality of the proposed algorithm makes it a perfect match for any application.

An important advantage of the proposed method is its regularity. Since it does not require edge detection and the same interpolation formula is used for every pixel, the computational cost does not vary with different input images.

The proposed interpolation algorithm can be summarized as follows:



**Figure 7:** Wall detail: original image (a), bilinear interpolation (b), NEDI (c), AQua2 (d), and proposed method (e).

1. Extend the image border by mirroring the pixels near the border.
2. Choose the training window size (2 by 2, 4 by 4, or 6 by 6 are reasonable choices).
3. (Irrelevant for 2 by 2 training window) Choose the weights of training window

pixels in cost calculation (They could be given equal weight for simplicity, or the center pixels could be weighed more to preserve locality).

4. For a pixel to be interpolated, use the following formula derived above to find the interpolation coefficients automatically (refer to Figure 6):

$$\alpha_1 = \frac{\sum((I_{i1} + I_{i4}) - (I_{i2} + I_{i3}))(I_i - 0.5(I_{i2} + I_{i3}))}{\sum((I_{i1} + I_{i4}) - (I_{i2} + I_{i3}))^2}. \quad (11)$$

5. Find the interpolated pixel value using the following formula (refer to Figure 4):

$$\widehat{P}_1 = \alpha_1(I_1 + I_5) + (0.5 - \alpha_1)(I_2 + I_4). \quad (12)$$

6. Fill in all the interpolated pixel values by repeating steps 4-5, first all the diagonal pixels and then the rest.

7. Crop the output image border by twice the extension size used in step 1.

### 3.3 *Experimental Results*

We evaluated the efficacy of the proposed algorithm with the Kodak image set, which consists of 20 images. The results are compared with bilinear interpolation, NEDI algorithm [29], and fast edge directed polynomial interpolation (AQua2) algorithm [35]. Figure 7 shows a sample test image region interpolated with each algorithm for visual quality comparison.

The test images are first low-pass filtered and downsampled by two. Then, they are interpolated by each method and the outputs are compared both objectively and subjectively. For objective comparison, Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) results are used. The proposed method outperforms other edge directed methods in both measures.

Although it is well established that higher PSNR does not necessarily mean better picture quality, PSNR is still the most common measure of objective quality. The proposed method managed to outperform other algorithms for every single image in the test set in terms of PSNR. The comparison results are summarized in Table 1.

**Table 1:** Comparison of PSNR values for different interpolation methods.

image no	Bilinear	NEDI	AQua2	Proposed
1	25.11	24.90	25.02	<b>25.17</b>
2	32.24	32.22	32.31	<b>32.47</b>
3	25.25	25.79	25.82	<b>26.29</b>
4	26.71	26.78	26.77	<b>26.94</b>
5	31.81	32.33	32.35	<b>32.92</b>
6	22.66	22.70	22.73	<b>22.97</b>
7	31.13	31.55	31.37	<b>31.87</b>
8	30.84	31.54	31.38	<b>31.89</b>
9	28.35	28.46	28.47	<b>28.69</b>
10	31.54	31.69	31.68	<b>31.88</b>
11	23.14	23.02	23.09	<b>23.23</b>
12	30.11	30.33	30.31	<b>30.51</b>
13	30.45	30.36	30.44	<b>30.54</b>
14	30.58	30.74	30.92	<b>31.17</b>
15	26.96	26.93	26.99	<b>27.17</b>
16	27.12	27.14	27.27	<b>27.63</b>
17	30.11	30.58	30.47	<b>30.93</b>
18	27.63	27.51	27.62	<b>27.77</b>
19	29.64	29.62	29.64	<b>29.87</b>
20	25.78	25.77	25.76	<b>25.97</b>
mean	28.36	28.50	28.52	<b>28.79</b>

An important observation on the PSNR comparison results is the closeness of average PSNR values of the other two edge directed methods. AQua2 method slightly outperforms NEDI. However, the quality difference between them is no more than 0.02 dB on average. The proposed algorithm, on the other hand, outperforms these methods by 0.27-0.29 dB. The fact that it achieves this quality difference on a large number of assorted typical images as opposed to some singled out extreme cases is remarkable.

**Table 2:** Comparison of SSIM index results for different interpolation methods.

image no	Bilinear	NEDI	AQua2	Proposed
1	<b>0.734</b>	0.717	0.728	0.733
2	0.862	0.857	0.860	<b>0.863</b>
3	0.816	0.823	0.830	<b>0.842</b>
4	0.781	0.780	0.782	<b>0.789</b>
5	0.933	0.937	0.939	<b>0.944</b>
6	0.754	0.750	0.755	<b>0.764</b>
7	0.897	0.899	0.899	<b>0.904</b>
8	0.893	0.901	0.901	<b>0.908</b>
9	0.820	0.818	0.821	<b>0.826</b>
10	0.879	0.875	0.878	<b>0.880</b>
11	0.684	0.670	0.678	<b>0.685</b>
12	0.886	0.886	0.887	<b>0.890</b>
13	0.831	0.828	0.831	<b>0.834</b>
14	0.901	0.905	0.907	<b>0.912</b>
15	0.818	0.810	0.816	<b>0.822</b>
16	0.829	0.829	0.828	<b>0.836</b>
17	0.895	0.896	0.896	<b>0.900</b>
18	0.850	0.844	0.849	<b>0.853</b>
19	0.840	0.832	0.836	<b>0.841</b>
20	0.824	0.820	0.825	<b>0.832</b>
mean	0.836	0.834	0.837	<b>0.843</b>

Structural Similarity Index (SSIM) is a method developed for assessing the similarity between two images [40]. If one of the images is considered to have perfect quality, SSIM index returns the quality of the other image. Its range is from zero to one, one being the perfect match. The proposed algorithm is compared with other

methods in terms of SSIM index to assess the interpolation quality more meaningfully than simple error difference. It outperformed other edge directed methods on every image in the test set in terms of this measure, too. Similarly, it outperformed conventional bilinear interpolation for the most part, with the exception of the first image. AQua2 is the better performing one among the other methods, followed by bilinear interpolation and then NEDI. The SSIM index comparison results are given in Table 2.

## CHAPTER IV

### DEMOSAICING ON THE BAYER PATTERN

#### 4.1 *Edge Strength Filter Based Demosaicing*

##### 4.1.1 A New Filter

The basis of the proposed algorithm is the observation that the constant color difference assumption tends to fail across edges. If one can effectively utilize edge information to avoid averaging non-correlated color differences, demosaicing performance could increase dramatically. To be able to do that, we need to find a way to express the edge information meaningfully at the pixel level so that it is useful enough to improve demosaicing performance. Edge detection filters such as Sobel and Canny can tell whether an edge structure is present at a given pixel. However, they do not provide any information about the sharpness of luminance transition at that particular pixel.

P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>
P <sub>4</sub>	P <sub>5</sub>	P <sub>6</sub>
P <sub>7</sub>	P <sub>8</sub>	P <sub>9</sub>

**Figure 8:** Grayscale pixels.

We propose an edge strength filter that provides local, orientation-free luminance transition information. The filter has a 3 by 3 support size. Given a grayscale input image, it could be formulated as:

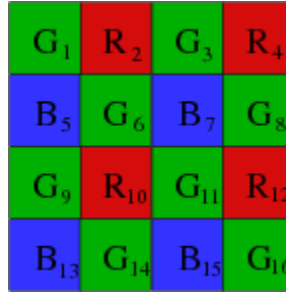
$$ES_{P_5} = \frac{|P_1 - P_9|}{2} + \frac{|P_3 - P_7|}{2} + |P_2 - P_8| + |P_4 - P_6|. \quad (13)$$



By applying the filter to all available pixels, we get the edge strength map of the input image. Note that, although the filter result for a single pixel does not provide any edge direction information, the relationship between neighboring pixel results yields the edge orientation in that neighborhood.

The proposed filter is very useful for finding edges in a grayscale image. However, a mosaicked image only has one of the three color channels available for every pixel location, and it certainly does not have complete luminance information at any pixel. That is why, the edge strength filter can only be applied to a mosaicked image by making an approximation. Instead of trying to estimate luminance information and taking estimated luminance difference of neighboring pixel pairs, we take the difference in terms of the available color channel for each pixel pair. For instance, for the red center pixel case the diagonal differences will come from the blue channel and the rest from the green channel:

$$ES_{R10} = \frac{|B_5 - B_{15}|}{2} + \frac{|B_7 - B_{13}|}{2} + |G_6 - G_{14}| + |G_9 - G_{11}|. \quad (14)$$



**Figure 9:** Bayer CFA pixels.

Similarly, the edge strength for green and blue pixels will be calculated as follows:

$$ES_{B7} = \frac{|R_2 - R_{12}|}{2} + \frac{|R_4 - R_{10}|}{2} + |G_3 - G_{11}| + |G_6 - G_8|$$

$$ES_{G6} = \frac{|G_1 - G_{11}|}{2} + \frac{|G_3 - G_9|}{2} + |R_2 - R_{10}| + |B_5 - B_7|$$

$$ES_{G11} = \frac{|G_6 - G_{16}|}{2} + \frac{|G_8 - G_{14}|}{2} + |B_7 - B_{15}| + |R_{10} - R_{12}|. \quad (15)$$

Figure 10 shows the mosaicked lighthouse image and its edge strength filter result. The edge strength map obtained from the mosaicked input image will help us both in initial green channel interpolation stage and in subsequent green channel update.



**Figure 10:** Mosaicked lighthouse image and its edge strength filter output.

#### 4.1.2 Initial Green Channel Interpolation

We propose making a hard decision based on the edge strength filter described above. For this purpose, every green pixel to be interpolated (red or blue pixel in the mosaicked image) is marked either horizontal or vertical by comparing the edge strength differences along each direction on a local window. For a window size of 5 by 5,

horizontal and vertical difference costs can be formulated as follows:

$$\begin{aligned}
HC_{i,j} &= \sum_{m=-2}^2 \left( \sum_{n=-2}^1 (ES_{i+m,j+n} - ES_{i+m,j+n+1}) \right) \\
VC_{i,j} &= \sum_{m=-2}^1 \left( \sum_{n=-2}^2 (ES_{i+m,j+n} - ES_{i+m+1,j+n}) \right),
\end{aligned} \tag{16}$$

where  $ES_{i,j}$  is the edge strength filter output at pixel location  $(i, j)$ .

The target pixel will be labeled horizontal if horizontal cost is less than vertical and vice versa. The rationale behind this decision scheme is that if there happens to be a horizontal edge in a given neighborhood, then the edge strength differences between vertical neighbors will vary more than those of horizontal neighbors. After all the pixels are labeled, the robustness of the direction decision can be improved by relabeling them based on the directions of their neighbors. For instance, considering the closest 8 neighbors of a target pixel and the pixel itself, the pixel will be labeled horizontal only if more than 4 of those 9 pixels are initially labeled horizontal.

Based on the final direction label, green channel is interpolated using the following formulas:

$$\tilde{G}_{i,j} = \begin{cases} B_{i,j} + \frac{\tilde{G}_{i,j}^H - B_{i,j}}{2} + \frac{G_{i,j-1} - \tilde{B}_{i,j-1}^H}{4} + \frac{G_{i,j+1} - \tilde{B}_{i,j+1}^H}{4}, & \text{if Horizontal} \\ B_{i,j} + \frac{\tilde{G}_{i,j}^V - B_{i,j}}{2} + \frac{G_{i-1,j} - \tilde{B}_{i-1,j}^V}{4} + \frac{G_{i+1,j} - \tilde{B}_{i+1,j}^V}{4}, & \text{if Vertical} \end{cases} \tag{17}$$

where directional estimations are calculated by:

$$\begin{aligned}
\tilde{G}_{i,j}^H &= \frac{G_{i,j-1} + G_{i,j+1}}{2} + \frac{2 * B_{i,j} - B_{i,j-2} - B_{i,j+2}}{4} \\
\tilde{G}_{i,j}^V &= \frac{G_{i-1,j} + G_{i+1,j}}{2} + \frac{2 * B_{i,j} - B_{i-2,j} - B_{i+2,j}}{4} \\
\tilde{B}_{i,j}^H &= \frac{B_{i,j-1} + B_{i,j+1}}{2} + \frac{2 * G_{i,j} - G_{i,j-2} - G_{i,j+2}}{4} \\
\tilde{B}_{i,j}^V &= \frac{B_{i-1,j} + B_{i+1,j}}{2} + \frac{2 * G_{i,j} - G_{i-2,j} - G_{i+2,j}}{4}.
\end{aligned} \tag{18}$$

Green channel estimation for red pixel locations is performed with the same formulas simply by replacing  $B$ 's with  $R$ 's.

#### 4.1.3 Green Channel Update

The second step of the proposed algorithm is updating the green channel. We make use of the constant color difference assumption combined with the edge strength filter to improve the initial green channel interpolation while avoiding averaging across edge structures. For every green pixel to be updated, the closest four neighbors with available color difference estimates are considered. The weight for each neighbor is inversely correlated with the total absolute edge strength difference in its direction. In other words, a neighbor will contribute less to the update result if there happens to be a strong edge between the target pixel and itself. Assuming we are updating the green channel value at a blue pixel:

$$\begin{aligned}
D_1 &= \frac{1}{|ES_{i,j} - ES_{i-1,j}| + |ES_{i-1,j} - ES_{i-2,j}| + |ES_{i-2,j} - ES_{i-3,j}| + C_1} \\
D_2 &= \frac{1}{|ES_{i,j} - ES_{i,j-1}| + |ES_{i,j-1} - ES_{i,j-2}| + |ES_{i,j-2} - ES_{i,j-3}| + C_1} \\
D_3 &= \frac{1}{|ES_{i,j} - ES_{i,j+1}| + |ES_{i,j+1} - ES_{i,j+2}| + |ES_{i,j+2} - ES_{i,j+3}| + C_1} \\
D_4 &= \frac{1}{|ES_{i,j} - ES_{i+1,j}| + |ES_{i+1,j} - ES_{i+2,j}| + |ES_{i+2,j} - ES_{i+3,j}| + C_1} \\
D_{Total} &= D_1 + D_2 + D_3 + D_4 \\
\hat{G}_{i,j} &= B_{i,j} + W_1(\tilde{G}_{i,j} - B_{i,j}) + W_2 \left[ \frac{D_1}{D_{Total}}(\tilde{G}_{i-2,j} - B_{i-2,j}) + \right. \\
&\quad \left. \frac{D_2}{D_{Total}}(\tilde{G}_{i,j-2} - B_{i,j-2}) + \frac{D_3}{D_{Total}}(\tilde{G}_{i,j+2} - B_{i,j+2}) + \frac{D_4}{D_{Total}}(\tilde{G}_{i+2,j} - B_{i+2,j}) \right] \\
W_1 + W_2 &= 1.
\end{aligned} \tag{19}$$

Again, the green channel values at red pixel locations are updated in the same way by replacing  $B$ 's with  $R$ 's in the formulas above.  $\hat{G}_{i,j}$  stands for the updated

green channel result while  $\tilde{G}_{i,j}$  is the initial green channel interpolation.  $C_1$  is a nonzero constant to avoid zero denominator. Updating the green channel reduces color artifacts and improves PSNR. However, zipper artifacts become more prominent as the number of updates increase. Experiments on test images suggest that one or two green channel updates are adequate.

The performance of green channel update can be improved further by making  $W_1$  adaptive for each pixel by checking the total absolute difference between the closest known green pixels. The idea is that the green channel update should be more aggressive if there happens to be a lot of difference between known green pixels in that neighborhood because the initial interpolation is more likely to fail in such areas. The update formulas with adaptive weights are as follows:

$$GD_{i,j} = \min\left(\frac{|G_{i-1,j} - G_{i,j+1}| + |G_{i,j+1} - G_{i+1,j}|}{4} + \frac{|G_{i+1,j} - G_{i,j-1}| + |G_{i,j-1} - G_{i-1,j}|}{4}, C_2\right)$$

$$\tilde{G}'_{i,j} = B_{i,j} + (W_1 - GD_{i,j})(\tilde{G}_{i,j} - B_{i,j}) + (W_2 + GD_{i,j})\left[\frac{D_1}{D_{Total}}(\tilde{G}_{i-2,j} - B_{i-2,j}) + \frac{D_2}{D_{Total}}(\tilde{G}_{i,j-2} - B_{i,j-2}) + \frac{D_3}{D_{Total}}(\tilde{G}_{i,j+2} - B_{i,j+2}) + \frac{D_4}{D_{Total}}(\tilde{G}_{i+2,j} - B_{i+2,j})\right]$$

$$W_1 + W_2 = 1. \quad (20)$$

#### 4.1.4 Red and Blue Channel Interpolation

Once the green channel interpolation is finalized, we fill in red and blue channels using constant color difference assumption. For the red channel interpolation at blue pixels and the blue channel interpolation at red pixels, the diagonal neighbors are used adaptively based on the green channel gradients in both directions:

$$D_1 = \frac{1}{|\hat{G}_{i-2,j-2} - \hat{G}_{i,j}| + |\hat{G}_{i-1,j-1} - \hat{G}_{i+1,j+1}| + |\hat{G}_{i,j} - \hat{G}_{i+2,j+2}|}$$

$$D_2 = \frac{1}{|\hat{G}_{i-2,j+2} - \hat{G}_{i,j}| + |\hat{G}_{i-1,j+1} - \hat{G}_{i+1,j-1}| + |\hat{G}_{i,j} - \hat{G}_{i+2,j-2}|}$$

$$D_{Total} = D_1 + D_2. \quad (21)$$

If coordinate  $(i, j)$  is a red pixel location, the blue channel estimation is calculated by:

$$\begin{aligned} \hat{B}_{i,j} = & \hat{G}_{i,j} - \frac{D_1 * (\hat{G}_{i-1,j-1} - B_{i-1,j-1} + \hat{G}_{i+1,j+1} - B_{i+1,j+1})}{2 * D_{Total}} \\ & - \frac{D_2 * (\hat{G}_{i-1,j+1} - B_{i-1,j+1} + \hat{G}_{i+1,j-1} - B_{i+1,j-1})}{2 * D_{Total}}. \end{aligned} \quad (22)$$

Similarly, if it is a blue pixel location, the red channel estimation is:

$$\begin{aligned} \hat{R}_{i,j} = & \hat{G}_{i,j} - \frac{D_1 * (\hat{G}_{i-1,j-1} - R_{i-1,j-1} + \hat{G}_{i+1,j+1} - R_{i+1,j+1})}{2 * D_{Total}} \\ & - \frac{D_2 * (\hat{G}_{i-1,j+1} - R_{i-1,j+1} + \hat{G}_{i+1,j-1} - R_{i+1,j-1})}{2 * D_{Total}}. \end{aligned} \quad (23)$$

For the red and blue channel estimation at green pixels, we employ bilinear interpolation over color differences since the considered adaptive approaches do not provide any performance gain. Here, only the closest two neighbours for which the original pixel value available are used:

$$\begin{aligned} \hat{B}_{2i,2j} &= G_{2i,2j} - \frac{(\hat{G}_{2i-1,2j} - B_{2i-1,2j}) + (\hat{G}_{2i+1,2j} - B_{2i+1,2j})}{2} \\ \hat{B}_{2i+1,2j+1} &= G_{2i+1,2j+1} - \frac{(\hat{G}_{2i+1,2j} - B_{2i+1,2j}) + (\hat{G}_{2i+1,2j+2} - B_{2i+1,2j+2})}{2} \\ \hat{R}_{2i,2j} &= G_{2i,2j} - \frac{(\hat{G}_{2i,2j-1} - R_{2i,2j-1}) + (\hat{G}_{2i,2j+1} - R_{2i,2j+1})}{2} \\ \hat{R}_{2i+1,2j+1} &= G_{2i+1,2j+1} - \frac{(\hat{G}_{2i,2j+1} - R_{2i,2j+1}) + (\hat{G}_{2i+2,2j+1} - R_{2i+2,2j+1})}{2}. \end{aligned} \quad (24)$$

By the end of this step, we filled in all the missing color channel values in the input image. We utilized a simple edge strength filter both to determine the initial green channel interpolation direction and to avoid applying the constant color difference rule across edge structures.

## ***4.2 Color Difference Gradients Based Demosaicing***

### **4.2.1 Algorithm Background**

We developed the color difference gradients based demosaicing algorithm by addressing a few limitations we observed in the DLMMSE method [42]. Firstly, as a result of its directional nature, the DLMMSE algorithm uses only a subset of a target pixel’s neighbors (pixels that share the same column or row with the target pixel) to find out how much each direction should contribute to the color difference calculation. Although the solution is optimal in its own domain, unconsidered neighbor pixels might provide additional information that could improve the color difference estimation. That is why, we want to include every neighbor pixel inside a given local window to the decision making process. However, since available color difference at every pixel is either (G-R) or (G-B), we cannot apply the variance metric as the DLMMSE method does. For this reason, we use gradients of color differences to come up with weights for each direction.

Secondly, the DLMMSE method operates on horizontal and vertical directions like other directional methods. However, for a given direction, the conditions might be different for pixels falling to the opposite sides of the target pixel especially near edges or in texture regions. For this reason, we decouple north-south and east-west directions from each other and consider them separately. Instead of making a hard direction decision, we combine estimations from every direction, which eliminates the need for setting thresholds.

### **4.2.2 Green Channel Interpolation**

The proposed algorithm first interpolates the green channel in a single run, then uses its results to fill in red and blue channels. The first step of the algorithm is applying Hamilton and Adams’ [14] interpolation formula to all pixels in both vertical and horizontal directions. For red pixel locations, horizontal and vertical green channel

estimations are calculated as follows:

$$\begin{aligned}\tilde{G}_{i,j}^H &= (G_{i,j-1} + G_{i,j+1})/2 + (2 * R_{i,j} - R_{i,j-2} - R_{i,j+2})/4 \\ \tilde{G}_{i,j}^V &= (G_{i-1,j} + G_{i+1,j})/2 + (2 * R_{i,j} - R_{i-2,j} - R_{i+2,j})/4.\end{aligned}\quad (25)$$

Similarly, for green pixels with red vertical neighbors, vertical red channel estimation is:

$$\tilde{R}_{i,j}^V = (R_{i-1,j} + R_{i+1,j})/2 + (2 * G_{i,j} - G_{i-2,j} - G_{i+2,j})/4. \quad (26)$$

And for green pixels with horizontal red channel neighbors,

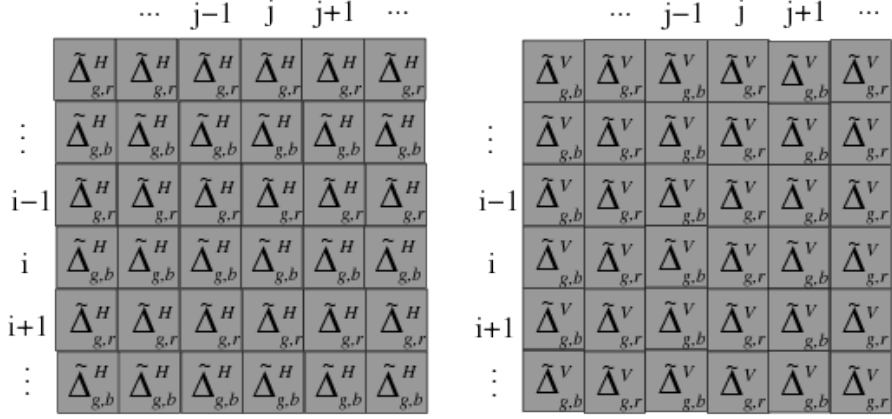
$$\tilde{R}_{i,j}^H = (R_{i,j-1} + R_{i,j+1})/2 + (2 * G_{i,j} - G_{i,j-2} - G_{i,j+2})/4. \quad (27)$$

Estimations with blue pixels are calculated in the same manner, simply by replacing  $R$  with  $B$  in the formulas above. The next step is to find horizontal and vertical color difference estimations using original and directionally estimated pixel values.

$$\begin{aligned}\tilde{\Delta}_{g,r}^V(i,j) &= \begin{cases} \tilde{G}_{i,j}^V - R_{i,j}, & \text{G is interpolated} \\ G_{i,j} - \tilde{R}_{i,j}^V, & \text{R is interpolated} \end{cases} \\ \tilde{\Delta}_{g,r}^H(i,j) &= \begin{cases} \tilde{G}_{i,j}^H - R_{i,j}, & \text{G is interpolated} \\ G_{i,j} - \tilde{R}_{i,j}^H, & \text{R is interpolated} \end{cases}\end{aligned}\quad (28)$$

Again, horizontal and vertical (G-B) difference estimations are calculated similarly. By the end of this step, we have two difference maps, one for horizontal and the other one for vertical estimations as shown in Figure 11.





**Figure 11:** Horizontal and vertical color difference maps.

Next, directional color differences are combined to form the final difference estimation for the target pixel:

$$\begin{aligned}\tilde{\Delta}_{g,r}(i, j) = & [w_N * f * \tilde{\Delta}_{g,r}^V(i - 4 : i, j) + \\ & w_S * f * \tilde{\Delta}_{g,r}^V(i : i + 4, j) + \\ & w_E * \tilde{\Delta}_{g,r}^H(i, j - 4 : j) * f' + \\ & w_W * \tilde{\Delta}_{g,r}^H(i, j : j + 4) * f'] / w_T\end{aligned}$$

$$w_T = w_N + w_S + w_E + w_W$$

$$f = [1 \ 1 \ 1 \ 1 \ 1] / 5. \quad (29)$$

The vector  $f$  could be modified to put more weight to color differences closer to the target pixel. The weight for each direction ( $w_N, w_S, w_E, w_W$ ) is calculated by adding color difference gradients in that direction over a local window. For a window size of 5 by 5:

$$\begin{aligned}w_N &= 1 / \left( \sum_{a=i-4}^i \sum_{b=j-2}^{j+2} D_{a,b}^V \right)^2 \\ w_S &= 1 / \left( \sum_{a=i}^{i+4} \sum_{b=j-2}^{j+2} D_{a,b}^V \right)^2\end{aligned}$$

$$\begin{aligned}
w_E &= 1 / \left( \sum_{a=i-2}^{i+2} \sum_{b=j-4}^j D_{a,b}^H \right)^2 \\
w_W &= 1 / \left( \sum_{a=i-2}^{i+2} \sum_{b=j}^{j+4} D_{a,b}^H \right)^2,
\end{aligned} \tag{30}$$

where gradients are defined as:

$$\begin{aligned}
D_{i,j}^V &= |\tilde{\Delta}_{i-1,j}^V - \tilde{\Delta}_{i+1,j}^V| \\
D_{i,j}^H &= |\tilde{\Delta}_{i,j-1}^H - \tilde{\Delta}_{i,j+1}^H|.
\end{aligned} \tag{31}$$

Finally, the target green pixel value is calculated by adding the estimated color difference to the available (red or blue) target pixel:

$$\begin{aligned}
\tilde{G}(i, j) &= R(i, j) + \tilde{\Delta}_{g,r}(i, j) \\
\tilde{G}(i, j) &= B(i, j) + \tilde{\Delta}_{g,b}(i, j).
\end{aligned} \tag{32}$$

#### 4.2.3 Red and Blue Channel Interpolation

After the green channel processing is finished, we start filling in red and blue channels. The red pixel values at blue locations and the blue pixel values at red locations are interpolated using the following filter that was proposed in [37]:

$$p_{rb} = \begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 10 & 0 & 10 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 10 & 0 & 10 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 0 \end{bmatrix} * \frac{1}{32}$$

$$\begin{aligned}
\tilde{R}_{i,j} &= \tilde{G}_{i,j} - \tilde{\Delta}_{g,r}(i-3:i+3, j-3:j+3) \otimes p_{rb} \\
\tilde{B}_{i,j} &= \tilde{G}_{i,j} - \tilde{\Delta}_{g,b}(i-3:i+3, j-3:j+3) \otimes p_{rb},
\end{aligned} \tag{33}$$

where  $\otimes$  denotes element-wise matrix multiplication and then summation of elements.

For red and blue pixels at green locations, we use bilinear interpolation over the closest four neighbors. The immediate vertical neighbors of a green pixel are either red or blue pixels. For the red pixel case, red and blue pixel values at a green pixel location are interpolated as follows:

$$\begin{aligned}
\tilde{R}(i, j) &= G(i, j) - (\tilde{G}_{i-1, j} - R_{i-1, j})/4 - (\tilde{G}_{i+1, j} - R_{i+1, j})/4 \\
&\quad - (\tilde{G}_{i, j-1} - \tilde{R}_{i, j-1})/4 - (\tilde{G}_{i, j+1} - \tilde{R}_{i, j+1})/4 \\
\tilde{B}(i, j) &= G(i, j) - (\tilde{G}_{i-1, j} - \tilde{B}_{i-1, j})/4 - (\tilde{G}_{i+1, j} - \tilde{B}_{i+1, j})/4 \\
&\quad - (\tilde{G}_{i, j-1} - B_{i, j-1})/4 - (\tilde{G}_{i, j+1} - B_{i, j+1})/4.
\end{aligned} \tag{34}$$

The interpolation formulas are similar for the blue vertical neighbor case. At this point, we interpolated the missing pixels for every channel and reconstructed our color image.

### 4.3 *Multiscale Gradients Based Demosaicing*

#### 4.3.1 Algorithm Background

Gradients are useful for extracting directional data from digital images. Several demosaicing methods including a recent integrated gradients method proposed in [10] made use of them. We demonstrated in [38] that the gradients of color difference signals could be valuable features to adaptively combine directional color difference estimates. In this method, the horizontal and vertical color difference estimates are blended based on the ratio of the total absolute values of vertical and horizontal color difference gradients over a local window.

The first step of the algorithm is to get initial directional color channel estimates. The Bayer pattern is comprised of blue&green and red&green rows and columns as

depicted in Figure 1. For red&green rows and columns in the input mosaic image, the directional estimates for the missing red and green pixel values are:

$$\begin{aligned}
\tilde{G}^H(i, j) &= \frac{G(i, j-1) + G(i, j+1)}{2} + \frac{2.R(i, j) - R(i, j-2) - R(i, j+2)}{4} \\
\tilde{R}^H(i, j) &= \frac{R(i, j-1) + R(i, j+1)}{2} + \frac{2.G(i, j) - G(i, j-2) - G(i, j+2)}{4} \\
\tilde{G}^V(i, j) &= \frac{G(i-1, j) + G(i+1, j)}{2} + \frac{2.R(i, j) - R(i-2, j) - R(i+2, j)}{4} \\
\tilde{R}^V(i, j) &= \frac{R(i-1, j) + R(i+1, j)}{2} + \frac{2.G(i, j) - G(i-2, j) - G(i+2, j)}{4}, \quad (35)
\end{aligned}$$

where  $H$  and  $V$  denote horizontal and vertical directions and  $(i, j)$  is the pixel location. For every pixel coordinate, we now have a true color channel value and two directional estimates. By taking their difference, we get the directional color difference estimate:

$$\begin{aligned}
\tilde{\Delta}_{g,r}^H(i, j) &= \begin{cases} \tilde{G}^H(i, j) - R(i, j), & \text{if G is interpolated} \\ G(i, j) - \tilde{R}^H(i, j), & \text{if R is interpolated} \end{cases} \\
\tilde{\Delta}_{g,r}^V(i, j) &= \begin{cases} \tilde{G}^V(i, j) - R(i, j), & \text{if G is interpolated} \\ G(i, j) - \tilde{R}^V(i, j), & \text{if R is interpolated} \end{cases}
\end{aligned} \quad (36)$$

where  $\tilde{\Delta}_{g,r}^H$  stands for the horizontal difference estimate between green and red channels. The equations are similar for blue&green rows and columns. The generated horizontal and vertical color difference maps are shown in Figure 11. As mentioned above, the directional estimates are combined adaptively using the color difference gradients. The absolute color difference gradients at pixel coordinates  $(i, j)$  are given by:

$$\begin{aligned}
D^H(i, j) &= |\tilde{\Delta}^H(i, j-1) - \tilde{\Delta}^H(i, j+1)| \\
D^V(i, j) &= |\tilde{\Delta}^V(i-1, j) - \tilde{\Delta}^V(i+1, j)|.
\end{aligned} \quad (37)$$

It could be argued that the performance of such an algorithm relies on its ability to successfully combine directional estimates. The color difference gradients calculated above are used to find weights for each direction. The horizontal color difference gradient equation above can be written in terms of red and green pixel values as follows:

$$\begin{aligned}
D^H(i, j) &= |(G(i, j-1) - \tilde{R}^H(i, j-1)) - (G(i, j+1) - \tilde{R}^H(i, j+1))| \\
&= |(\frac{2.G(i, j-1) + G(i, j-3) + G(i, j+1)}{4} - \frac{R(i, j-2) + R(i, j)}{2}) - \\
&\quad (\frac{2.G(i, j+1) + G(i, j-1) + G(i, j+3)}{4} - \frac{R(i, j) + R(i, j+2)}{2})|. \tag{38}
\end{aligned}$$

We observe that there are  $R(i, j)$  terms present and they cancel out each other. Rearranging with respect to different color channels leaves us with:

$$\begin{aligned}
D^H(i, j) &= |\frac{R(i, j+2) - R(i, j-2)}{2} - \\
&\quad \frac{(G(i, j+3) + G(i, j+1)) - (G(i, j-3) + G(i, j-1))}{4}|. \tag{39}
\end{aligned}$$

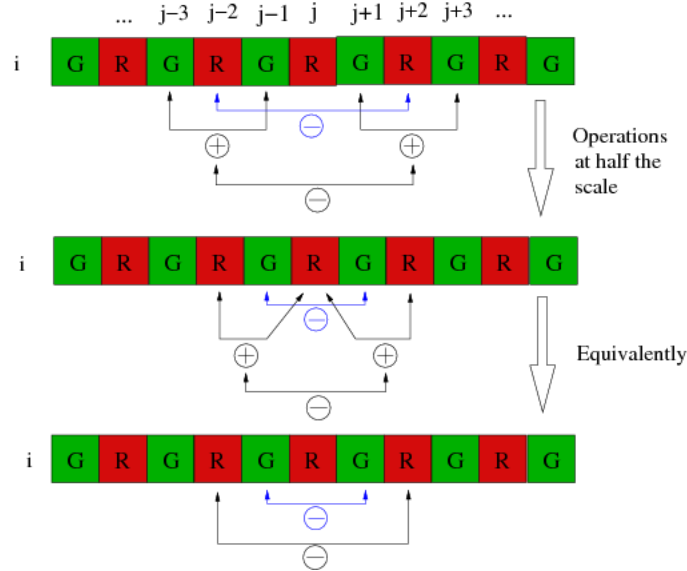
There are two important observations that we made on the equation above. First, what our color difference gradient corresponds to is taking the difference between the available color channel values two pixels away from the target pixel, doing the same operation in terms of the other color channel by using simple averaging, and then finding the difference between these two operations as illustrated in the top portion of Figure 12. If these two color channels are changing in parallel with each other along this direction, then the resulting absolute value would be small. On the other hand, if there is an abrupt color change, then the result would be large and the color difference estimate along this direction would be given a small weight in combined color difference calculation. Our second and more important observation is that, we can do these same operations at half the scale:

$$D^h(i, j) = |\frac{G(i, j+1) - G(i, j-1)}{2} - \frac{(R(i, j+2) + R(i, j)) - (R(i, j-2) + R(i, j))}{4}|, \tag{40}$$

where  $D^h(i, j)$  denotes the horizontal estimation at half the scale. A smaller scale is more desirable because it allows the local color dynamics to be captured at a better resolution. Note that the available color channels are replaced at this scale, but we are still performing the same operations: We take the difference between the available color channel values one pixel (instead of two pixels) away from the target pixel, we do the same operation in terms of the other channel by using its closest samples, and then we take the difference between these two. At this scale, the  $R(i, j)$  terms cancel each other out and we are left with:

$$D^h(i, j) = \left| \frac{G(i, j+1) - G(i, j-1)}{2} - \frac{R(i, j+2) - R(i, j-2)}{4} \right|. \quad (41)$$

We observe that the first part of this equation is the green channel gradient, and the second part is the red channel gradient at twice the scale normalized by the distance between their operands as shown in the bottom part of Figure 12.



**Figure 12:** Relationship between the color difference gradients equation and the multiscale gradients equation.

Like the color difference gradient equation (Equation no. 39), this equation checks whether different color channel pixels along this direction are changing in agreement

with each other or not. However, we expected this new equation to be more successful with combining the directional estimates because we capture the color dynamics at a more local level and we do it without resorting to any simple averaging.

The fact that this equation combines two different scales of gradients together gave us the idea that it should be possible to incorporate even more scales into the equation. However, since the locality will get weaker with each additional scale, the larger scales should contribute less to the result. The easiest way of doing that is to optimize the normalizing terms in the denominators. The final multiscale gradients equations for red&green rows and columns can be given as follows:

$$\begin{aligned}
D^h(i, j) = & \left| \frac{G(i, j+1) - G(i, j-1)}{2} - \frac{R(i, j+2) - R(i, j-2)}{N_1} + \right. \\
& \left. \frac{G(i, j+3) - G(i, j-3)}{N_2} - \frac{R(i, j+4) - R(i, j-4)}{N_3} + \dots \right| \\
D^v(i, j) = & \left| \frac{G(i+1, j) - G(i-1, j)}{2} - \frac{R(i+2, j) - R(i-2, j)}{N_1} + \right. \\
& \left. \frac{G(i+3, j) - G(i-3, j)}{N_2} - \frac{R(i+4, j) - R(i-4, j)}{N_3} + \dots \right|, \tag{42}
\end{aligned}$$

where the  $N_i$  terms are the normalizers. The equations are similar for blue&green rows and columns.

#### 4.3.2 Initial Green Channel Interpolation

Like most demosaicing methods designed for the Bayer pattern including our methods described in previous sections, the multiscale gradients based algorithm starts with interpolating the green channel. After updating the initial green channel interpolation results in one pass, the red and blue channels are filled in using the constant color difference assumption. The ratio between the vertical and horizontal multiscale gradients results over a local window is employed at every stage.

For initial green channel interpolation, we have directional color difference estimates around every green pixel to be interpolated as given in Equation (36) and we combine them adaptively:

$$\begin{aligned}
\hat{\Delta}_{g,r}(i, j) &= [w_V \cdot \mathbf{f} \cdot \tilde{\Delta}_{g,r}^V(i-1 : i+1, j) + \\
&\quad w_H \cdot \tilde{\Delta}_{g,r}^H(i, j-1 : j+1) \cdot \mathbf{f}'] / w_T \\
w_T &= w_V + w_H \\
\mathbf{f} &= [1/4 \ 2/4 \ 1/4].
\end{aligned} \tag{43}$$

For a local window size of 5 by 5, the weight for each direction is calculated as follows:

$$\begin{aligned}
w_V &= 1 / \left( \sum_{k=i-2}^{i+2} \sum_{l=j-2}^{j+2} D_{k,l}^v \right)^2 \\
w_H &= 1 / \left( \sum_{k=i-2}^{i+2} \sum_{l=j-2}^{j+2} D_{k,l}^h \right)^2.
\end{aligned} \tag{44}$$

The division operation can be avoided by defining the weights as the denominators and exchanging them (The ratio of  $1/a$  to  $1/b$  is equal to the ratio of  $b$  to  $a$  provided that both are nonzero).

### 4.3.3 Green Channel Update

After the directional color difference estimates are combined as explained in the previous section, we can directly calculate the green channel and move onto completing the other channels. However, it is possible to improve the green channel results by updating the initial color difference estimates. We consider the closest four neighbors to the target pixel with each one having its own weight:

$$\begin{aligned}
\tilde{\Delta}_{g,r}(i, j) &= \hat{\Delta}_{g,r}(i, j) \cdot (1 - w) + \\
&\quad [w_N \cdot \hat{\Delta}_{g,r}(i-2, j) + \\
&\quad w_S \cdot \hat{\Delta}_{g,r}(i+2, j) + \\
&\quad w_E \cdot \hat{\Delta}_{g,r}(i, j-2) + \\
&\quad w_W \cdot \hat{\Delta}_{g,r}(i, j+2)] \cdot w / w_T
\end{aligned}$$



$$w_T = w_N + w_S + w_E + w_W. \quad (45)$$

Again, the weights  $(w_N, w_S, w_E, w_W)$  are calculated by finding the total multiscale color gradients over a local window. For a 3 by 5 window for horizontal and a 5 by 3 window for vertical components, the weight calculations can be given as follows:

$$\begin{aligned} w_N &= 1 / \left( \sum_{k=i-4}^i \sum_{l=j-1}^{j+1} D_{k,l}^v \right)^2 \\ w_S &= 1 / \left( \sum_{k=i}^{i+4} \sum_{l=j-1}^{j+1} D_{k,l}^v \right)^2 \\ w_W &= 1 / \left( \sum_{k=i-1}^{i+1} \sum_{l=j-4}^j D_{k,l}^h \right)^2 \\ w_E &= 1 / \left( \sum_{k=i-1}^{i+1} \sum_{l=j}^{j+4} D_{k,l}^h \right)^2. \end{aligned} \quad (46)$$

Once the color difference estimate is finalized, we add it to the available target pixel to obtain the estimated green channel value:

$$\begin{aligned} \tilde{G}(i, j) &= R(i, j) + \tilde{\Delta}_{g,r}(i, j) \\ \tilde{G}(i, j) &= B(i, j) + \tilde{\Delta}_{g,b}(i, j). \end{aligned} \quad (47)$$

#### 4.3.4 Red and Blue Channel Interpolation

For red and blue channel interpolation, we first complete the missing diagonal samples i.e. red pixel values at blue locations and blue pixel values at red locations. These pixels are interpolated using the 7 by 7 filter proposed in [37]:

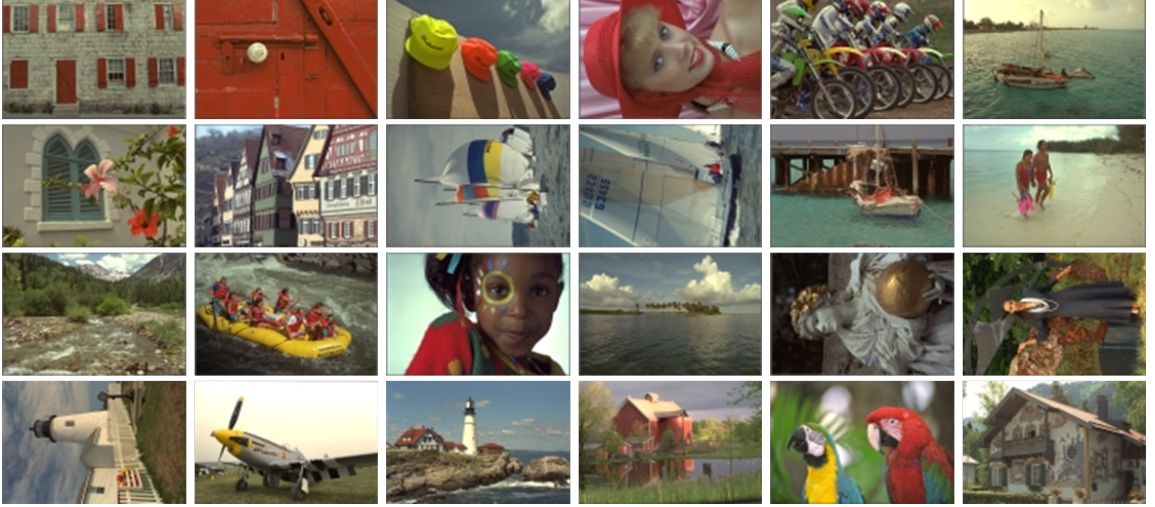
$$\begin{aligned}
p_{rb} &= \begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 10 & 0 & 10 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 10 & 0 & 10 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 0 \end{bmatrix} \cdot \frac{1}{32} \\
\tilde{R}_{i,j} &= \tilde{G}_{i,j} - \tilde{\Delta}_{g,r}(i-3:i+3, j-3:j+3) \otimes p_{rb} \\
\tilde{B}_{i,j} &= \tilde{G}_{i,j} - \tilde{\Delta}_{g,b}(i-3:i+3, j-3:j+3) \otimes p_{rb}, \tag{48}
\end{aligned}$$

where  $\otimes$  denotes element-wise matrix multiplication and subsequent summation.

The red and blue pixels at green locations are interpolated adaptively. In order to avoid repetitive weight calculations, we reuse the directional weights  $(w_H, w_V)$  defined in Equation (44). The immediate vertical neighbors of a green pixel are either red or blue pixels. For the red pixel case the interpolation is carried out as follows:

$$\begin{aligned}
\tilde{R}(i, j) &= G(i, j) - \frac{w_V \cdot (\tilde{G}(i-1, j) - R(i-1, j) + \tilde{G}(i+1, j) - R(i+1, j))}{2 \cdot (w_V + w_H)} \\
&\quad - \frac{w_H \cdot (\tilde{G}(i, j-1) - \tilde{R}(i, j-1) + \tilde{G}(i, j+1) - \tilde{R}(i, j+1))}{2 \cdot (w_V + w_H)} \\
\tilde{B}(i, j) &= G(i, j) - \frac{w_V \cdot (\tilde{G}(i-1, j) - \tilde{B}(i-1, j) + \tilde{G}(i+1, j) - \tilde{B}(i+1, j))}{2 \cdot (w_V + w_H)} \\
&\quad - \frac{w_H \cdot (\tilde{G}(i, j-1) - B(i, j-1) + \tilde{G}(i, j+1) - B(i, j+1))}{2 \cdot (w_V + w_H)}. \tag{49}
\end{aligned}$$

The equations for the blue vertical neighbor case are similar. With the completion of red and blue pixel values at green coordinates, we obtain the full color image.



**Figure 13:** 24 image Kodak dataset.

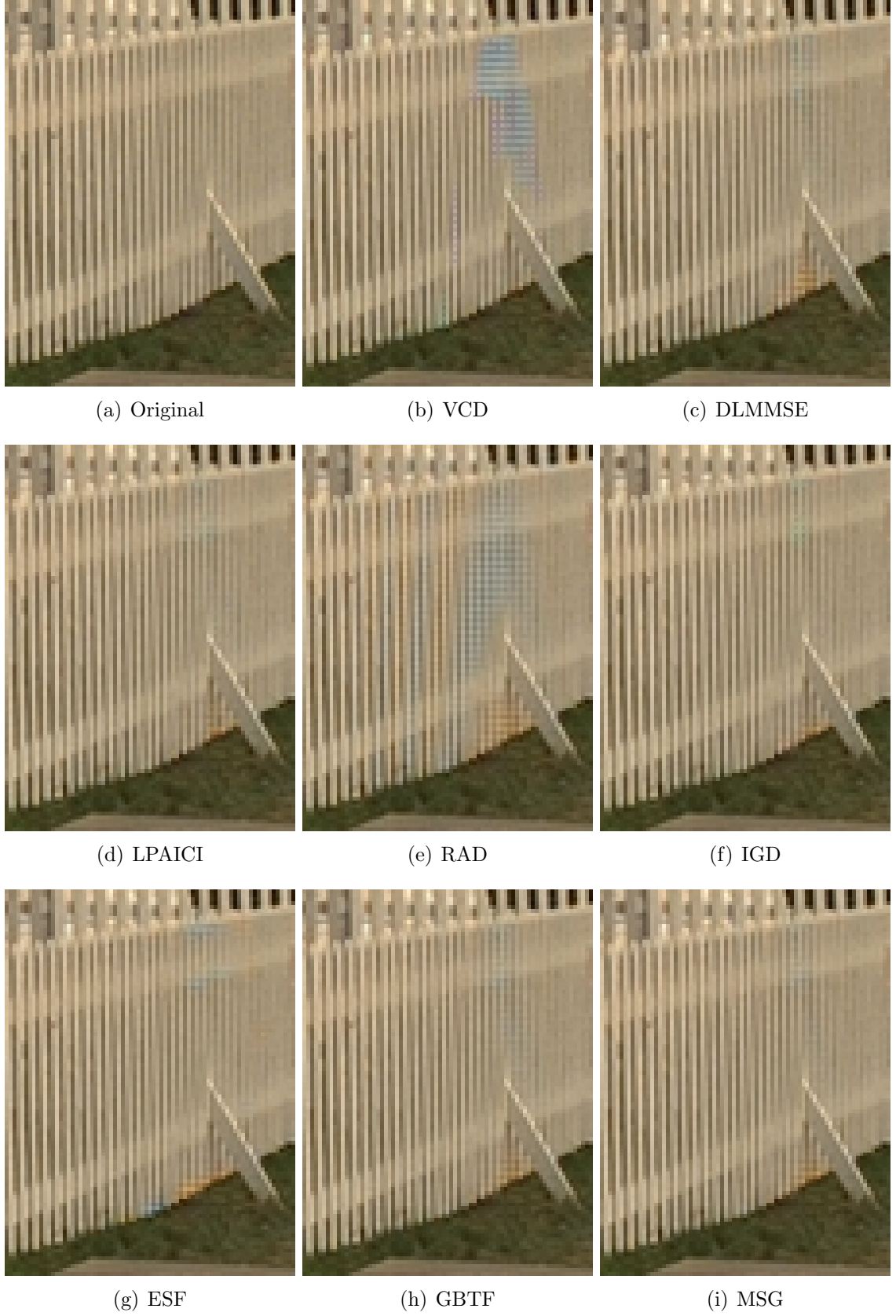
#### 4.4 *Experimental Results on the Bayer Pattern*

The results of the three proposed methods are compared with eleven state of the art algorithms included in a recent survey paper [28] and two methods introduced after this survey paper. These eleven methods are: Lu and Tan’s method (LT), Alternating Projection (AP), Adaptive Homogeneity-directed (AHD), Successive Approximation with edge-weighted improvement (SA), Lukac’s method with post-processing (CCA), Frequency-domain Demosaicing (FD), Directional Filtering and a posteriori Decision (DFPD), Variance of color-difference (VCD), Directional Linear Minimum Mean Square-Error Estimation (DLMMSE), Local Polynomial Approximation (LPA), and Adaptive Filtering for demosaicing (AF). The two more recent methods are Regularization Approaches to Demosaicking (RAD) [34], and Integrated Gradients (IGD) [10] methods. Twelve images from the Kodak PhotoCD dataset are used as the test images. The PSNR comparison results are summarized in Table 3. Among the sixteen methods in total, the proposed multiscale gradients based algorithm (labeled *MSG* in Table 3) has the best overall PSNR average with 42.97 dB. Our gradient based (GBTF) method comes third closely after the IGD method with 42.53 dB.

LPA method follows it with 42.39 dB, and the proposed ESF based algorithm is behind LPA with 42.35 dB. The fence region from the lighthouse image is shown in Figure 14 for subjective quality comparison. The complete Kodak dataset is shown in Figure 13.

**Table 3:** Comparison of PSNR values for different demosaicing methods.

No.		LT	AP	AHD	SA	CCA	FD	DFPD	VCD	DL	LPA	AF	RAD	IGD	ESF	GBTf	MSG
1	R	42.90	42.07	41.42	41.90	41.14	38.86	42.48	42.97	42.90	<b>43.86</b>	42.93	42.02	43.20	42.51	43.35	43.81
	G	46.24	45.33	45.16	46.32	45.56	44.16	46.15	46.74	47.56	47.75	46.98	46.32	47.18	46.88	47.66	<b>47.95</b>
	B	43.06	42.69	42.23	42.96	42.13	41.41	43.13	43.50	43.86	<b>44.46</b>	43.65	42.97	43.68	43.41	44.10	44.44
2	R	38.00	39.06	38.58	40.21	39.58	37.40	40.27	40.93	41.39	42.10	38.74	40.89	42.17	42.15	42.11	<b>42.51</b>
	G	39.82	42.75	40.08	43.48	43.03	42.30	42.04	43.83	43.69	44.81	41.67	43.80	45.12	45.37	45.02	<b>45.55</b>
	B	37.59	38.97	38.05	39.54	38.96	38.23	39.68	40.37	40.44	41.07	38.18	39.74	41.27	41.09	41.10	<b>41.41</b>
3	R	43.33	42.53	41.13	42.53	41.74	39.35	42.38	42.92	42.95	43.77	43.48	42.95	43.41	43.15	43.57	<b>44.12</b>
	G	45.88	44.91	43.67	44.52	45.21	43.49	44.97	45.58	46.26	46.53	46.66	46.19	46.39	46.44	46.77	<b>47.41</b>
	B	42.44	41.51	40.33	40.48	40.87	41.03	41.47	41.85	41.80	42.65	42.41	41.93	42.14	41.97	42.26	<b>42.96</b>
4	R	34.88	35.20	34.17	35.96	35.65	32.28	35.56	36.57	36.33	37.42	35.51	36.25	37.51	37.01	37.26	<b>37.84</b>
	G	37.15	39.66	36.14	40.37	39.77	37.58	38.08	40.25	39.63	41.15	38.88	40.04	41.60	41.27	41.12	<b>41.86</b>
	B	34.98	35.58	34.35	36.78	36.22	32.95	35.85	37.10	36.66	37.85	35.63	36.46	<b>38.27</b>	37.65	37.59	38.15
5	R	42.85	42.50	41.76	42.93	42.76	39.93	42.86	43.70	43.71	44.26	43.16	42.78	44.35	43.85	44.29	<b>44.64</b>
	G	44.90	45.30	44.06	46.12	45.98	43.60	45.44	46.71	46.68	47.01	46.29	45.84	47.35	46.98	47.33	<b>47.74</b>
	B	41.90	42.31	41.08	42.33	41.94	41.11	42.49	43.10	42.93	43.42	42.69	42.15	43.26	43.06	43.37	<b>43.79</b>
6	R	38.81	39.08	37.64	39.18	39.03	36.88	39.29	39.73	39.98	40.44	39.38	39.71	40.45	40.41	40.66	<b>41.16</b>
	G	40.88	42.78	39.89	43.29	43.32	42.74	41.75	43.33	43.19	43.85	42.63	43.27	44.46	44.56	44.42	<b>45.03</b>
	B	39.35	40.02	38.41	40.53	40.46	39.23	39.94	40.82	40.96	41.39	39.80	40.22	41.71	41.49	41.65	<b>42.04</b>
7	R	41.01	42.18	42.37	43.40	42.45	40.10	43.77	44.47	44.75	44.91	41.61	44.06	45.51	45.56	45.33	<b>45.74</b>
	G	42.88	45.75	43.77	46.42	45.72	44.84	45.41	47.03	46.82	47.15	44.53	46.86	47.96	48.27	47.87	<b>48.35</b>
	B	40.55	41.70	41.57	42.24	41.62	40.62	42.94	43.55	43.54	43.77	40.99	42.68	44.13	44.03	43.98	<b>44.35</b>
8	R	40.11	40.02	39.16	40.98	40.66	37.12	40.56	41.10	41.69	42.18	40.58	40.76	42.35	41.99	42.35	<b>42.76</b>
	G	41.95	43.86	40.77	44.33	43.96	42.05	42.55	44.09	44.14	44.72	43.49	43.88	45.32	45.11	45.25	<b>45.76</b>
	B	39.46	39.95	38.62	40.33	39.95	37.90	40.15	40.60	40.88	41.46	39.92	39.79	41.72	41.18	41.57	<b>41.95</b>
9	R	42.14	41.79	40.22	42.15	42.48	39.78	41.40	42.31	42.77	42.95	42.37	42.34	43.09	42.83	43.27	<b>43.60</b>
	G	43.47	44.59	41.66	45.07	45.49	43.59	43.26	44.89	44.88	45.14	44.94	45.07	45.78	45.63	45.83	<b>46.29</b>
	B	40.47	40.66	39.01	40.33	40.90	40.41	40.24	40.98	40.95	41.36	40.79	40.50	41.50	41.07	41.48	<b>41.84</b>
10	R	38.87	39.51	37.62	40.32	40.71	37.80	38.83	40.21	40.40	40.91	39.61	40.35	41.44	41.28	40.94	<b>41.59</b>
	G	40.19	42.79	38.88	43.45	43.75	42.17	40.61	42.89	42.43	43.13	42.16	43.08	44.04	44.19	43.57	<b>44.31</b>
	B	38.21	39.07	36.93	39.41	39.60	38.44	38.28	39.41	39.34	39.77	38.77	39.20	40.20	40.07	39.83	<b>40.33</b>
11	R	39.20	38.61	37.01	38.54	38.18	37.10	38.33	38.93	39.20	39.37	39.29	39.16	39.46	39.31	39.60	<b>39.85</b>
	G	41.40	41.23	39.46	41.56	41.67	40.53	40.91	41.97	42.26	42.32	42.39	42.20	42.50	42.57	42.90	<b>43.14</b>
	B	38.65	38.19	36.82	38.13	38.12	37.61	38.22	38.77	38.84	39.10	38.91	38.60	39.04	38.96	39.25	<b>39.50</b>
12	R	36.05	36.78	34.58	37.18	36.20	36.89	36.32	37.29	37.99	37.78	36.80	37.91	37.61	37.43	38.09	<b>38.15</b>
	G	37.22	39.07	36.49	39.88	39.68	39.83	38.25	39.86	39.87	40.05	39.04	40.23	40.46	40.00	40.30	<b>40.66</b>
	B	34.59	35.07	33.90	35.68	35.55	35.65	35.11	35.75	36.22	36.17	35.15	35.99	36.22	35.77	36.18	<b>36.30</b>
Avg		40.32	40.92	39.36	41.36	41.11	39.58	40.81	41.78	41.89	42.39	41.11	41.56	42.55	42.35	42.53	<b>42.97</b>



**Figure 14:** Comparison on the Bayer pattern. Fence region from the lighthouse image.

## CHAPTER V

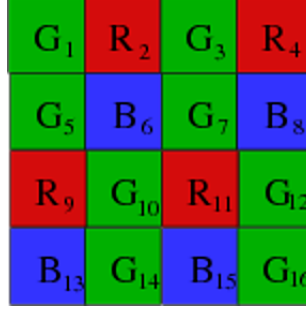
### DEMOSAICING ON THE LUKAC PATTERN

#### **5.1 *Motivation***

Although designed for the Bayer mosaic pattern, the proposed methods can be modified to be applied to other mosaic patterns. However, such an application may not be feasible for all mosaic patterns because of the restrictions dictated by the directional nature of our approach. When the modification is feasible, an important question would be whether the changes needed to comply with the new pattern layout lead to a significant performance loss or not. To find out if we can outperform other available solutions on a different pattern layout, we modified our Edge Strength Based algorithm for the Lukac pattern. Encouraged by the experimental results, we also applied the multiscale gradients idea behind our latest demosaicing method to the Lukac pattern.

#### **5.2 *Application of the ESF Based Method to the Lukac Pattern***

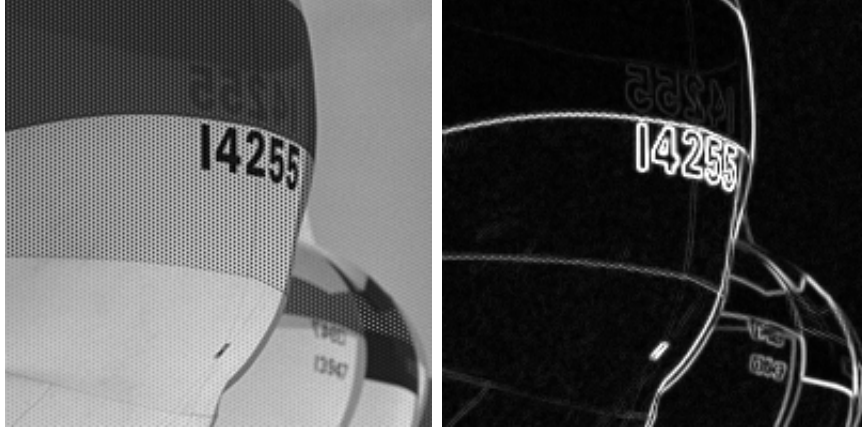
Lukac mosaic pattern is similar to Bayer pattern in the sense that it consists of pure RGB components. When we shift every other row in a Bayer pattern by one pixel to either side, we obtain the Lukac pattern. Hence, the horizontal relationship between the pixels is still the same, but the vertical arrangement is significantly altered. An inspection on the Lukac mosaic pattern reveals that it is possible to take gradients in three directions as opposed to four on the Bayer pattern. While we still have the horizontal component, the vertical one is gone and the diagonal components lean more towards the vertical direction. Based on this observation, we modify the edge strength filter as follows:



**Figure 15:** Numbered Lukac pattern layout.

$$E_{R_{11}} = \frac{|B_6 - B_{15}| + |G_7 - G_{16}|}{2} + \frac{|B_8 - B_{15}| + |G_7 - G_{14}|}{2} + |G_{10} - G_{12}|, \quad (50)$$

where  $E_{R_i}$  stands for the filter output at the red pixel coordinate  $R_i$ . The filter equation is similar for green and blue pixel locations. By running the filter through the whole mosaicked input image, we obtain an edge strength map. This map will be used to interpolate and update the green channel adaptively. Figure 16 shows a mosaicked input sample region and its generated edge strength map.



**Figure 16:** Mosaicked region and its edge strength filter output.

### 5.2.1 Green Channel Interpolation

Similar to the Bayer pattern, Lukac pattern has twice as many green samples as red and blue ones. Hence, the green channel is less prone to aliasing and the natural choice for initial interpolation. The first step of the green channel interpolation is to

find directional color difference estimations. Horizontal estimation is straightforward because all the rows are comprised of either green-red or green-blue interleaved pixels like the Bayer pattern:

$$\begin{aligned}\tilde{B}^H(i, j) &= \frac{B(i, j-1) + B(i, j+1)}{2} + \frac{2 * G(i, j) - G(i, j-2) - G(i, j+2)}{4} \\ \tilde{G}^H(i-1, j-1) &= \frac{G(i-1, j-2) + G(i-1, j)}{2} + \frac{2 * R(i-1, j-1) - R(i-1, j-3) - R(i-1, j+1)}{4}.\end{aligned}\quad (51)$$

However, it is not possible to carry out the same operation in the vertical direction because the channel values are not in place. That is why, instead of a 5 by 1 column, we consider a window of 5 by 3 around the target pixel. If a channel value is missing, then it is estimated by taking the average of closest horizontal samples as follows:

$$\begin{aligned}\tilde{R}^V(i, j) &= \frac{\frac{R(i-1, j-1) + R(i-1, j+1)}{2} + R(i+1, j)}{2} + \frac{2 * G(i, j) - \frac{G(i-2, j-1) + G(i-2, j+1)}{2} - \frac{G(i+2, j-1) + G(i+2, j+1)}{2}}{4} \\ \tilde{G}^V(i-1, j-1) &= \frac{\frac{G(i-2, j-1) + \frac{G(i, j-2) + G(i, j)}{2}}{2} + \frac{2 * R(i-1, j-1) - \frac{R(i-3, j-2) + R(i-3, j)}{2} - \frac{R(i+1, j-2) + R(i+1, j)}{2}}{4}.\end{aligned}\quad (52)$$

Next, we calculate the horizontal and vertical color difference estimates for each pixel location. For green pixel locations, we observe that the vertical and horizontal estimates do not belong to the same channel. To get around this problem, we make another approximation and use the closest vertical color difference estimation.

$$\begin{aligned}\tilde{\Delta}_{g,b}^H(i, j) &= G(i, j) - \tilde{B}^H(i, j) \\ \tilde{\Delta}_{g,b}^V(i, j) &= G(i-1, j) - \tilde{B}^V(i-1, j)\end{aligned}$$



$$\begin{aligned}
\tilde{\Delta}_{g,r}^H(i-1, j-1) &= \tilde{G}^H(i-1, j-1) - R(i-1, j-1) \\
\tilde{\Delta}_{g,r}^V(i-1, j-1) &= \tilde{G}^V(i-1, j-1) - R(i-1, j-1).
\end{aligned} \tag{53}$$

Once we have the complete horizontal and vertical color difference information, we combine them adaptively using the edge strength map generated above. Since the constant color difference assumption is likely to break across edges, we calculate edge strength differences in both directions over a local window and give more weight to the direction with less difference. Assuming a window size of 5 by 5, the weights for horizontal and vertical directions are:

$$\begin{aligned}
H(i, j) &= 1 / \left( \sum_{m=-2}^2 \sum_{n=-2}^1 (E(i+m, j+n) - E(i+m, j+n+1)) \right) \\
V(i, j) &= 1 / \left( \sum_{m=-2}^1 \sum_{n=-2}^2 (E(i+m, j+n) - E(i+m+1, j+n)) \right).
\end{aligned} \tag{54}$$

And the green channel interpolation is carried out as follows:

$$\begin{aligned}
\hat{\Delta}_{g,b}(i, j-1) &= [V(i, j-1) \cdot f_v \cdot \tilde{\Delta}_{g,b}^V(i-2 : i+2, j-1) + \\
&\quad H(i, j-1) \cdot \tilde{\Delta}_{g,b}^H(i, j-2 : j) \cdot f'_h] / T_{i,j-1} \\
\hat{\Delta}_{g,r}(i-1, j-1) &= [V(i-1, j-1) \cdot f_v \cdot \tilde{\Delta}_{g,r}^V(i-3 : i+1, j-1) \\
&\quad + H(i-1, j-1) \cdot \tilde{\Delta}_{g,r}^H(i-1, j-2 : j) \cdot f'_h] / T(i-1, j-1) \\
T(i, j) &= H(i, j) + V(i, j) \\
f_h &= [1 \ 2 \ 1] / 4 \\
f_v &= [1 \ 0 \ 2 \ 0 \ 1] / 4.
\end{aligned} \tag{55}$$

Again, we needed to modify the vertical component because the required color difference estimate is not available for the immediate vertical neighbor. That is why we skip one pixel and bring the estimate from the next closest vertical resource. After the initial interpolation, we update the green channel interpolation reusing the weights derived above. The green channel update treats north-south and east-west

directions separately. However, since there are no perfectly vertical red or blue pixels (to any red or blue pixel) available in the Lukac pattern, we take the simple average of the closest samples again:

$$\begin{aligned}
\tilde{\Delta}_{g,b}(i, j-1) &= \hat{\Delta}_{g,b}(i, j-1) \cdot (1-w) + \\
&[V(i-2, j-1) \cdot (\hat{\Delta}_{g,b}(i-2, j-2) + \hat{\Delta}_{g,b}(i-2, j)) / 2 + \\
&V(i+2, j-1) \cdot (\hat{\Delta}_{g,b}(i+2, j-2) + \hat{\Delta}_{g,b}(i+2, j)) / 2 + \\
&H(i, j-3) \cdot \hat{\Delta}_{g,b}(i, j-3) + \\
&H(i, j+1) \cdot \hat{\Delta}_{g,b}(i, j+1)] \cdot w / T(i, j-1) \\
\tilde{\Delta}_{g,r}(i-1, j-1) &= \hat{\Delta}_{g,r}(i-1, j-1) \cdot (1-w) + \\
&[V(i-3, j-1) \cdot (\hat{\Delta}_{g,r}(i-3, j-2) + \hat{\Delta}_{g,r}(i-3, j)) / 2 + \\
&V(i+1, j-1) \cdot (\hat{\Delta}_{g,r}(i+1, j-2) + \hat{\Delta}_{g,r}(i+1, j)) / 2 + \\
&H(i-1, j-3) \cdot \hat{\Delta}_{g,r}(i-1, j-3) + \\
&H(i-1, j+1) \cdot \hat{\Delta}_{g,r}(i-1, j+1)] \cdot w / T(i-1, j-1) \\
T(i, j) &= V(i-2, j) + V(i+2, j) + H(i, j-2) + H(i, j+2). \tag{56}
\end{aligned}$$

Then, we add the finalized color difference estimate to the target pixel value to get the green channel estimate:

$$\begin{aligned}
\tilde{G}(i, j-1) &= B(i, j-1) + \tilde{\Delta}_{g,b}(i, j-1) \\
\tilde{G}(i-1, j-1) &= R(i-1, j-1) + \tilde{\Delta}_{g,r}(i-1, j-1). \tag{57}
\end{aligned}$$

### 5.2.2 Red and Blue Channel Interpolation

After the green channel interpolation is complete, we fill in red and blue channels using the closest color difference estimates available. For the red channel interpolation, we consider the closest four neighbors for pixels on the red-green rows, and the closest

three neighbors for pixels on the green-blue rows.

$$\begin{aligned}
\tilde{R}(i, j) &= G(i, j) - \frac{\tilde{G}(i+1, j) - R(i+1, j)}{2} \\
&\quad - \frac{\tilde{G}(i-1, j-1) - R(i-1, j-1) + \tilde{G}(i-1, j+1) - R(i-1, j+1)}{4} \\
\tilde{R}(i, j-1) &= G(i, j-1) - \frac{\tilde{G}(i-1, j-1) - R(i-1, j-1)}{2} \\
&\quad - \frac{\tilde{G}(i+1, j-2) - R(i+1, j-2) + \tilde{G}(i+1, j) - R(i+1, j)}{4} \\
\tilde{R}(i-1, j) &= G(i-1, j) - \\
&\quad \frac{\tilde{G}(i-1, j-1) - R(i-1, j-1) + \tilde{G}(i-1, j+1) - R(i-1, j+1)}{2.5} \\
&\quad - \frac{\tilde{G}(i-3, j) - R(i-3, j) + \tilde{G}(i+1, j) - R(i+1, j)}{10}.
\end{aligned} \tag{58}$$

The interpolation formulas are similar for the blue channel. By the end of this step, we completed all the missing samples in the input image.

### 5.3 *Application of the MSG Based Method to the Lukac Pattern*

We tested our ESF based algorithm on the Kodak image set and compared its results to other solutions available for the Lukac pattern. It outperformed other algorithms on every image in the test set. Based on this result, we wanted to see if the performance of our multiscale gradients solution can carry onto the Lukac pattern as well. The rest of this section describes the changes that were needed to apply the MSG algorithm to the Lukac pattern.

#### 5.3.1 Green Channel Interpolation

As a result of the Lukac pattern layout, it is not possible to take immediate vertical gradients. However, we observe that we can take vertical gradients when we double the scale. So we modified our vertical multiscale gradients equation accordingly:

$$D^v(i, j) = \left| \frac{G(i+2, j) - G(i-2, j)}{M_0} - \frac{R(i+4, j) - R(i-4, j)}{M_1} + \frac{G(i+6, j) - G(i-6, j)}{M_2} - \frac{R(i+8, j) - R(i-8, j)}{M_3} + \dots \right|, \quad (59)$$

where the  $M_i$  terms are the normalizers.

The layout of the Lukac pattern also necessitates a change in vertical color difference estimation. Since all the required channel values are not available in the same column, we estimate the missing values by taking simple average using samples from adjacent columns:

$$\begin{aligned} \tilde{R}^V(i, j) &= \frac{\frac{R(i-1, j-1) + R(i-1, j+1)}{2} + R(i+1, j)}{2} + \\ &\frac{2 \cdot G(i, j) - \frac{G(i-2, j-1) + G(i-2, j+1)}{2} - \frac{G(i+2, j-1) + G(i+2, j+1)}{2}}{4} \\ \tilde{G}^V(i-1, j-1) &= \frac{G(i-2, j-1) + \frac{G(i, j-2) + G(i, j)}{2}}{2} + \\ &\frac{2 \cdot R(i-1, j-1) - \frac{R(i-3, j-2) + R(i-3, j)}{2} - \frac{R(i+1, j-2) + R(i+1, j)}{2}}{4}. \end{aligned} \quad (60)$$

Another problem we faced with the Lukac pattern was the mismatch between vertical and horizontal color difference estimates at green channel coordinates. Namely, the calculated vertical and horizontal color differences at these locations belong to different color pairs. That is why we bring the needed vertical color difference estimate from the closest available resource:

$$\begin{aligned} \tilde{\Delta}_{g,b}^V(i, j) &= G(i-1, j) - \tilde{B}^V(i-1, j) \\ \tilde{\Delta}_{g,r}^V(i-1, j) &= G(i, j) - \tilde{R}^V(i, j). \end{aligned}$$

Also, the combined color difference estimate equations are modified to bring the neighboring vertical estimates from two pixels away instead of one:

$$\begin{aligned}
\hat{\Delta}_{g,r}(i-1, j-1) &= [w_V \cdot \mathbf{f}_v \cdot \tilde{\Delta}_{g,r}^V(i-3 : i+1, j) + \\
&\quad w_H \cdot \tilde{\Delta}_{g,r}^H(i-1, j-2 : j) \cdot \mathbf{f}_h'] / w_T \\
w_T &= w_V + w_H \\
\mathbf{f}_h &= [1/4 \ 2/4 \ 1/4] \\
\mathbf{f}_v &= [1/4 \ 0 \ 2/4 \ 0 \ 1/4].
\end{aligned} \tag{61}$$

Similarly, we modify the green channel update equation as follows:

$$\begin{aligned}
\tilde{\Delta}_{g,r}(i, j) &= \hat{\Delta}_{g,r}(i, j) \cdot (1 - w) + \\
&\quad [w_N \cdot (\hat{\Delta}_{g,r}(i-2, j-1) + \hat{\Delta}_{g,r}(i-2, j+1) + \hat{\Delta}_{g,r}(i-4, j)) / 3 + \\
&\quad w_S \cdot (\hat{\Delta}_{g,r}(i+2, j-1) + \hat{\Delta}_{g,r}(i+2, j+1) + \hat{\Delta}_{g,r}(i+4, j)) / 3 + \\
&\quad w_E \cdot \hat{\Delta}_{g,r}(i, j-2) + \\
&\quad w_W \cdot \hat{\Delta}_{g,r}(i, j+2)] \cdot w / w_T \\
w_T &= w_N + w_S + w_E + w_W.
\end{aligned} \tag{62}$$

### 5.3.2 Red and Blue Channel Interpolation

The Lukac pattern layout necessitates modifications in red and blue channel interpolation as well. We still estimate the missing red and blue samples using the closest color difference estimates, but their orientation is different from the Bayer pattern. For the red channel interpolation, the pixels on green&blue rows use estimates from three neighbors and the ones on green&red rows use four:

$$\begin{aligned}
\tilde{R}(i, j) &= G(i, j) - \frac{\tilde{G}(i+1, j) - R(i+1, j)}{2} \\
&\quad - \frac{\tilde{G}(i-1, j-1) - R(i-1, j-1) + \tilde{G}(i-1, j+1) - R(i-1, j+1)}{4}
\end{aligned}$$

$$\begin{aligned}
\tilde{R}(i, j-1) = & G(i, j-1) - \frac{\tilde{G}(i-1, j-1) - R(i-1, j-1)}{2} \\
& - \frac{\tilde{G}(i+1, j-2) - R(i+1, j-2) + \tilde{G}(i+1, j) - R(i+1, j)}{4} \\
\tilde{R}(i-1, j) = & G(i-1, j) - \\
& \frac{\tilde{G}(i-1, j-1) - R(i-1, j-1) + \tilde{G}(i-1, j+1) - R(i-1, j+1)}{2.5} \\
& - \frac{\tilde{G}(i-3, j) - R(i-3, j) + \tilde{G}(i+1, j) - R(i+1, j)}{10}.
\end{aligned} \tag{63}$$

Blue channel interpolation is similar to the red channel interpolation described above. Although we needed to make several changes to apply the algorithm to the Lukac pattern, the main structure of the MSG method is maintained.

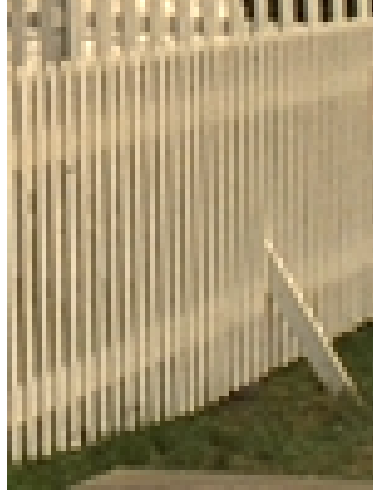
#### 5.4 *Experimental Results on the Lukac Pattern*

We tested the proposed algorithms on the 20 image Kodak test set. The results are compared to methods featured in a recent paper on regularization approaches to demosaicing [34]. These methods are the adaptive (AA) and quadratic (QA) approaches [34], the recursive filtering (RF) method proposed in [24], and the universal solution (US) proposed in [32]. The comparison results are summarized in Table 4. The proposed MSG based algorithm outperforms other methods for every image in the test set in terms of CPSNR. Our ESF based solution comes second with 0.25 dB behind MSG. The next highest performing method (adaptive approach from [34]) trails our MSG and ESF methods by 1.17 dB and 0.92 dB, respectively. A sample region from the lighthouse image (image no. 16) is shown in Figure 17 for visual quality comparison.

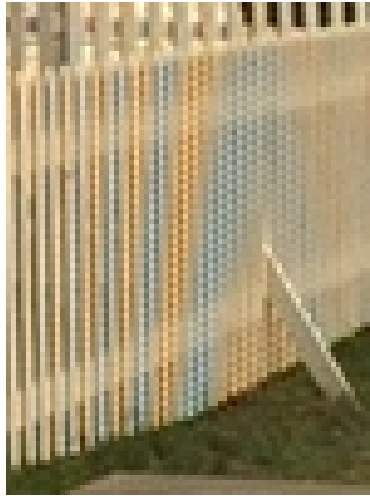
These results clearly show our algorithms can successfully be applied to a non-Bayer pattern to the extent that they outperform other available solutions by a clear margin in terms of objective CPSNR measure while providing better visual results.

**Table 4:** Comparison of CPSNR values for different demosaicing methods on the Lukac pattern.

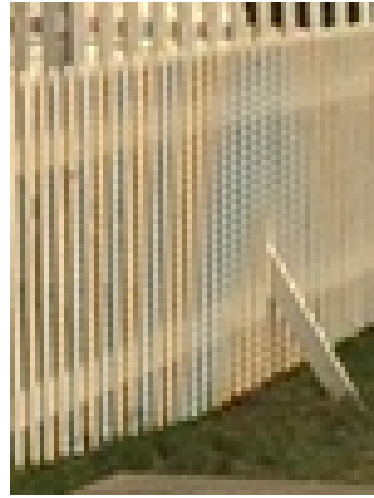
No.	US	RF	QA	AA	ESF	MSG
1	30.78	34.05	36.58	37.80	39.52	<b>39.90</b>
2	36.57	37.82	36.84	37.90	40.15	<b>40.51</b>
3	30.93	32.25	35.59	36.73	36.96	<b>37.21</b>
4	32.02	35.22	37.13	39.08	40.31	<b>40.95</b>
5	37.09	38.19	39.74	41.35	41.63	<b>41.87</b>
6	28.67	32.27	34.95	36.14	37.11	<b>37.40</b>
7	36.64	39.04	40.20	41.61	42.77	<b>42.90</b>
8	36.80	38.98	40.61	41.83	42.48	<b>42.49</b>
9	33.78	36.06	38.25	39.32	40.42	<b>40.74</b>
10	37.11	39.58	40.20	42.34	43.23	<b>43.50</b>
11	28.55	30.55	34.48	34.34	35.55	<b>35.92</b>
12	35.81	37.10	37.69	38.57	38.89	<b>38.90</b>
13	35.46	38.90	40.32	42.70	43.91	<b>44.49</b>
14	36.37	37.60	40.58	40.97	41.71	<b>41.73</b>
15	31.73	33.40	35.99	36.22	36.28	<b>36.87</b>
16	32.51	35.53	37.72	39.27	40.39	<b>40.72</b>
17	32.93	37.07	38.19	39.35	40.80	<b>40.91</b>
18	32.83	35.40	37.75	38.76	40.02	<b>40.02</b>
19	34.35	36.01	37.00	37.65	38.16	<b>38.26</b>
20	30.32	32.56	34.78	34.97	35.12	<b>35.17</b>
Avg	33.56	35.88	37.73	38.85	39.77	<b>40.02</b>



(a) Original



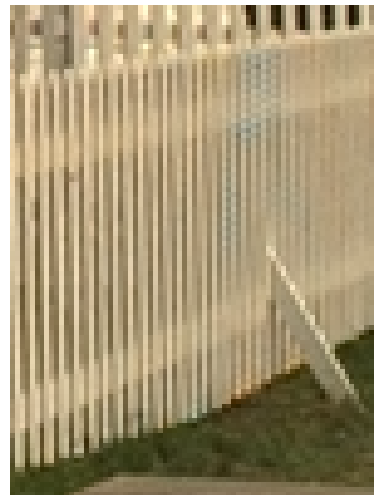
(b) QA



(c) AA



(d) ESF



(e) MSG

**Figure 17:** Comparison on the Lukac pattern. Fence region from the lighthouse image.



## CHAPTER VI

### LOW SPECTRAL CORRELATION DEMOSAICING

#### *6.1 Algorithm Background*

Most demosaicing methods assume that there is a strong correlation between the color channels. This assumption holds true most of the time and leads to both objective and subjective quality improvement. However, if the spectral correlation is weaker than anticipated, allowing different color channels to affect each other’s interpolation results too much will lead to false color and zipper artifacts. A very recent paper [43] asserts that the commonly used Kodak test set does not accurately represent modern digital images and proposes a new test set with vivid colors and sharp color changes. Consequently, this new test set has higher saturation and lower spectral correlation than the Kodak test set on average. The images included in this dataset are shown in Figure 18. The paper proposes fusing directional estimates using gradients on a very small local window and then enhancing the results with non-local averaging. This interpolation method outperforms all the classic demosaicing approaches on the new test set because it does not heavily rely on spectral correlation. On the other hand, we suspect that it would perform poorly on the Kodak test set for the very same reason.

Although the method proposed in [43] offers superior objective and subjective quality for images with low spectral correlation, its performance comes with a high computational cost because of the non-local nature of the algorithm. We wanted to see if we can achieve a similar performance using only local information. Our approach was to apply the multiscale gradients idea that we described earlier to the low spectral correlation case with some modifications combined with some other ideas.



where  $N_1$  is a normalizing constant, and  $D^h(i, j)$  and  $D^v(i, j)$  stand for the horizontal and vertical values at pixel location  $(i, j)$ , respectively.

We estimate the target green pixel in a single step by adaptively combining the closest four green neighbors and using the available target channel (either red or blue) as a feedback.

$$\begin{aligned}\tilde{G}(i, j) = & R(i, j) \cdot w - [w_N \cdot R(i - 2, j) + w_S \cdot R(i + 2, j) + \\ & w_E \cdot R(i, j - 2) + w_W \cdot R(i, j + 2)] * w / w_T + \\ & [w_N \cdot G(i - 1, j) + w_S \cdot G(i + 1, j) + \\ & w_E \cdot G(i, j - 1) + w_W \cdot G(i, j + 1)] / w_T\end{aligned}$$

$$w_T = w_N + w_S + w_E + w_W, \quad (65)$$

where the weights  $(w_N, w_S, w_E, w_W)$  are calculated by summing multiscale color gradients over a local window in that direction. The contribution of each pixel in this local window can be identical for simplicity or the center pixels can be given more weight to preserve the locality. For a window size of 3 by 3, the directional weights can be given as:

$$\begin{aligned}w_N &= 1 / \left( \sum_{k=i-2}^i \sum_{l=j-1}^{j+1} (D_{k,l}^v)^2 \right) \\ w_S &= 1 / \left( \sum_{k=i}^{i+2} \sum_{l=j-1}^{j+1} (D_{k,l}^v)^2 \right) \\ w_W &= 1 / \left( \sum_{k=i-1}^{i+1} \sum_{l=j-2}^j (D_{k,l}^h)^2 \right) \\ w_E &= 1 / \left( \sum_{k=i-1}^{i+1} \sum_{l=j}^{j+2} (D_{k,l}^h)^2 \right).\end{aligned} \quad (66)$$

The weight  $w$  that determines the amount of feedback given by the target pixel channel is a function of local green pixel average. If the average is closer to either 0 or 255 (in an 8 bit per channel color depth setup), then it is more likely for the constant color difference rule to break so we want less feedback from the target pixel channel.

The relationship between  $w$  and the local green channel average can be formulated as follows:

$$G_{avg} = \frac{G(i-1, j) + G(i+1, j) + G(i, j-1) + G(i, j+1)}{4}$$

$$T = \min(G_{avg}, 255 - G_{avg})$$

$$w = \begin{cases} w_1 & \text{if } T < t_1, \\ w_2 & \text{if } T > t_2, \\ w_1 + \frac{(T - t_1) \cdot (w_2 - w_1)}{(t_2 - t_1)} & \text{else.} \end{cases} \quad (67)$$

### 6.3 Red and Blue Channel Interpolation

Once we complete the green channel adaptively in a single pass, we move onto the red and blue channels. For red channel interpolation at blue pixels, we consider the four closest neighbors with available red channel and estimated green channel values. These pixels happen to be the diagonal neighbors of the target pixel because of the Bayer channel layout. We estimate the red target pixel in a similar way to the green channel estimation except the directions are shifted 45 degrees now. To deal with this change while retaining the multiscale gradient approach, we calculate the diagonal multiscale gradients:

$$D^{nw}(i, j) = \left| \frac{G(i+1, j+1) - G(i-1, j-1)}{2} - \frac{R(i+2, j+2) - R(i-2, j-2)}{N_1} \right|$$

$$D^{ne}(i, j) = \left| \frac{G(i+1, j-1) - G(i-1, j+1)}{2} - \frac{R(i+2, j-2) - R(i-2, j+2)}{N_1} \right|, \quad (68)$$

where  $D^{nw}(i, j)$  and  $D^{ne}(i, j)$  are the diagonal multiscale gradient outputs at the north-west and north-east directions, respectively. The multiscale gradients are then used to calculate the directional weights:

$$w_{NW} = 1 / \left( \sum_{k=i-2}^i \sum_{l=j-2}^j (D_{k,l}^{nw})^2 \right)$$

$$\begin{aligned}
w_{NE} &= 1 / \left( \sum_{k=i-2}^i \sum_{l=j}^{j+1} (D_{k,l}^{ne})^2 \right) \\
w_{SW} &= 1 / \left( \sum_{k=i}^{i+2} \sum_{l=j-2}^j (D_{k,l}^{ne})^2 \right) \\
w_{SE} &= 1 / \left( \sum_{k=i}^{i+2} \sum_{l=j}^{j+2} (D_{k,l}^{nw})^2 \right), \tag{69}
\end{aligned}$$

and the red channel estimation is given by:

$$\begin{aligned}
\tilde{R}(i, j) &= \tilde{G}(i, j) \cdot w - [w_{NW} \cdot \tilde{G}(i-1, j-1) + w_{NE} \cdot \tilde{G}(i-1, j+1) + \\
&\quad w_{SW} \cdot \tilde{G}(i+1, j-1) + w_{SE} \cdot \tilde{G}(i+1, j+1)] * w / w_T + \\
&\quad [w_{NW} \cdot R(i-1, j-1) + w_{NE} \cdot R(i-1, j+1) + \\
&\quad w_{SW} \cdot R(i+1, j-1) + w_{SE} \cdot R(i+1, j+1)] / w_T \\
w_T &= w_{NW} + w_{NE} + w_{SW} + w_{SE}. \tag{70}
\end{aligned}$$

The interpolation of blue channel values at red pixel locations is similar. The weight 'w' can be made adaptive the same way it was done for the green channel.

For red and blue channel interpolation at green pixel coordinates we follow the same logic. However, since there are only two immediate neighbors with the original desired channel value available for these locations, we modify the equations accordingly. The available neighbors share either the same row or the same column with the target pixel. For the case where they are in the same column, the red channel interpolation is given by:

$$\tilde{R}(i, j) = G(i, j) \cdot w - [\tilde{G}(i, j-1) + \tilde{G}(i, j+1)] * w / 2 + [R(i, j-1) + R(i, j+1)] / 2. \tag{71}$$

And for the same row case:

$$\tilde{R}(i, j) = G(i, j) \cdot w - [\tilde{G}(i-1, j) + \tilde{G}(i+1, j)] * w / 2 + [R(i-1, j) + R(i+1, j)] / 2. \tag{72}$$

Again, the interpolation for the blue channel is similar. By the end of this step, we interpolated all the missing pixel values.

## 6.4 Experimental Results

The proposed algorithm is tested on the new dataset which was recently released online [44]. This dataset features very challenging images with high saturation and low spectral correlation. We compared the results of our algorithm to the non-local method in [43] and other methods included in their paper. These methods are the Local Directional Interpolation Non-local Means (LDI-NLM) and the Local Directional Interpolation Non-local Adaptive Thresholding (LDI-NAT) methods which are the two slightly different versions of their algorithm. The other methods included in the comparison are the Self-similarity Driven (SSD) method [5], the Directional Linear Minimum Mean Square-error Estimation (DLMMSE) method [42], the Successive Approximation (SA) method [27], the Adaptive Homogeneity-directed (AHD) method [17], and the Second Order Laplacian Correction (SOLC) method [1].

**Table 5:** Comparison of PSNR values for different demosaicing methods on the new McMaster dataset.

img no		SOLC	AHD	SA	DLMMSE	SSD	LDI-NLM	LDI-NAT	Proposed
1	R	28.26	26.02	23.53	26.94	27.28	28.81	<b>29.29</b>	29.20
	G	31.22	29.82	25.17	30.63	30.68	32.31	32.67	<b>33.00</b>
	B	26.34	24.04	22.05	24.82	25.12	26.47	26.71	<b>26.98</b>
2	R	33.68	32.47	31.63	33.30	33.61	34.66	<b>35.02</b>	34.83
	G	37.62	37.20	34.00	37.66	37.81	39.01	39.08	<b>39.61</b>
	B	32.11	31.26	30.74	31.86	32.01	32.79	<b>32.92</b>	32.89
3	R	30.64	31.10	31.47	32.60	32.81	33.41	33.05	<b>33.64</b>
	G	33.73	33.49	32.75	35.28	35.05	35.50	35.51	<b>36.52</b>
	B	28.60	29.67	29.80	30.70	30.93	<b>30.99</b>	30.31	30.69
4	R	32.80	33.76	34.59	34.70	36.36	<b>37.41</b>	36.25	36.19
	G	37.16	35.66	34.05	36.99	38.98	39.01	40.33	<b>41.41</b>
	B	30.89	31.48	32.19	32.07	33.49	<b>34.02</b>	33.30	33.11
5	R	33.61	29.52	28.60	30.38	31.10	34.50	<b>35.05</b>	33.99
	G	36.28	34.73	30.97	35.11	35.43	37.67	<b>38.15</b>	38.08
	B	30.47	28.78	28.08	29.41	29.48	31.02	31.16	<b>31.35</b>
6	R	37.14	33.92	32.23	34.98	36.09	38.59	<b>39.40</b>	38.07
	G	40.30	37.72	32.50	38.61	38.85	41.70	<b>43.42</b>	42.56
	B	34.00	29.96	29.14	31.15	31.72	34.21	<b>34.97</b>	34.18
7	R	33.85	35.64	37.03	<b>38.30</b>	36.61	36.28	36.09	36.45
	G	36.34	37.36	40.39	<b>40.70</b>	37.62	37.66	37.41	38.27
	B	32.45	35.07	36.22	<b>37.29</b>	36.38	34.59	34.49	34.58
8	R	34.87	34.15	35.31	35.45	35.31	36.89	36.31	<b>37.04</b>
	G	39.09	39.45	38.49	<b>41.43</b>	40.34	40.44	40.29	41.34
	B	35.04	35.79	35.82	36.99	36.76	36.84	36.67	<b>37.18</b>
9	R	34.36	31.54	30.71	32.39	33.72	35.54	35.49	<b>35.76</b>
	G	39.62	37.99	33.83	38.73	39.52	41.56	41.73	<b>42.50</b>
	B	35.34	34.00	32.54	34.66	35.38	36.54	36.30	<b>37.00</b>

The PSNR comparison results are summarized in Table 5 and Table 6. Based

**Table 6:** Comparison of PSNR values for different demosaicing methods on the new McMaster dataset (continued).

		SOLC	AHD	SA	DLMMSE	SSD	LDI-NLM	LDI-NAT	Proposed
10	R	36.86	33.99	34.03	34.70	36.33	37.64	<b>38.26</b>	37.63
	G	40.86	39.17	36.15	40.00	40.23	42.19	42.64	<b>42.87</b>
	B	36.08	34.88	34.78	35.55	36.13	36.51	36.83	<b>36.91</b>
11	R	38.12	36.13	36.16	36.91	38.16	39.25	<b>39.82</b>	39.29
	G	40.78	39.34	37.11	40.44	40.19	41.66	<b>42.57</b>	42.45
	B	37.19	34.73	34.33	35.75	36.81	37.50	37.66	<b>38.57</b>
12	R	37.13	33.60	34.49	34.74	35.37	37.62	<b>38.36</b>	37.92
	G	40.17	40.09	37.66	39.59	39.70	41.45	41.49	<b>41.67</b>
	B	35.70	36.24	36.24	36.47	37.11	37.51	37.59	<b>37.74</b>
13	R	39.80	37.91	38.11	38.66	40.01	<b>42.23</b>	41.77	40.96
	G	43.46	42.16	39.90	42.57	43.82	<b>45.55</b>	44.89	45.14
	B	37.65	36.20	36.51	36.75	37.19	37.88	38.13	<b>38.24</b>
14	R	37.85	37.33	36.82	37.74	38.66	39.28	39.39	<b>39.44</b>
	G	41.37	40.65	38.79	41.13	41.93	42.62	42.84	<b>43.30</b>
	B	35.64	34.30	34.45	34.78	35.00	35.82	36.12	<b>36.36</b>
15	R	36.44	34.88	34.87	35.32	36.23	<b>37.34</b>	36.95	37.30
	G	41.20	40.27	38.13	40.71	40.75	42.39	42.68	<b>43.06</b>
	B	38.17	36.84	36.52	37.30	37.90	38.49	38.99	<b>39.35</b>
16	R	32.75	30.95	28.75	31.95	32.21	34.18	<b>34.97</b>	34.80
	G	34.09	32.36	28.60	33.22	32.99	35.00	<b>35.59</b>	35.33
	B	31.63	26.85	24.87	28.06	28.30	31.12	31.53	<b>34.08</b>
17	R	31.24	27.12	25.35	28.32	29.24	31.60	<b>32.14</b>	31.74
	G	35.17	32.13	26.68	33.31	33.62	37.31	37.62	<b>38.19</b>
	B	30.69	26.65	25.06	27.77	28.38	30.78	30.91	<b>31.31</b>
18	R	32.69	32.30	31.61	33.32	33.24	<b>34.63</b>	34.58	34.06
	G	36.20	35.69	33.84	37.02	35.91	37.30	37.27	<b>37.47</b>
	B	33.43	31.90	31.11	32.93	33.44	34.87	34.30	<b>35.56</b>
Avg	R	34.71	33.05	32.68	34.06	34.71	36.10	<b>36.23</b>	36.02
	G	38.11	37.10	34.63	38.10	38.08	39.46	39.79	<b>40.15</b>
	B	33.41	32.30	31.87	33.15	33.47	34.33	34.38	<b>34.78</b>

on the results, the proposed algorithm has the best green channel quality with 40.15 dB. LDI-NAT method comes second with 39.79 dB, and LDI-NLM is the third with 39.46 dB. The proposed method offers the best green channel result in 11 images out of 18. In blue channel, the proposed method is 0.40 dB above LDI-NAT with 34.78 dB. And in terms of number of images with the highest blue channel result, the proposed method comes first with 13 images, followed by LDI-NAT and LDI-NLM with 2 each, and DLMMSE with 1. And finally for the red channel, the proposed method is behind LDI-NAT and LDI-NLM with 36.02 dB. Overall, the proposed method offers comparable results to the non-local LDI-NAT and LDI-NLM methods, and significantly better results than the rest of the methods. This is significant because the proposed method uses only local information to interpolate missing pixel values which makes it less computationally complex than non-local methods. To put the complexity into context, it takes 8.5 seconds to interpolate an image with the proposed method while it takes 1460 seconds to process the same image on the same computer with LDI-NAT. Hence, the complexity of the proposed solution is about two orders of magnitude less than a comparable non-local method.



## CHAPTER VII

### A NEW FAMILY OF CFA PATTERNS

#### *7.1 Introduction*

Our work in the demosaicing field until this chapter focused on algorithm design. Namely, we tried to come up with better ways to interpolate a mosaicked image back to a full color image for a given mosaic pattern layout. We described algorithms designed for Bayer and Lukac mosaic patterns. However, there are many more pattern designs proposed in the literature. While some of these CFA patterns (including Bayer and Lukac) are pure RGB based, others are comprised of combinations of RGB channels [18]. Some patterns even incorporate panchromatic pixels into their design [15]. We will refer to the patterns with components that are combinations of RGB channels as mixed patterns.

#### *7.2 Pattern Design*

The Bayer pattern repeats itself in two pixels in both directions. Hence, its smallest building block consists of four pixels. Assuming we start with the green channel, we can simply write the order of these pixels as [GRBG]. Most demosaicing solutions in the literature are developed for this layout. However, if we think in more general terms, we realize that these demosaicing algorithms are agnostic to the actual channels. In other words, a demosaicing algorithm developed for the Bayer pattern simply needs three channels to be arranged in a special order, but the channels themselves can be anything. Let us denote these channels with numbers. Then, we can write the generalized Bayer pattern layout as [1231] and the actual Bayer pattern [GRBG] becomes a special subset of this layout. The question is whether we can find a better

subset than the Bayer pattern.

Our work on images with low spectral correlation led us to think that mixed patterns might lead to better demosaicing results since the spectral correlation between their components are higher than that of pure RGB channels. However, many mixed pattern designs have more than 3 unique channels and their layouts are not as straightforward as the Bayer pattern layout. That is why, we wanted to stick with the generalized Bayer layout [1231] and select its channels so as to maximize their correlation. However, we cannot select all three channels to be linear combinations of RGB colors with nonzero weights for each color because that would lead to amplified estimation errors when we solve for individual color values.

These considerations led us to a hybrid pattern design where we have two components with pure color channels and a third component with a combination of color channels. One way of achieving this is to replace the G channel with a linear combination of RGB and keep R and B channels unchanged in the Bayer pattern. This way we obtain a hybrid pattern that still satisfies the [1231] layout. Compared to the Bayer pattern, this hybrid pattern is expected to have more correlation between its first and second, and also first and third components because these pairs now have common channel inputs.

Table 7 shows the spectral correlation between first and second, and first and third components of the Bayer pattern and the proposed pattern on the 12 image Kodak test set. For the Bayer pattern, these channels are red and green, and blue and green. For the proposed pattern, the green component is replaced with a color combination. Selecting the RGB weights as  $[1/6, 2/3, 1/6]$  respectively leads to equal contributions from all channels in the overall pattern. The results show that on average the proposed pattern has slightly higher spectral correlation between its components than the Bayer pattern. However, we observe that the spectral correlation for the Bayer pattern was already high to begin with on this test set. We wanted to find out if the proposed

**Table 7:** Spectral correlation comparison on the Kodak dataset

	Bayer Pattern		Proposed Pattern	
img no:	red-green	blue-green	red-mixed	blue-mixed
1	0.6168	0.6330	<b>0.7316</b>	<b>0.7374</b>
2	0.9798	0.9915	<b>0.9854</b>	<b>0.9915</b>
3	0.8333	0.9051	<b>0.8835</b>	<b>0.9318</b>
4	0.9682	0.9774	<b>0.9743</b>	<b>0.9794</b>
5	0.9496	0.8536	<b>0.9535</b>	<b>0.8779</b>
6	0.8376	0.9786	<b>0.8773</b>	<b>0.9793</b>
7	<b>0.9854</b>	0.9531	0.9835	<b>0.9599</b>
8	<b>0.9671</b>	0.9182	0.9667	<b>0.9310</b>
9	<b>0.9960</b>	0.9827	0.9952	<b>0.9862</b>
10	0.9024	0.9158	<b>0.9045</b>	<b>0.9202</b>
11	0.8680	0.8815	<b>0.9110</b>	<b>0.9201</b>
12	0.9795	0.9712	<b>0.9845</b>	<b>0.9786</b>
avg	0.9070	0.9135	<b>0.9293</b>	<b>0.9328</b>

pattern led to a more pronounced difference for images with low spectral correlation. That is why we generated the results for the McMaster test set and presented them in Table 8. Figure 19 shows the proposed pattern layout.

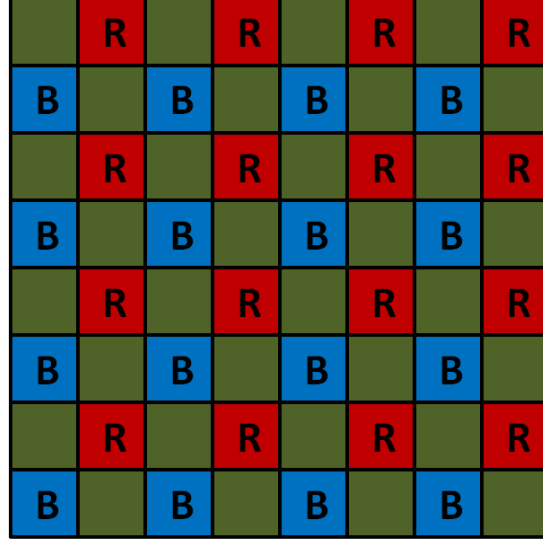
The McMaster dataset has lower average spectral correlation and the average difference between the two patterns is higher than it was on the Kodak set. Now, the question is whether better spectral correlation in the new pattern can lead to better demosaicing performance. As we mentioned earlier, the demosaicing methods designed for the Bayer pattern can also work on the proposed hybrid pattern since both patterns share the same general layout. The only additional step needed is to extract the green channel result from the interpolation output. Assuming the proposed pattern is implemented with the following transformation matrix:

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/6 & 2/3 & 1/6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \quad (73)$$

the original RGB channels are obtained using the inverse transformation matrix

**Table 8:** Spectral correlation comparison on the McMaster dataset

img no:	Bayer Pattern		Proposed Pattern	
	red-green	blue-green	red-mixed	blue-mixed
1	0.6574	0.5122	<b>0.7413</b>	<b>0.6347</b>
2	0.8697	0.8209	<b>0.8961</b>	<b>0.8544</b>
3	0.9281	0.9152	<b>0.9536</b>	<b>0.9460</b>
4	0.9691	0.9716	<b>0.9861</b>	<b>0.9878</b>
5	0.8697	0.8516	<b>0.8931</b>	<b>0.8810</b>
6	0.8429	0.5652	<b>0.8708</b>	<b>0.6407</b>
7	0.8398	<b>0.9965</b>	<b>0.8856</b>	0.9929
8	0.9647	0.9868	<b>0.9744</b>	<b>0.9884</b>
9	0.7783	0.9024	<b>0.8545</b>	<b>0.9251</b>
10	0.5658	0.5795	<b>0.7159</b>	<b>0.6700</b>
11	0.4409	0.5328	<b>0.6479</b>	<b>0.6262</b>
12	0.7695	0.8542	<b>0.8254</b>	<b>0.8795</b>
13	0.8590	0.8339	<b>0.8901</b>	<b>0.8685</b>
14	0.8517	0.6542	<b>0.8791</b>	<b>0.7172</b>
15	0.4824	0.8367	<b>0.6530</b>	<b>0.8951</b>
16	0.9730	0.1802	<b>0.9774</b>	<b>0.2050</b>
17	-0.0100	-0.1142	<b>0.3385</b>	<b>0.1012</b>
18	0.7486	0.9247	<b>0.8404</b>	<b>0.9530</b>
avg	0.7445	0.7114	<b>0.8235</b>	<b>0.7648</b>



$$\text{Green Square} = R/6 + 2G/3 + B/6$$

**Figure 19:** Proposed Bayer based pattern.

after CFA interpolation is completed:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1/4 & 3/2 & -1/4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix}. \quad (74)$$

### 7.3 *Experimental Results*

We tested the performance of several demosaicing algorithms on the Kodak and McMaster datasets with the original Bayer pattern and the proposed pattern. Table 9 summarizes the results for the 12 image Kodak dataset. The demosaicing methods included in the comparison are the Regularization Approaches to Demosaicking (RAD) [34] method and the Directional Linear Minimum Mean Square-Error Estimation (DLMMSE) method in addition to our gradient based (GBTF) and multiscale gradients based (MSG) methods. In the table, R, G, and B columns list the PSNR results for each individual color component and the C column gives the combined

CPSNR results. Our first observation is that the proposed pattern always leads to higher PSNR results for red and blue channels and lower results for the green channel. This is expected because we no longer have known green pixels in the input but on the other hand we have more correlation between the pattern components. We check the combined CPSNR results to find out if the performance is improved overall. Here we see that the CPSNR for the proposed pattern is lower than the Bayer pattern by 0.10 dB for RAD and 0.02 dB for DLMMSE. However, it outperforms the Bayer pattern by 0.08 dB for GBTF and by 0.12 dB for MSG methods. Hence, the CPSNR results of the proposed and Bayer patterns are very close to each other. Another observation from the data is that the performance of the proposed pattern increases as the PSNR level of the employed demosaicing algorithm gets higher.

**Table 9:** Comparison between Bayer and proposed pattern on the Kodak dataset.

img no	GBTF								MSG							
	Bayer				Proposed				Bayer				Proposed			
	R	G	B	C	R	G	B	C	R	G	B	C	R	G	B	C
1	43.35	47.66	44.10	44.67	44.11	46.36	44.70	<b>44.96</b>	43.81	47.95	44.44	45.06	44.57	46.67	45.05	<b>45.34</b>
2	42.11	45.02	41.10	<b>42.45</b>	42.38	43.65	41.48	42.41	42.51	45.55	41.41	<b>42.84</b>	42.80	44.20	41.82	42.83
3	43.57	46.77	42.26	43.82	44.41	45.38	43.18	<b>44.23</b>	44.12	47.41	42.96	44.46	44.99	46.05	43.89	<b>44.89</b>
4	37.26	41.12	37.59	38.34	37.66	39.80	37.88	<b>38.34</b>	37.84	41.86	38.15	38.94	38.28	40.52	38.47	<b>38.98</b>
5	44.29	47.33	43.37	44.70	44.84	45.83	44.13	<b>44.87</b>	44.64	47.74	43.79	45.09	45.26	46.22	44.55	<b>45.29</b>
6	40.66	44.42	41.65	<b>41.97</b>	41.16	43.00	41.83	41.93	41.16	45.03	42.04	<b>42.46</b>	41.66	43.67	42.25	42.45
7	45.33	47.87	43.98	<b>45.45</b>	45.60	46.33	44.45	45.39	45.74	48.35	44.35	<b>45.85</b>	46.05	46.77	44.85	45.82
8	42.35	45.25	41.57	<b>42.79</b>	42.68	43.84	42.00	42.78	42.76	45.76	41.95	43.20	43.12	44.40	42.40	<b>43.23</b>
9	43.27	45.83	41.48	43.18	43.51	44.49	41.95	<b>43.19</b>	43.60	46.29	41.84	43.54	43.89	45.02	42.33	<b>43.60</b>
10	40.94	43.57	39.83	<b>41.18</b>	41.12	42.21	40.17	41.09	41.59	44.31	40.33	<b>41.78</b>	41.82	42.96	40.71	41.73
11	39.60	42.90	39.25	40.30	40.41	41.62	39.90	<b>40.58</b>	39.85	43.14	39.50	40.55	40.69	41.95	40.17	<b>40.87</b>
12	38.09	40.30	36.18	37.87	38.48	39.19	36.59	<b>37.94</b>	38.15	40.66	36.30	38.02	38.58	39.70	36.72	<b>38.16</b>
avg	41.73	44.84	41.03	42.23	42.20	43.48	41.52	<b>42.31</b>	42.15	45.34	41.42	42.65	42.64	44.01	41.93	<b>42.77</b>
img no	RAD								DLMMSE							
	Bayer				Proposed				Bayer				Proposed			
	R	G	B	C	R	G	B	C	R	G	B	C	R	G	B	C
1	42.02	46.32	42.97	43.41	42.42	45.30	43.26	<b>43.50</b>	42.88	47.54	43.82	44.33	43.55	45.93	44.34	<b>44.50</b>
2	40.89	43.80	39.74	<b>41.16</b>	41.05	42.23	39.96	40.98	41.39	43.68	40.41	<b>41.62</b>	41.66	42.17	40.77	41.49
3	42.95	46.19	41.93	43.35	43.40	44.74	42.46	<b>43.44</b>	42.95	46.24	41.77	43.28	43.67	44.58	42.58	<b>43.53</b>
4	36.25	40.04	36.46	<b>37.27</b>	36.51	38.63	36.65	37.16	36.32	39.63	36.65	<b>37.30</b>	36.69	38.25	36.93	37.23
5	42.78	45.84	42.15	<b>43.32</b>	43.05	44.40	42.63	43.30	43.69	46.66	42.90	44.14	44.21	44.97	43.60	<b>44.23</b>
6	39.71	43.27	40.22	<b>40.81</b>	40.05	41.71	40.29	40.62	39.98	43.19	40.93	<b>41.17</b>	40.44	41.61	41.08	41.02
7	44.06	46.86	42.68	<b>44.21</b>	44.19	45.18	42.95	44.01	44.75	46.80	43.47	<b>44.80</b>	44.98	45.13	43.89	44.63
8	40.76	43.88	39.79	<b>41.16</b>	40.91	42.48	40.01	41.02	41.68	44.13	40.85	<b>42.01</b>	41.99	42.59	41.26	41.91
9	42.34	45.07	40.50	<b>42.25</b>	42.43	43.53	40.79	42.10	42.77	44.86	40.92	<b>42.56</b>	42.97	43.32	41.36	42.47
10	40.35	43.08	39.20	<b>40.59</b>	40.47	41.43	39.42	40.36	40.40	42.42	39.32	<b>40.53</b>	40.55	40.88	39.62	40.31
11	39.16	42.20	38.60	39.73	39.63	40.96	38.94	<b>39.76</b>	39.20	42.25	38.81	39.84	39.96	40.79	39.40	<b>40.01</b>
12	37.91	40.23	35.99	<b>37.71</b>	38.15	38.79	36.25	37.59	37.98	39.87	36.20	<b>37.76</b>	38.39	38.52	36.60	37.75
avg	40.77	43.90	40.02	<b>41.25</b>	41.02	42.45	40.30	41.15	41.16	43.94	40.50	<b>41.61</b>	41.59	42.39	40.95	41.59

After confirming that the proposed hybrid pattern offers comparable performance to the Bayer pattern on the Kodak dataset, we move onto the McMaster dataset. We

compared the performance of the patterns using the Local Directional Interpolation Non-local Adaptive Thresholding (LDI-NAT) method [43] and our local low correlation method (LLC) presented in the previous section. The comparison results are summarized in Table 10. The hybrid pattern outperforms the Bayer pattern by 0.48 dB for the LDI-NAT method and by 0.31 dB for the local low correlation method (LLC). Hence, the proposed pattern enables better interpolation quality than the Bayer pattern for these two highest performing methods on the McMaster dataset. A sample image region is presented in Figure 20 for subjective quality comparison.



**Figure 20:** Image 17 interpolated with the low correlation method on the Bayer pattern and the proposed pattern.

#### 7.4 *Extension of the Proposed Pattern*

The proposed change to the Bayer pattern can be extended to other available patterns such as the Lukac pattern. Following the same reasoning with the Bayer pattern, we replace the G channels with a combination of RGB, and leave the R and B channels unchanged. The proposed Lukac based pattern is given in Figure 21.

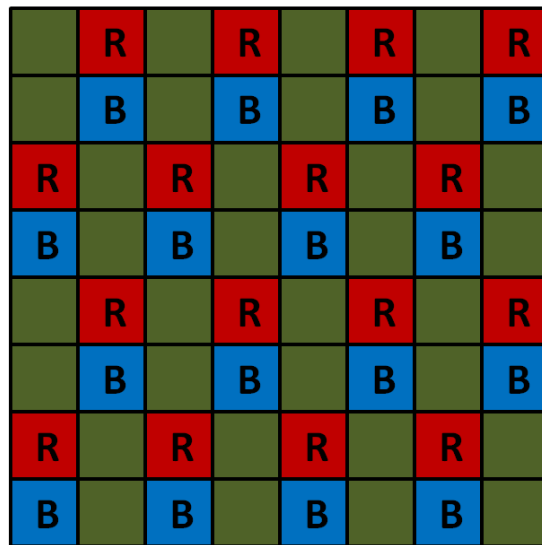
Another dimension for the extension of the proposed pattern is to replace the

**Table 10:** Comparison between Bayer and proposed pattern on the McMaster dataset.

img no	LDI-NAT								LLC							
	Bayer				Proposed				Bayer				Proposed			
	R	G	B	C	R	G	B	C	R	G	B	C	R	G	B	C
1	29.29	32.67	26.71	28.92	29.97	31.80	27.87	<b>29.58</b>	29.20	33.00	26.98	29.08	29.94	31.75	27.86	<b>29.56</b>
2	35.02	39.08	32.92	35.00	35.36	38.24	33.52	<b>35.30</b>	34.83	39.61	32.89	34.98	35.39	38.21	33.53	<b>35.31</b>
3	33.05	35.51	30.31	32.45	34.37	34.57	31.28	<b>33.13</b>	33.64	36.52	30.69	32.99	34.35	34.52	31.27	<b>33.11</b>
4	36.25	40.33	33.30	35.75	37.63	38.26	34.30	<b>36.36</b>	36.19	41.41	33.11	35.73	37.59	38.40	34.31	<b>36.39</b>
5	35.05	38.15	31.16	33.87	34.83	36.49	31.98	<b>34.03</b>	33.99	38.08	31.35	33.67	34.77	36.40	31.97	<b>33.99</b>
6	39.40	43.42	34.97	<b>37.97</b>	38.70	41.08	35.21	37.66	38.07	42.56	34.18	37.04	38.66	40.92	35.17	<b>37.60</b>
7	36.09	37.41	34.49	<b>35.83</b>	36.64	36.02	34.69	35.71	36.45	38.27	34.58	<b>36.17</b>	36.65	35.99	34.70	35.71
8	36.31	40.29	36.67	37.43	37.38	39.78	37.47	<b>38.08</b>	37.04	41.34	37.18	<b>38.12</b>	37.38	39.73	37.45	38.06
9	35.49	41.73	36.30	37.11	36.59	40.66	37.76	<b>38.03</b>	35.76	42.50	37.00	37.60	36.61	40.64	37.75	<b>38.03</b>
10	38.26	42.64	36.83	38.63	38.86	41.26	37.73	<b>39.05</b>	37.63	42.87	36.91	38.46	38.86	41.22	37.75	<b>39.05</b>
11	39.82	42.57	37.66	39.57	39.96	41.39	39.13	<b>40.06</b>	39.29	42.45	38.57	39.80	39.94	41.34	39.12	<b>40.04</b>
12	38.36	41.49	37.59	38.85	39.07	40.73	38.00	<b>39.12</b>	37.92	41.67	37.74	38.77	39.04	40.70	38.04	<b>39.13</b>
13	41.77	44.89	38.13	40.74	41.66	43.91	38.99	<b>41.06</b>	40.96	45.14	38.24	40.61	41.68	43.89	38.99	<b>41.06</b>
14	39.39	42.84	36.12	38.63	39.61	41.88	36.92	<b>39.01</b>	39.44	43.30	36.36	38.84	39.67	41.84	36.93	<b>39.02</b>
15	36.95	42.68	38.99	38.95	37.98	41.71	39.81	<b>39.57</b>	37.30	43.06	39.35	39.31	38.01	41.64	39.81	<b>39.57</b>
16	34.97	35.59	31.53	33.64	34.65	34.90	34.91	<b>34.82</b>	34.80	35.33	34.08	34.71	34.62	34.80	34.89	<b>34.77</b>
17	32.14	37.62	30.91	32.74	33.07	36.15	32.60	<b>33.68</b>	31.74	38.19	31.31	32.84	33.00	35.98	32.54	<b>33.59</b>
18	34.58	37.27	34.30	35.19	34.58	36.59	36.19	<b>35.69</b>	34.06	37.47	35.56	35.48	34.62	36.56	36.17	<b>35.70</b>
avg	36.23	39.79	34.38	36.18	36.72	38.63	35.46	<b>36.66</b>	36.02	40.15	34.78	36.34	36.71	38.59	35.46	<b>36.65</b>

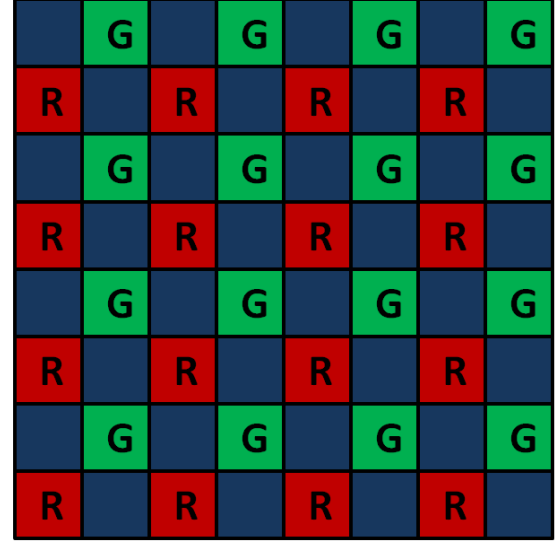
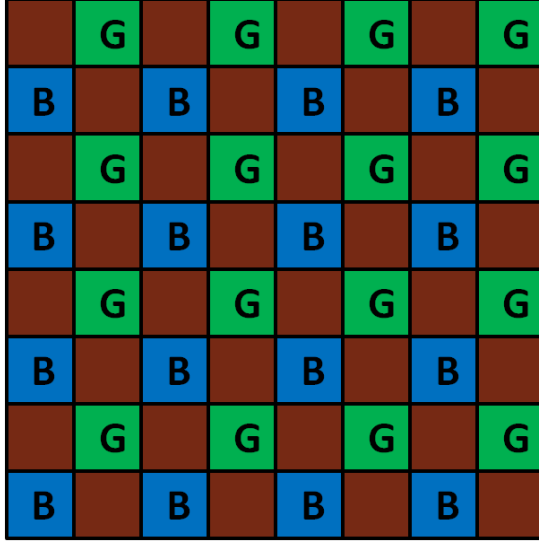
channel locations, i.e. having red and green or blue and green pure channels and the corresponding mixed channels instead of the original red and blue pure channel configuration. Again, these changes do not affect the compatibility of the proposed pattern with the algorithms designed for the Bayer pattern because we still maintain the generalized Bayer layout. Finally, using the same argument, we can expand the proposed Lukac layout pattern by replacing the channel locations. The proposed patterns are illustrated in Figure 22.







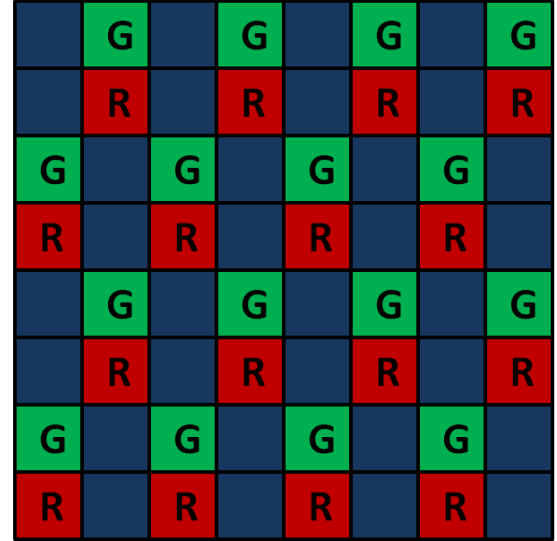
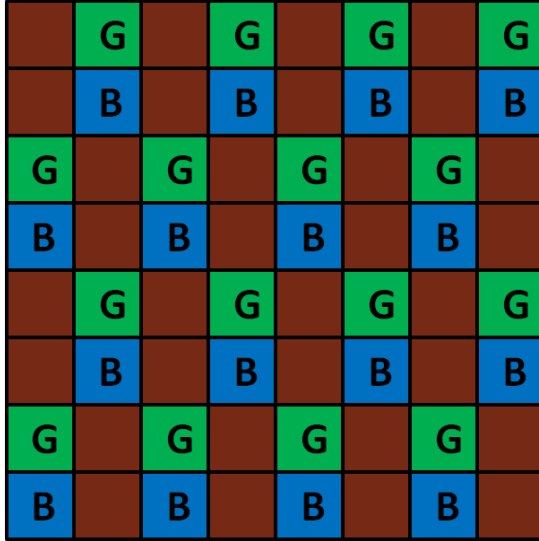
$$\text{Olive Green} = R/6 + 2G/3 + B/6$$

**Figure 21:** Proposed Lukac based pattern.



 =  $2R/3 + G/6 + B/6$

 =  $R/6 + G/6 + 2B/3$



**Figure 22:** Patterns generated by replacing channel locations in Bayer and Lukac based layouts.

## CHAPTER VIII

### CONCLUSIONS AND FUTURE WORK

In this thesis we focused on resolution enhancement and demosaicing areas. We first looked into the image interpolation problem for which edge preservation is an important issue. Instead of taking the edge detection route, we approached the problem from a geometric perspective and derived a general formula to be applied to all pixels without any classification. We then turned our attention to demosaicing, which can be considered a special form of image interpolation. We developed several algorithms with each one giving us more insight to the problem and enabling the development of the next solution.

The spectral correlation between color channels might be the most important source of information for the demosaicing problem. However, this information source may not always be as reliable as one might assume. Highly saturated images with rapid color changes pose an important challenge to demosaicing algorithms. Although non-local averaging can alleviate the complications caused by low spectral correlation, it also increases computational complexity by several orders of magnitude. That is why we pursued a local solution to the low spectral correlation problem, and described the resulting algorithm in this thesis. Finally, we looked into the CFA pattern design problem and generated a new family of hybrid patterns that are compatible with algorithms developed for the Bayer or Lukac pattern.

Future work will focus on extending our single frame image interpolation solution to multiple frames. We believe that the advantages of our approach in the spatial domain may lead to improved super-resolution performance. There are several possible

improvement areas in the super-resolution problem. Firstly, we will investigate replacing the usual interpolation methods found in super-resolution algorithms with our edge directed interpolation algorithm. However, in its current spatial domain form, our algorithm operates on diagonal pixel values. In the super-resolution setting, these values will not be readily available due to random motion between different frames. That is why, we will need to either make an approximation or change our formulation to handle such cases. We also need to find out how much of an effect these choices have on the performance of our edge directed approach. Secondly, we will look into subpixel accurate motion estimation, which is the backbone of super-resolution. We will examine the latest developments in motion estimation research and look for a solution to couple with our interpolation method. Among the current approaches, probabilistic motion estimation shows a lot of promise [39]. If needed, we will try to develop our own motion estimation solution, aiming for a balance between low computational complexity and high motion vector accuracy.

## REFERENCES

- [1] ADAMS, C., “Intersections between color plane interpolation and other image processing functions in electronic photography,” in *Proceedings of SPIE*, vol. 2416, pp. 144–151, 1995.
- [2] ADAMS JR, J. and HAMILTON JR, J., “Adaptive color plane interpolation in single sensor color electronic camera,” Apr. 9 1996. US Patent 5,506,619.
- [3] ATKINS, C., BOUMAN, C., and ALLEBACH, J., “Optimal image scaling using pixel classification,” in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 3, pp. 864–867, IEEE, 2001.
- [4] BAYER, B., “Color imaging array,” July 20 1976. US Patent 3,971,065.
- [5] BUADES, A., COLL, B., MOREL, J., and SBERT, C., “Self-similarity driven color demosaicking,” *Image Processing, IEEE Transactions on*, vol. 18, no. 6, pp. 1192–1202, 2009.
- [6] CAREY, W., CHUANG, D., and HEMAMI, S., “Regularity-preserving image interpolation,” *Image Processing, IEEE Transactions on*, vol. 8, no. 9, pp. 1293–1297, 1999.
- [7] CHANG, S., CVETKOVIC, Z., and VETTERLI, M., “Locally adaptive wavelet-based image interpolation,” *Image Processing, IEEE Transactions on*, vol. 15, no. 6, pp. 1471–1485, 2006.
- [8] CHEN, Q. and WEINHOUS, M., “Subpixel shift with fourier transform to achieve efficient and high-quality image interpolation,” in *Proceedings of SPIE*, vol. 3661, p. 728, 1999.
- [9] CHUNG, K. and CHAN, Y., “Color demosaicing using variance of color differences,” *Image Processing, IEEE Transactions on*, vol. 15, no. 10, pp. 2944–2955, 2006.
- [10] CHUNG, K. and CHAN, Y., “Low-complexity color demosaicing algorithm based on integrated gradients,” *Journal of Electronic Imaging*, vol. 19, p. 021104, 2010.
- [11] FOSTER, G. and NAMAZI, N., “Interpolation and gradient estimation of images using the discrete cosine transform,” in *System Theory, 2002. Proceedings of the Thirty-Fourth Southeastern Symposium on*, pp. 167–170, IEEE, 2002.
- [12] GLOTZBACH, J., SCHAFER, R., and ILLGNER, K., “A method of color filter array interpolation with alias cancellation properties,” in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 1, pp. 141–144, IEEE, 2001.

- [13] GUNTURK, B., ALTUNBASAK, Y., and MERSEREAU, R., “Color plane interpolation using alternating projections,” *Image Processing, IEEE Transactions on*, vol. 11, no. 9, pp. 997–1013, 2002.
- [14] HAMILTON JR, J. and ADAMS JR, J., “Adaptive color plan interpolation in single sensor color electronic camera,” May 13 1997. US Patent 5,629,734.
- [15] HAMILTON JR, J. and COMPTON, J., “Processing color and panchromatic pixels,” July 28 2005. US Patent App. 20,070/024,879.
- [16] HIBBARD, R., “Apparatus and method for adaptively interpolating a full color image utilizing luminance gradients,” Jan. 17 1995. US Patent 5,382,976.
- [17] HIRAKAWA, K. and PARKS, T., “Adaptive homogeneity-directed demosaicing algorithm,” *Image Processing, IEEE Transactions on*, vol. 14, no. 3, pp. 360–369, 2005.
- [18] HIRAKAWA, K. and WOLFE, P., “Spatio-spectral color filter array design for optimal image recovery,” *Image Processing, IEEE Transactions on*, vol. 17, no. 10, pp. 1876–1890, 2008.
- [19] HU, H., HOFMAN, P., and DE HAAN, G., “Content-adaptive neural filters for image interpolation using pixel classification,” *Proceedings of SPIE, Applications of Neural Networks and Machine Learning in Image Processing IX*, 2005.
- [20] JENSEN, K. and ANASTASSIOU, D., “Subpixel edge localization and the interpolation of still images,” *Image Processing, IEEE Transactions on*, vol. 4, no. 3, pp. 285–295, 1995.
- [21] KEYS, R., “Cubic convolution interpolation for digital image processing,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 29, no. 6, pp. 1153–1160, 1981.
- [22] KIMMEL, R., “Demosaicing: image reconstruction from color ccd samples,” *Image Processing, IEEE Transactions on*, vol. 8, no. 9, pp. 1221–1228, 1999.
- [23] LAROCHE, C. and PRESCOTT, M., “Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients,” Dec. 13 1994. US Patent 5,373,322.
- [24] LAVARENE, B., ALLEYSSON, D., DURETTE, B., and HERAULT, J., “Efficient demosaicing through recursive filtering,” in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 2, pp. II–189, IEEE, 2007.
- [25] LEE, S. and PAIK, J., “Image interpolation using adaptive fast b-spline filtering,” in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 5, pp. 177–180, IEEE, 1993.

- [26] LENKE, S. and SCHRÖDER, H., “Classification based polynomial image interpolation,” in *Proceedings of SPIE*, vol. 6812, p. 681214, 2008.
- [27] LI, X., “Demosaicing by successive approximation,” *Image Processing, IEEE Transactions on*, vol. 14, no. 3, pp. 370–379, 2005.
- [28] LI, X., GUNTURK, B., and ZHANG, L., “Image demosaicing: A systematic survey,” in *Proc. IS&T/SPIE Conf. on Visual Communication and Image Processing*, vol. 6822, Citeseer.
- [29] LI, X. and ORCHARD, M., “New edge-directed interpolation,” *Image Processing, IEEE Transactions on*, vol. 10, no. 10, pp. 1521–1527, 2001.
- [30] LIAN, N., CHANG, L., TAN, Y., and ZAGORODNOV, V., “Adaptive filtering for color filter array demosaicking,” *Image Processing, IEEE Transactions on*, vol. 16, no. 10, pp. 2515–2525, 2007.
- [31] LUKAC, R. and PLATANOTIS, K., “A normalized model for color-ratio based demosaicking schemes,” in *Image Processing, 2004. ICIP’04. 2004 International Conference on*, vol. 3, pp. 1657–1660, IEEE.
- [32] LUKAC, R. and PLATANOTIS, K., “Universal demosaicking for imaging pipelines with an rgb color filter array,” *Pattern Recognition*, vol. 38, no. 11, pp. 2208–2212, 2005.
- [33] MENON, D., ANDRIANI, S., and CALVAGNO, G., “Demosaicing with directional filtering and a posteriori decision,” *Image Processing, IEEE Transactions on*, vol. 16, no. 1, pp. 132–141, 2007.
- [34] MENON, D. and CALVAGNO, G., “Regularization approaches to demosaicking,” *Image Processing, IEEE Transactions on*, vol. 18, no. 10, pp. 2209–2220, 2009.
- [35] MURESAN, D., “Fast edge directed polynomial interpolation,” in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 2, pp. II–990, IEEE, 2005.
- [36] MURESAN, D. and PARKS, T., “Adaptively quadratic (aqua) image interpolation,” *Image Processing, IEEE Transactions on*, vol. 13, no. 5, pp. 690–698, 2004.
- [37] PALIY, D., KATKOVNIK, V., BILCU, R., ALENIUS, S., and EGIAZARIAN, K., “Spatially adaptive color filter array interpolation for noiseless and noisy data,” *International Journal of Imaging Systems and Technology*, vol. 17, no. 3, pp. 105–122, 2007.
- [38] PEKKUCUKSEN, I. and ALTUNBASAK, Y., “Gradient based threshold free color filter array interpolation,” in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 137–140, IEEE, 2010.

- [39] PROTTER, M. and ELAD, M., “Super resolution with probabilistic motion estimation,” *Image Processing, IEEE Transactions on*, vol. 18, no. 8, pp. 1899–1904, 2009.
- [40] WANG, Z., BOVIK, A., SHEIKH, H., and SIMONCELLI, E., “Image quality assessment: From error visibility to structural similarity,” *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, 2004.
- [41] XUE, K., WINANS, A., and WALOWIT, E., “An edge-restricted spatial interpolation algorithm (journal paper),” *Journal of Electronic Imaging*, vol. 1, no. 02, pp. 152–161, 1992.
- [42] ZHANG, L. and WU, X., “Color demosaicking via directional linear minimum mean square-error estimation,” *Image Processing, IEEE Transactions on*, vol. 14, no. 12, pp. 2167–2178, 2005.
- [43] ZHANG, L., WU, X., BUADES, A., and LI, X., “Color demosaicking by local directional interpolation and non-local adaptive thresholding,” *Journal of Electronic Imaging*, vol. 20, no. 2, 2011.
- [44] ZHANG, L., “Mcmaster dataset.” <http://www4.comp.polyu.edu.hk/~cslzhang/CDM.Dataset.htm>.
- [45] ZHANG, L. and WU, X., “An edge-guided image interpolation algorithm via directional filtering and data fusion,” *Image Processing, IEEE Transactions on*, vol. 15, pp. 2226 –2238, aug. 2006.



## VITA

Ibrahim Pekkucuksen was born in Konya, Turkey in 1983. He received the BS degree in electrical engineering from Texas A&M University, College Station, in 2005, and the MS degree in electrical engineering from Georgia Institute of Technology, Atlanta, in 2007. He will receive the PhD degree in electrical engineering from Georgia Institute of Technology, Atlanta, in 2011. His research interests include resolution enhancement, demosaicing, and motion estimation.