UNIVERSITY OF NAIROBI

SCHOOL OF ENGINEERING

Department of Electrical and Information Engineering

A Hybrid Heuristic-Based Localised Area Demosaicking Technique for Panchromatic Colour Filter Arrays

By

Kinyua Wachira

F56/69105/2011

A thesis submitted in partial fulfilment for the Degree of Master of Science in Electrical and Electronic Engineering in the Department of Electrical and Information Engineering in the University of Nairobi

July 2018

Declaration

This thesis is my original work and has not been presented for a degree in any other university.

Kinyua Wachira F56/69105/2011

Signature: ____________________________ Date: __________________

This thesis has been submitted for examination with our approval as university supervisors.

Prof. Elijah Mwangi

First Supervisor, University of Nairobi

Signature: ____________________________ Date: __________________

Prof. Gwang-gil Jeon

Second Supervisor, Incheon National University, Republic of Korea

Signature: ____________________________ Date: __________________

This work has been passed through the *Turintin*® Plagiarism and Similarity Checker software. The resulting originality report and similarilty statistics are to be found at the end of this thesis document.

Dedication

This work is dedicated to my parents, family and friends.

Acknowledgment

The author would first and foremost like to thank his supervisors Prof. Elijah Mwangi and Prof. Gwang-gil Jeon for their unwavering support, insightful criticism and patient guidance in the completion of this work.

The author would also like to extend his gratitude to Prof. Heywood Ouma Absaloms & Dr. Wilfred N. Mwema of the University of Nairobi; Dr. Keigo Hirakawa of the University of Dayton, Ohio in the United States of America; Prof. Eric Dubois of the University of Ottawa in Canada and Dr. Xiangdong Chen of Nanjing University of Posts and Telecommunications in the People's Republic of China for their willingness to discuss and critique this work.

Special thanks to the Kenya Education Network (KENET) for providing several student travel grants to the author to present this work in peer-reviewed international conferences and publish in peer-reviewed periodicals.

Finally, the author would like to acknowledge the moral, financial support and gentle patience of family and friends over the many years it has taken to complete this work.

Abstract

Images have always been and remain a common and integral mode of human communication. In the 21st century, a two-pronged communications revolution in the form of mobile hand-held devices and social media has led to a higher reliance on visual communication through digital photographic images. A large portion of these digital images are captured through embedded cameras integrated in mobile devices. More so than in older stand-alone digital cameras.

To generate colour images while maintaining an affordable camera cost, a spectrally selective filter is placed on top of the raw data camera sensor. This filter termed a colour filter array, or CFA, subsamples colour data in a scene and a software-defined interpolation process termed demosaicking is performed later to fully reconstruct the image taken to make it more representative of the original scene. Many camera manufacturers employ the original array called the Bayer array. Recent studies, however, have shown that a newer array class referred to as panchromatic colour filter arrays possess superior light intensity properties and has a spectral selectivity distribution more in line with the human visual system than the prevalent Bayer array. This is an attractive property that can be exploited primarily in low to medium resolution integrated cameras that form a significant percentage of mobile device cameras.

Demosaicking is primarily biased to the Bayer array due to its prevalence. However, more and more manufacturers are beginning to explore alternatives to the Bayer array to improve image quality. This work presents a novel demosaicking algorithm for panchromatic arrays; in particular the RGBW panchromatic array class that is the most promising panchromatic array. The algorithm encodes light intensity information in a Bayerisation conversion process and uses combinatorial geometry, specifically polyominoes, to provide a novel adaptive weighting mechanism to reduce the introduction of visual artefacts during image interpolation. Interpolation is done in the ordinal directions and a variable plane relationship is established. A corrective mechanism is also introduced into the algorithm to improve image acuity.

Performance of the proposed demosaicking algorithm is objectively assessed using four documented image quality assessment metrics (MSE, CPSNR, SSIM and $FSIM_C$) over five standard image sets (USC-SIPI, Kodak, McMaster-IMAX, Condat, ARRI) and one user-defined custom image set. Each set allows for the analysis of a unique property encountered in mobile device camera photography. This assessment is performed through simulation using the *MATLAB*® software platform. The algorithm results in a lowering of MSE by a factor of 1.6 and a rise in CPSNR by at least 2.49 dB in the RGBW domain. The algorithm also produces a robust SSIM and $FSIM_C$ profile with values greater than 0.98 in both measures. These improvements, noted through the aforementioned image assessment metrics, justify further adoption and study of the newer panchromatic class of arrays in integrated cameras.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Acronyms

ACR – Average-based Colour Reconstruction algorithm

arri_im – member of the ARRI Image Set database

BI – Bi-linear Interpolation

B.C. – Year Before Christ (unit of measure)

BCI – Bi-cubic Interpolation

CCD – Charge-coupled Device

CDBI – Constant Difference Based Interpolation

CFA – Colour Filter Array

CFM – Colour Filter Mosaic

CMOS – Complementary Metal Oxide Semiconductor

codim – member of the Laurent Condat Image Set database

cusim – member of the Custom Image Set database generated by the author

CPSNR – Colour Signal-to-Noise Ratio

CRBI – Constant Ratio Based Interpolation

dB – decibels (unit of measure)

DSC – Digital Still Camera

EDCR – Edge Detection-based Colour Reconstruction algorithm

EDI – Edge Directed Interpolation

ESFBI – Edge Strength Filter Based Interpolation

EM – Electromagnetic Spectrum

FSIM – Feature Similarity Index

$FSIM_C$ – Feature Similarity Index with chrominance included

GBD – Gradient-based Demosaicking

IEEE – Institute of Electrical and Electronic Engineers inf. – Infinite Value (unit of measure)

JPEG – Joint Photographic Experts Group

kodim – member of the Kodak Image Set database

mcm – member of the McMaster-IMAX Image Set database

McGill – the McGill Image Set database

MHC – Malvar-He-Cutler algorithm

MHD – Mobile Hand-held Device

MP – Mega pixel (unit of measure)

POCS – Projection onto Convex Sets

PSNR – Peak Signal-to-Noise Ratio

RGB – Red Green Blue Colour Space

RGBG – Red Green Blue Green Colour Filter Array

RGBW – Red Green Blue White Colour Filter Array

ROI – region of interest

sipi_im – member of the USC-SIPI Image Set database

SOD – Sum-of-Differences

SPIE – Society of Photo-optical and Instrumentation Engineers

SSIM – Structural Similarity Index

Win – Windows 7, 8, 8.1 and 10 Default Image Set

YIQ – Luminance (Y) In-phase Quadrature Colour Space

1 INTRODUCTION

1.1 Background

A common idiom in the English language states that a picture is worth a thousand words. Any image is classically defined as a two-dimensional function used as a non-linguistic communication medium. It describes information in a spatial manner bound in the two dimensions. Three dimensional images are often described as a superset of their two dimensional counterparts.

For millennia, images have played a vital role in helping humans communicate ideas. Cave paintings are described as one of the earliest forms of communication. Figure 1.1 illustrates four cave painting images. The Pettakare Cave images in Indonesia were created around 33,000 B.C. [1] while those in Cueva de las Manos in Argentina are from 11,000 to 9,000 B.C. [2]. Both sets of paintings are believed to have been generated as part of a communal activity. Lion and mammoths were depicted in France’s Chauvet Cave created around 30,000 B.C. and hunting and dancing activities occurring around 6,000 B.C. are shown the Serra da Capivara in Brazil [3].

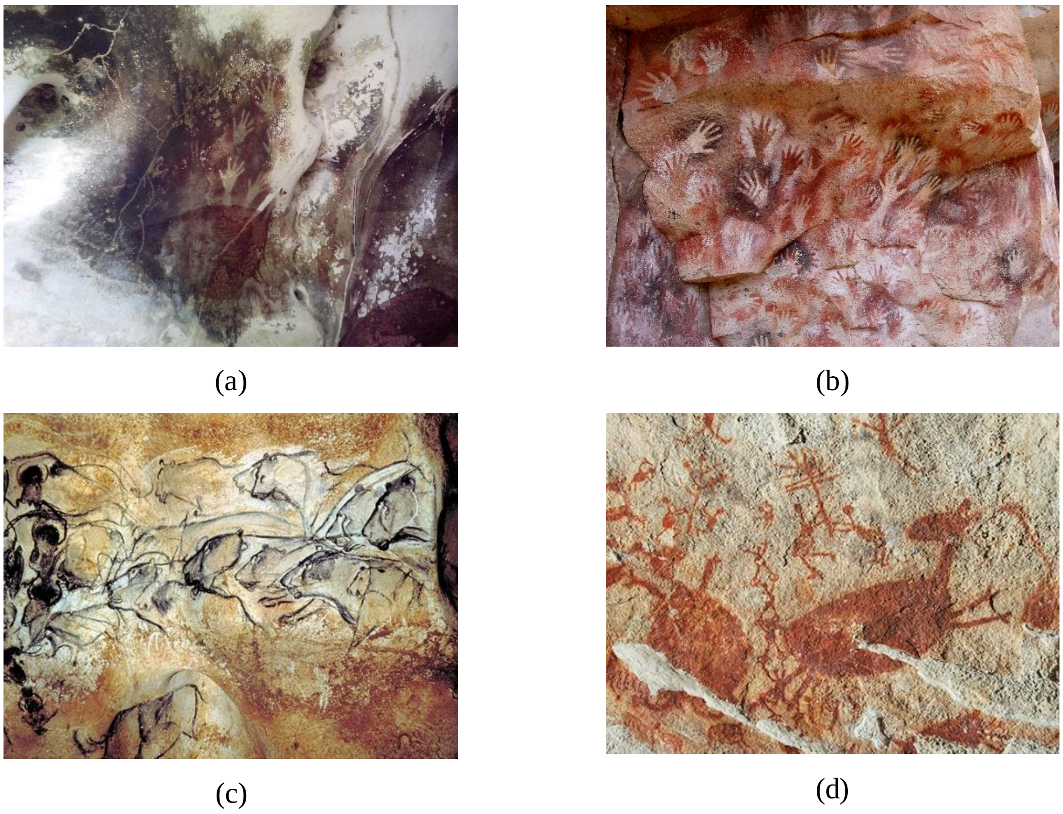


Figure 1.1 Cave paintings: (a) Pettakere Cave in Indonesia, (b) Cueva de las Manos in Argentina, (c) Chauvet Cave in France and (d) Serra da Capivara in Brazil

Looking ahead, images have been used in an attempt to communicate with intelligent beings that may inhabit the extra-Solar region. The Pioneer plaque [4] on Pioneer 10 and the Arecibo message [5] both shown in Figure 1.2 are images that carry information about Earth's position in the Solar system (shown in yellow), information on human helix DNA composition (shown in blue/white and purple), human physiology (shown in red), the population at the time and our decimal numbering system (shown in white).

From its basic definition, an image may refer to a graph, drawing, computer rendering, photograph, logo, pictogram, painting or map. However, the term image is often used to describe a visual copy of an object or set of objects in a scene captured by an optical medium. Optical devices can be man-made using mirrors and lens-based devices such as telescopes, microscopes and cameras. Naturally occurring optical devices include the eyes and water bodies with reflective properties. This work is devoted to camera-based image capture and its perception using the human eye.

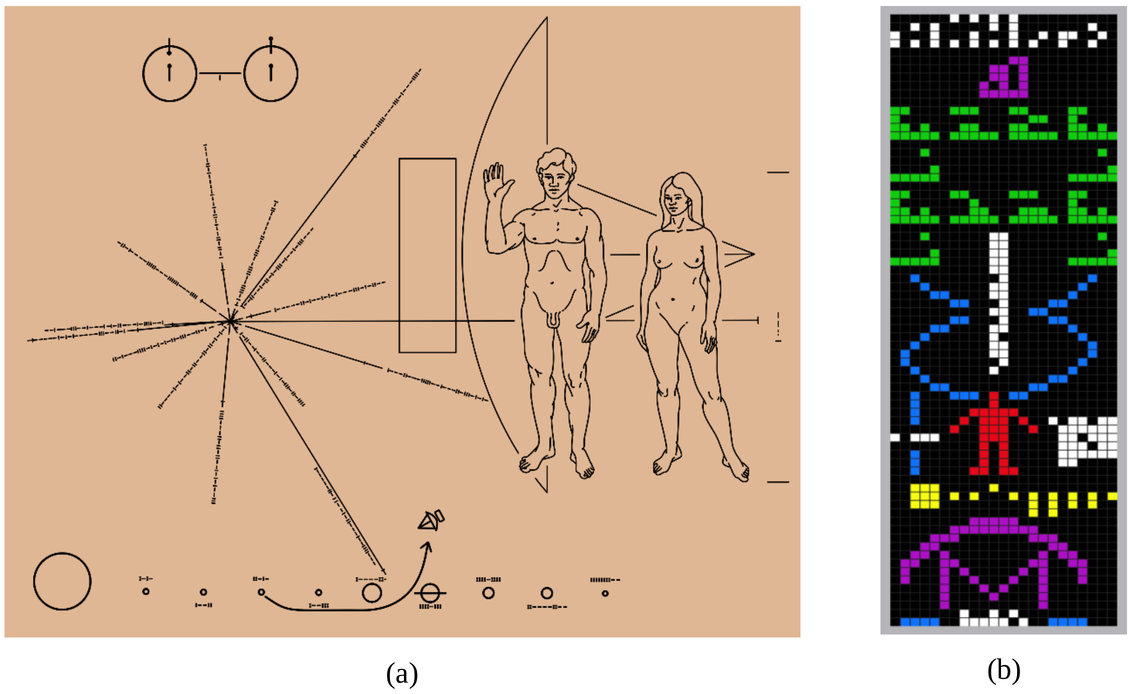

(a) (b)

Figure 1.2 Extra-Solar missives: (a) Pioneer plaque and (b) Arecibo message

The 21st century has seen a two pronged communications revolution; both the manner and the tools with which human beings communicate have changed dramatically.

The key catalyst of this change is the worldwide proliferation of mobile hand-held devices (MHD). This term covers mobile cellular telephones (cell phones), tablets, embedded systems and wearable technology. This is due to advances in circuit component miniaturisation that have led to the mass production and affordable cost of these devices. In particular, the cell phone has become a ubiquitous feature of modern society. In 2016, it was estimated that there were over 4.66 billion mobile phone users [6] and over 7 billion mobile phone user subscriptions [7]. This has led to the mobile phone being termed as the world's first "truly personal computer".

The way communication is done has also changed. The last 10 years has seen the advent of social networking and the social media age. This coupled with mobile telephony has led to more and more people interacting with one another than ever before. In a single day, people worldwide send 8.3 trillion text messages and in 2017, it is projected that 1.016 billion images will be taken using smart mobile devices [8], [9]. The trend in the last decade has been from an audio-only communication to a multimedia rich communication norm [10].

Many mobile devices have an integrated digital still camera (DSC). Due to this, more and more consumers are also starting to use images rather than words to communicate ideas. Wholly image-based social media service *Snap Inc.* (formerly *Snapchat*) has seen a threefold increase in active users in the last two years when compared to *Twitter* that is a wholly text-based service [11]. *Instagram*, another image-based social platform has more than 400 million active daily users and at least 600 million active monthly users all sharing images [12]. Low cost integrated cameras are also being used with embedded microprocessor/microcontroller systems such as the *Raspberry Pi* and *Arduino* platforms in all manner of monitoring systems; from plant phenotyping [13] to space exploration and surveillance [14], [15].

1.2 Justification

The human visual process is physical, physiological and psychological by design [16], [17]. The physical part of the process involves the creation of an image to be transmitted to the human eye and this forms the study of optics. The physiological elements of the visual process are the human eye, associative connective pathways and the brain. They are concerned with the passing of the image from the human eye to the visual and visual associative cortices in the brain. Finally, the psychological process attempts to provide an interpretation to the image. This three stage process that consists of seeing, analysing and interpreting images is referred to as human visual perception.

The physiological and psychological sections are highly variable and differ from individual to individual. Physiology structures may be similar but individual variations exist. Interpretation similarly is subject to variation between individuals. A recent example of this variability was observed in February 2015 when a washed-out image now called "the dress" [18]–[21] was uploaded to social media leading to a worldwide debate on its colour. A copy of the original image (and its colour corrected variants) is shown in Appendix C. Over 10 million "tweets" were sent on whether the dress in the image was blue and black or white and gold. Different people perceived the two widely different colour schemes when presented with this image due to physiological and psychological differences.

To ensure correct communication of image information, the physical process of image creation which is the only invariable process must be optimised. In the case of “the dress” image, the washed out effect of the image was identified as the key cause for the varying responses. This highlights the critical nature of proper image creation or image capture.

With images becoming more important in the transfer of information, the way these images are generated becomes critical. Mobile telephones are equipped with low, medium and high resolution integrated digital cameras. The cost of mobile telephones on the whole grows with the resolution of its integrated camera. A higher camera resolution allows more sample points to describe the image being taken. This results in a more accurate facsimile of the scene being captured. However, it also leads to a more expensive image sensor being integrated to the mobile device whose cost of fabrication is passed on to the consumer.

This work intends to use image processing techniques rather than electronic fabrication to improve image capture. The cost of the improvement becomes the mathematical complexity of the algorithm rather than the cost of fabrication of the sensor. By employing an image processing algorithm in reproducing the scene, the author intends to demonstrate that low or medium resolution cameras can still take high quality images of the scenes they capture.

1.3 Problem Statement

The pervasion of mobile hand-held devices integrated with digital cameras is becoming the norm in our society. These devices are expected to represent half of the total online activity by 2018 [22]. Also with the emerging trend of image-based rather than text-based communication, a proper understanding of image generation is warranted.

The work presents an image processing demosaicking algorithm that employs sampled data collected from an image sensor to produce a high quality image of the scene being captured. The image sensor samples and measures both light intensity (luminosity) and colour (chromaticity). A demosaicking algorithm uses the sampled information to fully reconstruct an image of the scene. As such, the problem is one of image reconstruction and enhancement.

This designed algorithm is created to operate with a particular subclass of image sensors called panchromatic colour filter array sensors. These sensors are low cost and have attractive light sensitivity properties [23] suitable for mobile hand-held devices when compared to more traditional sensor referred to as the Bayer sensor.

The work also presents a heuristic approach to solving this image reconstruction problem. The reconstruction cannot be done by simple interpolation of the samples taken due to the fact that the elements within a scene are never predetermined. While no exact reproduction of missing components may be possible – a high quality approximation is just as good. This approach is to ensure the creation of a computationally inexpensive algorithm.

1.4 Objectives

1.4.1 Main Objective

The main objective is to create a robust heuristic demosaicking algorithm to reconstruct images taken using low and medium resolution panchromatic image sensors.

1.4.2 Specific Objectives

The specific objectives in this work are:

i. To study and determine how the camera mimics the human visual system and generates discernable colour images.
ii. To determine the main types of artefacts generated in the camera image processing pipeline and how to suppress them if possible; particularly for low to medium mobile phone cameras.
iii. To generate a heuristic based demosaicking algorithm to reconstruct images taken by a low resolution camera as well as medium resolution cameras.
iv. To compare performance of the algorithm with previously established methods by employing several established metrics, namely mean square error (MSE), colour peak signal-to-noise ratio (CPSNR), feature similarity index (FSIM) and structure similarity index (SSIM).
v. To observe the algorithms robustness to image variance by exposing it to several standard and custom image sets.

1.5 Scope of Work

The work focuses on images generated by low or medium resolution mobile phone cameras. High resolution images taken by high end mobile hand-held devices are not intrinsically considered.

The test-bed of images uses five standard image sets and one custom image set. In order to test and compare the performance of the algorithm with established techniques, the work assumes and employs the image sets as ground truth references. Therefore, no non-reference image performance metrics are used.

The work presents all forms of visual artefacts that commonly present themselves in the image reconstruction process. However, the algorithm design focuses on those particular artefacts produced by demosaicking and looks for ways to mitigate them. This is because these types of artefacts are the most commonly observed and are invariant of the camera physical characteristics.

The work presents several classes of demosaicking algorithms but constrains itself to the spatial sub-class of heuristic demosaicking methods. This demosaicking algorithm type is mature with a large number of state-of-the-art methods that can form an adequate comparison test bed.

1.6 Organisation of the Thesis

The remainder of this thesis is organised as follows. Chapter 2 provides a review of the types of sensors in common use. It also highlights the concept of visual artefacts and details the classes of demosaicking algorithms that have been developed over time to migitate their undesirable effects. The chapter concludes with an analysis of the knowledge gaps present in literature and the contribution of this work in addressing those gaps.

Chapter 3 provides a theoretical framework with insight on the decisions governing the choice of the CFA, algorithm class, image sets and assessment mechanisms. Chapter 4 details the algorithm design process showing the Bayerisation process and the creation of the gradient based algorithm. It also shows some of the synthesis of the novel notions influencing the design choices taken and assumption made prior to the simulation process.

Chapter 5 shows the simulation process, experimental testing and results generated using the designed algorithm working in the *MATLAB®* environment with particular comparisons made to established techniques. Chapter 6 presents a detailed analysis and discussion of the results generated. In particular the colour reconstruction performance and object fidelity measures are scrutinised. Chapter 7 gives a conclusion to the work, lists publications derived from this work and provides recommendations for any subsequent study in the area. The list of References then follows.

A series of appendices containing supplementary information are then provided and are as follows: Appendix A provides the *MATLAB®* algorithm blocks used in the work for image acquisition, image demosaicking and image comparison. Appendix B provides the raw image quality assessment data generated from the *MATLAB®* simulation. Appendix C presents all the image sets used. Appendix D shows the spectral curves underpinning the trichromatic nature of human vision. Appendix E illustrates the different forms of visual artefacts and aberrations that may be encountered in an image. Appendix F presents some publication statistics in the area of demosaicking from several widely used publication repositories. Appendix G contains the publications resulting from this study that have been internationally peer-reviewed and published in referreed peiodicals. Finally, Appendix H presents the similarity statistics for this entire document from the *Turnitin®* plagiarism checker platform.

2 LITERATURE REVIEW

2.1 The Human Eye and Colour Vision

The eye is the starting point in the physiology of human vision. It contains a naturally occurring image capture setup in the form of a lens and a retina. The retina contains two types of light sensitive photoreceptor cells termed rods and cones that help to capture image information [16], [24]. The rods and cones are sensitive to two types of light [25]. These are:

i. *Achromatic light* that is recorded by the rod photoreceptor cells. Rods are extremely sensitive and are triggered at low-light. In this case, the rods act alone leading to a monochromatic signal void of colour content and the light is termed achromatic.
ii. *Chromatic light* that is recorded by the cone photoreceptor cells. Three types of cones are present in the eye and they are triggered within the 400nm – 700nm wavelength range of the electromagnetic (EM) spectrum. The result is three different colour signals and the light is termed chromatic.

Chromatic light and images formed by them are attractive because colour is a powerful descriptor in images. From a communication perspective, chromatic images yield more information than their achromatic equivalents. Proof of this is evident in image display devices transitioning from monochromatic devices to full colour. The Young-Helmholtz theory postulated the trichromatic nature of human vision as early as the 19th century [16]. In 1956, George Wald et al. [26], [27] proved this theory empirically by showing that each of the three types of cone photoreceptors contained a type of light sensitive protein-base termed *Opsin* that reacts to a particular wavelength range within the EM spectrum. This is shown in Table 2.1 below.

Table 2.1 Spectral properties of human cone photoreceptors

Cone Type	**Spectral Curve**	**Approximate Light Sensitivity Range in nm**	**Approximate Peak Wavelength in nm**
S-cone (OPN1SW or blue sensitive *Opsin*)	β	400 – 550	440
M-cone (OPN1MW or green-sensitive *Opsin*)	γ	450 – 630	545
L-cone (OPN1LW or red-sensitive *Opsin*)	ρ	470 – 700	580

It can be noted from Table 2.1 that while the L-cone does not have a spectral peak explicitly within the red region of the EM spectrum, it is still called red-sensitive *Opsin* because of its proximity towards the red region compared with the other two cone types. The spectral curves and their characteristics are shown in Appendix D.

2.2 The Camera and the Image Processing Pipeline

A camera is a man-made image capturing device that borrows its design from human physiology. It works using the intromission theory of vision that was experimentally proven by Ibn al-Haytham in the 11th century [28]. Its primary components are an aperture, lens and sensor (or film) arrangement corresponding to the pupil, lens and retina of the human eye as illustrated in Figure 2.1.

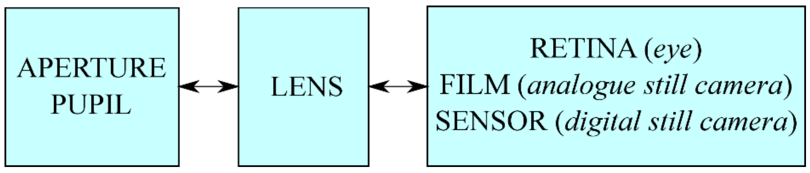

APERTURE
PUPIL
LENS
RETINA (*eye*)
FILM (*analogue still camera*)
SENSOR (*digital still camera*)

Figure 2.1 Basic image capture components in the eye and camera

The aperture and lens sections are common to all cameras. An analogue still camera uses a photographic film coated with a gelatine emulsion containing light sensitive silver halide crystals [29]. Light impinging on the film alters the crystals. Developing chemicals are then used to bring out the image. Digital still cameras (DSC) are the predominant form of camera in use worldwide. The two dominant digital camera technologies are the Charged Coupled Device (CCD) sensor-based cameras and the Complementary Metal Oxide Semiconductor (CMOS) sensor-based type. Of the two leading technologies, the CMOS sensors have a higher market penetration due to advances in circuit component miniaturisation and mass production [30], [31]. In addition, the CMOS sensors are also low power consumption devices. As such, they are also widely available as integrated elements in other electronic components such as mobile hand-held devices.

The image processing pipeline involved in a digital still camera from the observation to the display of a scene is a three phase process [32]. The phases illustrated in Figure 2.2 are:

i. Image acquisition
ii. Image processing
iii. Image storage and display

In the acquisition phase, light reflected off the surface of objects within a scene passes through an aperture and lens arrangement. Exposure, focus control and other mechanical operations are performed in this arrangement block. The light intensity and chromaticity information is then transmitted to the camera sensor assembly for recording. A filter array may be used to sub-sample the light information prior to its being recorded by the sensor.

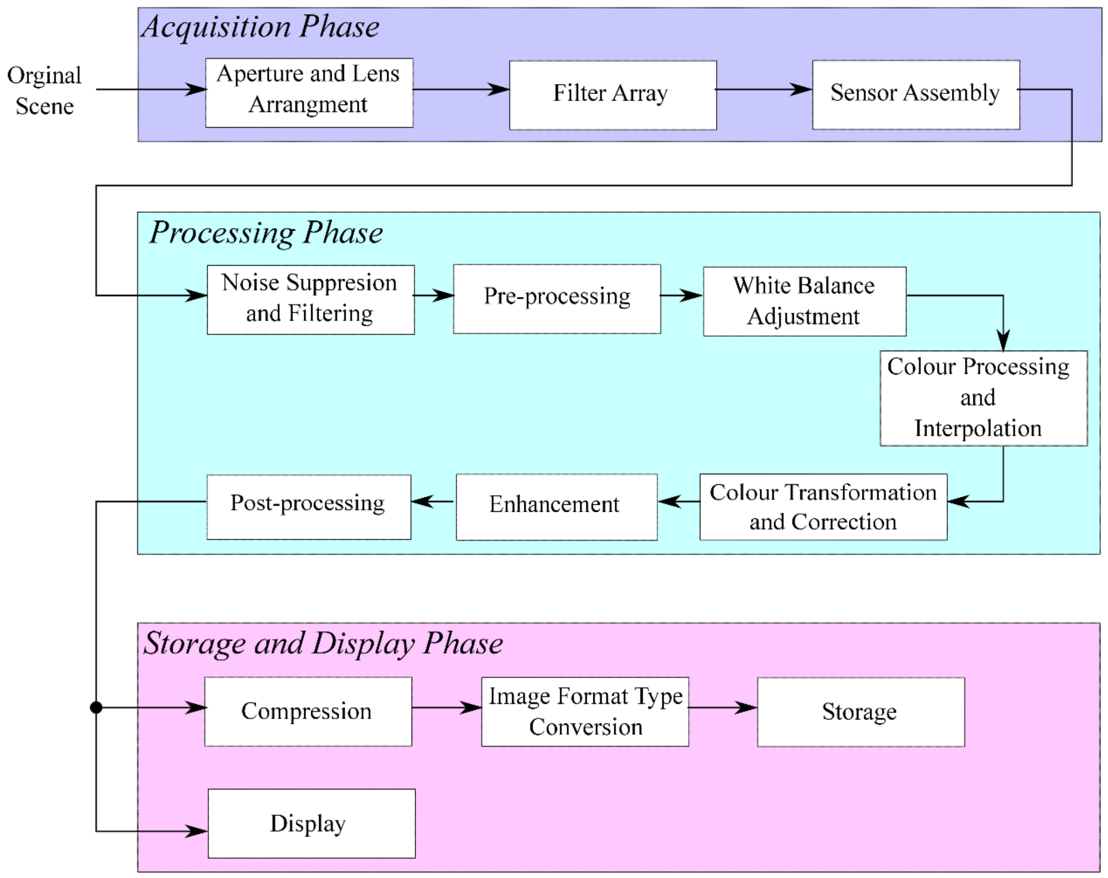

Acquisition Phase
Orginal Scene
Aperture and Lens Arrangment
Filter Array
Sensor Assembly
Processing Phase
Noise Suppresion and Filtering
Pre-processing
White Balance Adjustment
Colour Processing and Interpolation
Colour Transformation and Correction
Enhancement
Post-processing
Storage and Display Phase
Compression
Image Format Type Conversion
Storage
Display

Figure 2.2 Image processing pipeline in a digital still camera (DSC)

The processing phase involves conversion of sensor data to meaningful information that can be displayed to a user. Noise components are filtered out. Some pre-processing and white balance adjustment is performed prior to the start of colour processing. Colour processing involves analysing the sensor data and establishing the appropriate colour content. In particular, when a filter array is used in the acquisition phase, the colour processing will involve an additional interpolation step, termed demosaicking, to reconstruct missing colour content removed by the filtering process. After the colour content is fully processed, the image may require an adjustment in the colour gamut or range so that an image displayed on the LCD display conveys optimum information of the scene. This is because the device may be incapable of displaying all visible colours [16]. This is accomplished in the colour transformation and correction step. Additional enhancement and post-processing are employed before the image is either stored or displayed by the device in the final phase of the pipeline.

Image reconstruction is focused primarily in the filter array and sensor portion of the acquisition phase and the colour processing, interpolation and enhancement sections of the pipeline. This is because these sections are largely device invariant. Other elements are subject to specific device construction and manufacturer choices. Due to this, three sensor schemes have been developed to work in the DSC filter and sensor arrangement. These are:

i. Layered sensor scheme
ii. Three-sensor or Tri-sensor scheme
iii. Single sensor scheme

2.2.1 Layered sensor scheme

This scheme involves a combined filter-enabled sensor placed after the optical system (after the aperture and lens arrangement). As light passes through the sensor, stack sections absorb only one of the trichromatic colours and record position and intensity information. The unabsorbed colours pass through to be recorded elsewhere in the sensor stack [33]. This is illustrated in Figure 2.3. The sensor data is then passed through to finally generate the colour image.

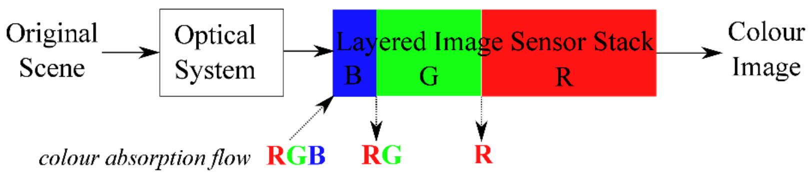


Figure 2.3 Layered sensor device

The stack colour region ordering does not matter because at each stage, the stack regions only absorb their corresponding colours. The length of the absorbing regions however matters with the region nearest to the lens/aperture receiving the highest amount of light intensity. As the light traverses the stack, its intensity drops and therefore, the stack regions correspondingly increase in size in order to ensure a uniform reading. The complexity of producing the sensor stack lead to cost and mass production limitations. As such, this scheme is not used in commercial integrated DSCs but finds application in specialised discrete (non-integrated) cameras.

2.2.2 Three sensor scheme

The three sensor scheme shown in Figure 2.4 uses three independent filters and sensors; each corresponding to one of the trichromatic colours. Light from the optical system is passed to each of these filters. The filtered component is then recorded by the associated sensor. The data from all three sensors is then combined and passed for processing to generate a full colour image [33], [34].

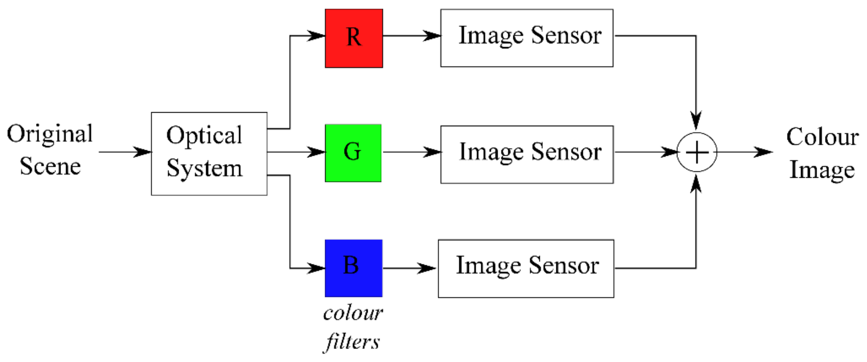


Figure 2.4 Three sensor device

The duplication in functionality by using multiple filters and sensors increases cost and device complexity. Consequently, this scheme is not employed in mobile hand-held device cameras but in high-end, high performance discrete cameras [35].

2.2.3 Single sensor scheme

The single sensor scheme makes use of a single filter called a colour filter array or colour filter mosaic (CFA/CFM). Individual colour lattices are combined to form a single array. Instead of having multiple filters, the array allows certain colours to be absorbed at certain pixel location points in the filter. This leads to the image sensor receiving sub-sampled colour data at all pixel location points. This is because rather than having the three colours at one pixel point, only one colour is sampled. Colour interpolation is done on the raw sub-sampled data in the processing pipeline in a process referred to as demosaicking. A full colour image is then generated. This scheme is illustrated in Figure 2.5.

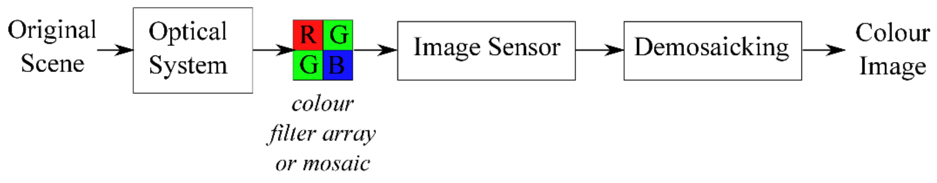


Figure 2.5 Single sensor device

Since the reconstruction is a software process, the device arrangement uses single components that are easy to fabricate and the entire assembly can be made robust. As such, virtually all integrated cameras use this scheme [35]. Many robust discrete DSCs, for example action cameras, also work with a single sensor scheme employing a colour filter array. For this reason, the author’s work is predominantly biased in analysing this class of sensor.

2.3 The Colour Filter Array (CFA) and the Demosaicking process

The colour filter array is a spectrally-selective, tessellate filter found in single sensor based devices [36], [37]. The array is made up of a replication of cells. Each cell is in turn composed of several elements as shown in Figure 2.6. Each element is responsible for filtering a specific colour component of incident light. The arrangement of the filter is done such that each array element is placed directly above one pixel point location of the image sensor.

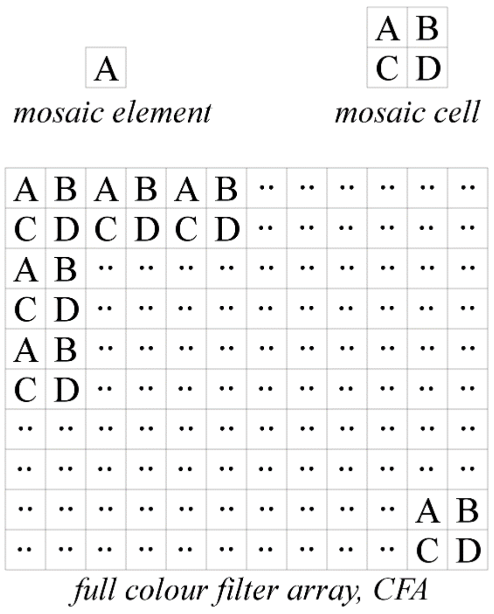


Figure 2.6 A generic CFA and its constituent components

Mathematically, the general colour filter array is a subset of a square plane tessellation structure with a Schläfi symbol of {4, 4}. Exceptions to this rule exist and they incorporate unique triangular, pentagonal or hexagonal tile structures [38]. However, these structures also alter the fabrication of the standard image sensor and the image processing algorithm leading to increase in cost and hence lowering their potential market penetration.

2.3.1 Classification of Colour Filter Arrays

In the history of integrated DSCs, various colour filters have been developed by numerous manufacturers. All filters, quadrille or otherwise, can be broadly classified into three dominant mosaic types. They are the additive mosaics, subtractive mosaics and panchromatic mosaics. Some commonly reported forms of each type are highlighted in Table 2.2

The additive CFAs are based on the additive concept in trichromatic theory. The filter allows the desired colour content to pass through while limiting the other two. By sampling the three primary colours, the trichromatic range of human vision is covered. After colour interpolation/demosaicking, the colours can be combined or added to form the full colour image. The Bayer CFA [39] was the first array of this type to be developed. Other additive colour inspired CFAs include the Yamanaka [40], Watanabe et al. [41], Roddy [42] and the diagonal stripe filters [35] .

Table 2.2 The development trend of various colour filter arrays

Additive Colour Based CFAs	R G G B Bayer, 1976	G R G B G B G R Yamanaka, 1977	G R G B B G R G Watanabe et al., 1985	R G B C G R C B B C R G C B G R Roddy, 2006	R B G G R B B G R Diagonal Stripe, 2008
Sub-tractive Colour Based CFAs	M C G Y M Y G C Morimura et al., 1986	M Y C G Hamilton et al., 2001	C M B Y Bean, 2003	M Y C G G C Y M Hirakawa et al., 2007	
Pan-chromatic Based CFAs	W R W B B W R W W B W R R W B W Dillon, 1977	W G W G R W B W W G W G B W R W Yamagami et al., 1994	M W W Y Bawolek et al., 1999	W R B G Gindele et al. and Sugiyama, 2002; 2005	W B W G B W G W W G W R G W R W Kodak A, 2007
	G W R W G W R W B W G W B W G W Kodak B, 2007	G W R W B W G W G W R W B W G W Kodak C, 2007			

The subtractive class of arrays are based on the use of the primary dye colours of magenta (*M*), yellow (*Y*) and cyan (*C*) instead of the primary light colours of red (*R*), green (*G*) and blue (*B*). Rather than the filter allowing the desired colour through, in the subtractive scheme, the filter colour is absorbed [16]. The colour component that is not part of the subtractive pigment passes through onward to the sensor. For incident white light (*W*), equation (2.1) demonstrates the subtractive process

$$\begin{aligned} W - M &= G \\ W - C &= R \\ W - Y &= B \end{aligned} \tag{2.1}$$

Subtractive colour CFAs include those developed by Morimura et al.[43], Hamilton et al. [44], Bean [45] and Hirakawa et al [46].

Panchromatic colour filter arrays allow full incident white light in some pixel point locations. The incident white light contains the full colour spectrum hence the term panchromatic. Dillon [47], Yamagami et al. [48], Bawolek et al. [49], Gindele et al. and Sugiyama [50], [51] and the three Kodak variants [35] are common implementations of this class of filter.

2.3.2 The Demosaicking Process

After the light passes through the colour filter array, each mosaic element filters one particular colour component to its associated sensor pixel point. This results in a one colour-per-sensor pixel point being recorded as illustrated in Figure 2.7 using the Bayer filter for demostration. The sensor data, $\boldsymbol{I}_{CFA}$, is a sub-sampled representation of the original scene, $\boldsymbol{I}$, conveyed by the optical system. The sub-sampled representation, $\boldsymbol{I}_{CFA}$, is then processed by the camera software. In this phase, a colour interpolation step termed demosaicking is performed. As the name implies, the missing sensor data is approximated and the mosaic effect in $\boldsymbol{I}_{CFA}$ is removed. The result is the reconstructed image, $\boldsymbol{I}_R$.

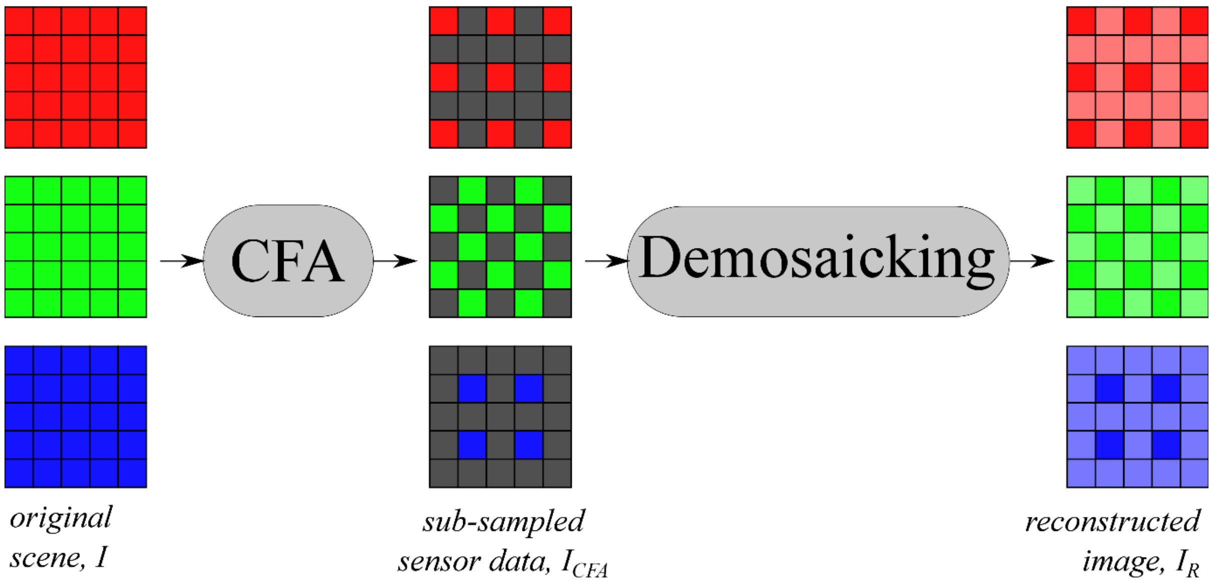

CFA
Demosaicking
original scene, I
sub-sampled sensor data, I_{CFA}
reconstructed image, I_R

Figure 2.7 The demosaicking process using the Bayer filter for demonstration

Mathematically, for an image of sensor pixel dimensions M×N, the two dimensional M×N matrix $\boldsymbol{I}_{\mathrm{CFA}}$ is a true subset of the three dimensional M×N×3 matrix $\boldsymbol{I}$. This is because three sensor will have three independent colour planes. The demosaicking process generates a new three dimensional matrix $\boldsymbol{I}_{\mathrm{R}}$ from $\boldsymbol{I}_{\mathrm{CFA}}$ that approximates $\boldsymbol{I}$ and their absolute difference is defined by a three dimensional M×N×3 matrix $\boldsymbol{E}$, as given in equation (2.2).

$$\boldsymbol{I}_{CFA} \in \boldsymbol{I} \tag{2.2a}$$

$$\boldsymbol{E} = |(\boldsymbol{I} - \boldsymbol{I}_R)| \; where \; \lim_{\boldsymbol{I} \to \boldsymbol{I}_R} \boldsymbol{E} = 0 \tag{2.2b}$$

The demosaicking process applies to all the aforementioned colour filters as given in equation (2.2). In the ideal demosaicking situation, $\boldsymbol{E} = 0$. This is not possible in practice because demosaicking is an interpolation process that results in approximations. The practical goal of the demosaicking process is to ensure the reconstructed image matrix, $\boldsymbol{I}_{\mathrm{R}}$, is as close as possible to the reference scene matrix, $\boldsymbol{I}$, for maximum information conveyance.

2.3.3 The Panchromatic CFA

Additive and subtractive CFAs are fully spectrally selective recording sub-sampled information at all pixel point locations. Panchromatic CFAs, from their definition, are partially selective. The white portions of the filter allow all colour components to pass through. This is illustrated in Figure 2.8.

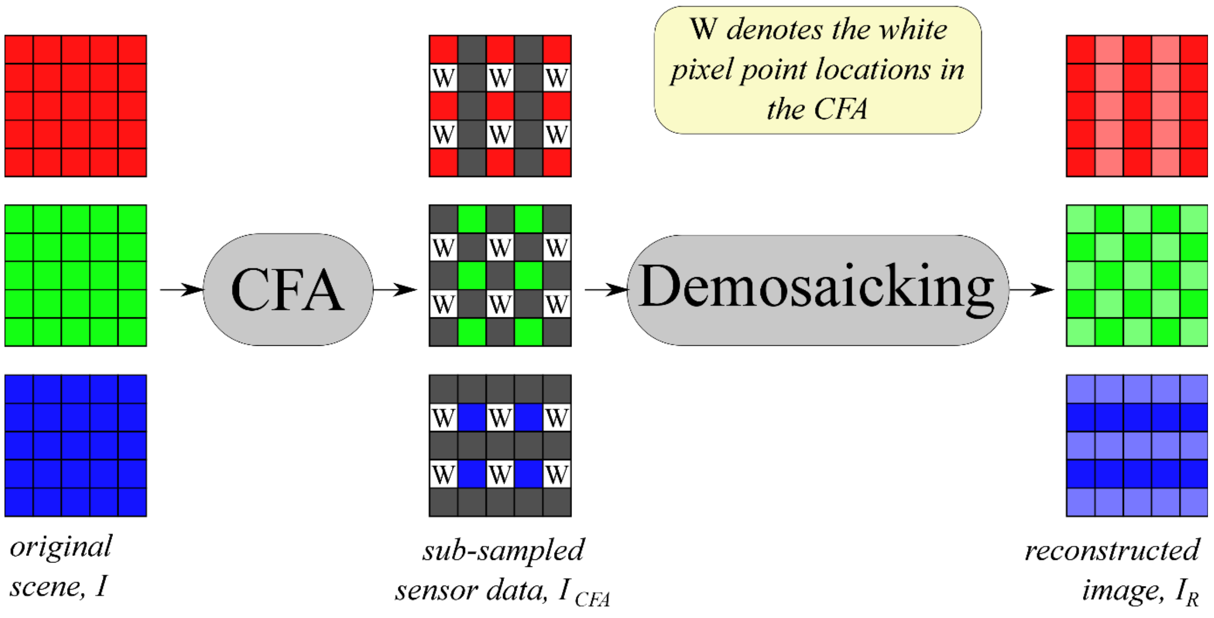

W denotes the white pixel point locations in the CFA
CFA
Demosaicking
W
original scene, I
sub-sampled sensor data, I CFA
reconstructed image, I R

Figure 2.8 The demosaicking process using the Gindele panchromatic filter for demostration

Comparing the panchromatic demosaicking process in Figure 2.8 with its non-panchromatic counterpart in Figure 2.7, more sample points are recorded in the red and blue colour channels but the green colour channel has the same sample point total.

As such the panchromatic sensor data, $\boldsymbol{I}_{\mathrm{CFA}}$, is an M×N×3 matrix that is still a subset of the image, $\boldsymbol{I}$. However,

$$\boldsymbol{I}_{CFA}(non\ panchromatic) \in \boldsymbol{I}_{CFA}\ (panchromatic) \in \boldsymbol{I} \tag{2.3}$$

Equation (2.3) shows the panchromatic sensor data carries more samples than its additive/subtractive equivalent. It should be noted that these extra samples take in the full visual colour spectrum instead of a single wavelength value dictated by the colour of the filter at these points.

This leads to a brighter image being taken at the expense of explicit colour accuracy. A white pixel point in the red lattice of Figure 2.8 collects not only the red filter colour wavelength, but also red content outside the filter specifications along with green and blue content. So the red content extracted from the white pixel point may not correspond exactly to one extracted if the white pixel point was replaced by a red pixel point.

In an ideal image panchromatic filter capture situation, the particular wavelength can be extracted from the full spectrum in processing, leading to a reconstructed image with less unknowns being used in the demosaicking process.

In practice, however, this is not wholly possible without introducing some approximations in the extraction. The process of panchromatic CFA demosaicking is aimed at ensuring reproduction at the white pixel points is superior to that of the blank cells points that rely solely on neighbourhood cell information for their reconstruction. Panchromatic CFAs are classified by the percentage of white pixels present in their base mosaic cell. In Table 2.2, two classes are represented:

i. 25% Type: Gindele/Sugiyama (also called the RGBW CFA)
ii. 50% Type: Dillon, Yamagami, Bawolek and Kodak variants

Other configurations with higher percentages exist such as the CFA proposed by Luo [52] which is a 75% type. They are not considered desirable in practice because more colour information is lost at the expense of a brighter, more luminous image.

2.3.4 Spectral Advantages of Panchromatic CFAs

The less restrictive filtering of panchromatic CFAs leads to several key advantages in image capture and recording:

i. Larger sample sizes in the red and blue planes ensure less uncertainty in their reconstruction.
ii. White pixel point locations record complete luminosity information leading to a brighter, more visually appealing image capture that is more representative of the scene. In the additive/subtractive CFAs, the most sampled colour is used to determine the luminosity content of the scene yielding darker images.

2.4 Visual Artefacts

Being a physical system, the digital still camera and the image processing pipeline it adopts are subject to the effects of noise and other unwanted aberrations.

The common artefacts experienced by integrated DSCs are given in Table 2.3.

Table 2.3 Visual artefacts and aberrations commonly generated in single sensor cameras

Type	**Area of Pipeline**	**Examples**
Optical Effects	Acquisition Phase	Chromatic aberrations, Comatic aberrations, Spherical aberrations, Lens flare and Vignetting effects
Image Noise	Acquisition Phase	Fixed pattern, Random, Banding
Demosaicking Artefacts	Processing Phase	Moiré effects, Zipper effects, Colour shifts, Blurring
Coloration and Exposure Shifts	Processing Phase	White imbalances and exposure effects
Compression Artefacts	Storage and Display Phase	Lossy JPEG format compression

2.4.1 *Optical Effects*

Optical effects are caused by lens and aperture arrangement. Improper lens design can lead to spherical and chromatic aberrations. *Spherical aberrations* shown in Appendix E: Figure E.1 are caused when light that passes through the lens is not focused into one focal point; instead having multiple convergence points occur, leading to a blurring halo or monocle artefact effect [53], [54]. *Chromatic aberration* occurs when the lens focuses different wavelengths of light refracted through along different focal points of the optical median line. This is the result of the lens's refractive indices having an inverse proportionality relationship with the different wavelengths of light [55], [56]. In the additive tristimulus theory of light the red, green and blue wavelengths would be focused at different points as shown in Figure 2.9. An example of this aberration is shown in Appendix E: Figure E.2.

Comatic aberrations occur when light from point sources enters the lens from an angle rather than parallel to the optical median line. This distorts the point sources to have a coma, tear-drop shape instead of an expected circular form. An example of comatic aberrations are given in Appendix E: Figure E.3.

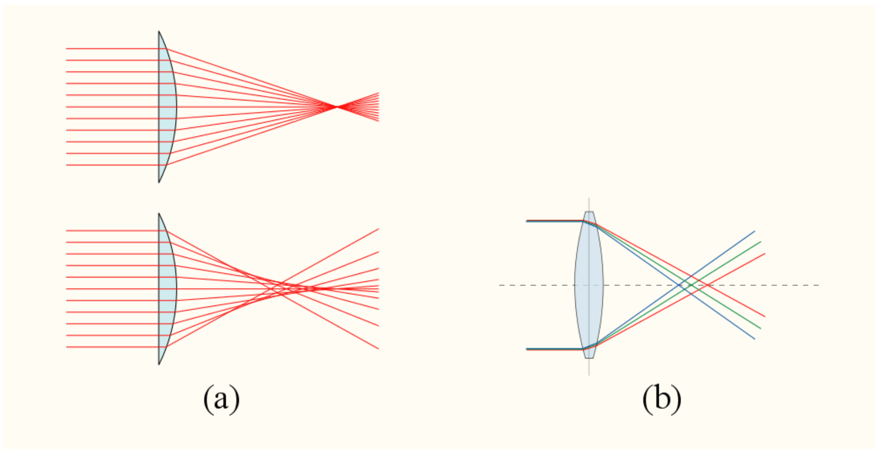


Figure 2.9 Generation of (a) Spherical aberration and (b) Chromatic aberration

Lens flare occurs when shooting a bright light source point. This source leads to unwanted internal reflection and scattering of light within the lens causing a bloom or flaring effect. In the image, this flaring is observed in the form of rings or bursts of light emanating from the source. *Vignetting* is the phenomenon where peripheral regions of the image appear darkened forming a border of sorts. The darkening is caused by brightness and saturation loss at the lens limits. This is sometimes performed intentionally in software but mainly occurs from limitations introduced at the lens periphery. An image showing this effect is presented in Appendix E: Figure E.5.

Lens flare, vignetting effects and spherical aberrations are unique because they are the only visual artefacts that are sometimes deemed desirable by the viewer. This is the case particularly where an emotional response from the viewer of the scene is desired along with the basic conveyance of visual information. An example is when lens flares were adopted in the rebooted Star Trek franchise films to heighten the drama and spectacle of space travel [57]. Some still images from the films are shown in Appendix E: Figure E.4.

2.4.2 Image Noise

This type of artefact is created in the sensor section and appears as speckles or bands in smooth regions of the image [33], [58]. Image noise is caused by random quantum noise effects and sensor element inhomogeneity. This form of noise is dependent on length of exposure, colour temperature variations, sensitivity settings and the physical characteristics of the sensor pixel photo-sites. There are several types of image noise. *Fixed pattern noise* is caused when the intensity recorded at a pixel point exceeds normal ambient noise values. They appear when the sensor is overexposed or made to operate at high thermal temperatures. The overexposed pixel point forms a bright colour spot dependent on the particular light filtered through.

Random noise is generated by intensity and colour fluctuations below and above the actual image values. This form of noise is always present in some degree and is mainly influenced by camera shutter speed. *Banding noise* is created when camera reads data from the sensor. The sequential nature of this process leads to the noise appearing as continuous bands. It is often seen in images taken in shadowy conditions. Sensor images showing fixed pattern, banding and random noise effects are shown in Appendix E: Figure E.6.

2.4.3 Demosaicking Artefacts

These errors are produced from the demosaicking process. Demosaicking involves the reconstruction of an image by approximating missing sensor data. When the approximation is underestimated, overestimated or completely off, undesirable visual artefacts are produced in the reconstructed image. Being camera invariant, these types of artefacts are commonly observed in practice. They are also the most documented in literature [59]–[63]. Demosaicking artefacts appear in four primary forms:

i. Moiré effect
ii. Zipper effect and ‘Jaggies’
iii. Colour shifts
iv. Blur effects

The *Moiré effect* is a spatial aliasing phenomenon that occurs when the scene contains a series of repetitive patterns that is of the same order or exceeds the resolution of the sensor. Consequently, the sensor is unable to adequately sample the high frequency information in the image scene, causing an aliasing problem. This aliasing presents itself in a grayscale or rainbow colour pattern being inserted into the image. Figure 2.10 illustrates how this artefact is generated in a Bayer CFA from a simple black and white stripe pattern. The pattern has two stripe types: Type 1 is of a similar order to the sensor pixel resolution while Type 2 has a lower resolution than the sensor. In Figure 2.10(a), the stripes are aligned to the sensor pixels and each pixel wholly records one type of colour. As such both stripe types are accurately reconstructed. In Figure 2.10(b), the alignment of the stripes is off. For the Type 1 stripe, each sensor records black stripe and white background colour information combining to form a grey stripe. In the case of the Type 2 stripe, only the edges suffer from this colour aliasing. This effect also occurs when patterns are of a higher resolution. Black and white images will produce grey patterns and colour images have rainbow-like patterns arising from this aliasing effect. An image showing this effect are given in Appendix E: Figure E.7.

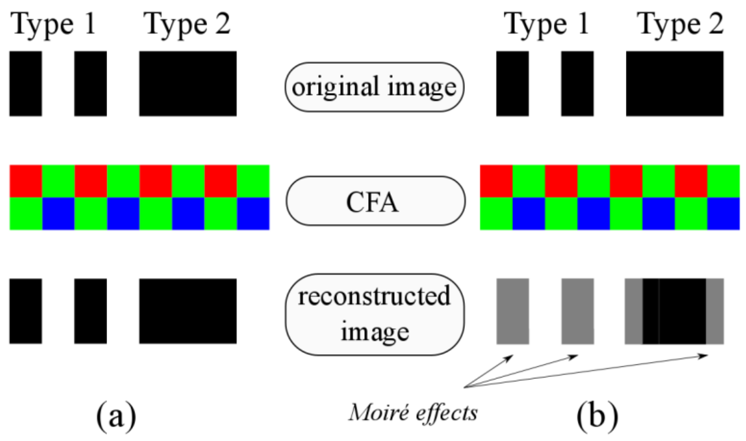


Figure 2.10 Generation of Moiré effects

Zipper effect occurs along edges of objects within a scene. The sudden transition is inaccurately approximated by the demosaicking process resulting in aliasing. This inaccurate approximation occurs on both sides of the edge producing a distinct zip-like effect seen in the reconstructed image [64]. If an object edge experiences only a slight deviation in their colour information from its adjacent surroundings, a staircase effect called ‘Jaggies’ is produced instead. Appendix E: Figure E.7 highlights some examples of this type of artefact.

Colour shifts occur within feature rich sections of an image when the demosaicking algorithm lacks adequate spectral information. Colour in these sections is misrepresented and a shift in the colour quality is observed in the reconstructed image. When image demosaicking is done with poor edge preservation, the structural integrity is lost and *image blurring* occurs. Similar to colour shifting, image blurring occurs in detail rich regions of the image. Image blurring and colour shifting examples are given in Appendix E: Figure E.7.

It should be noted that this class of artefacts vary in degree of severity depending on the scene data and the quality of the demosaicking algorithm. Considering the complexity of reconstructing unknown scene data, these artefacts cannot be removed completely but instead are only suppressed [65].

2.4.4 Coloration and Exposure Shifts

All cameras are designed and calibrated to operate within a particular sensitivity and colour light temperature setting. If an image is taken outside the prescribed camera light settings, the coloration of the scene in the image shifts from the original [33]. *Coloration shifts* are easily observable in achromatic white light because of the uniformity of the three additive colours. These shifts affect the image in its entirety and several modes of coloration shift are shown in Appendix E: Figure E.8 depending on what colour is in excess. White colour balance is the process performed to ensure the image is a true representation of the original scene.

The exposure of the sensor to the reflected light from objects within a scene also dictates the visual acuity of the image. The speed of the camera shutter determines the level of *exposure shift* experienced due to the amount of photons recorded. Overexposure and underexposure are the two shifts observed. Underexposure results in a darkened image while overexposure of the sensor results in an overly bright image. Both these variations are shown in Appendix E: Figure E.9.

2.4.5 Compression Artefacts

To reduce file size in storage, many integrated DSCs employ lossy compression. A common format of this type is the JPEG format. The JPEG compressed images are created by storing colour and intensity information for a region of similar pixels instead of recording original individual pixel data. This often reduces the size of the image file at the expense of pixel information. Consequently, edge and fine detail information is lost and JPEG images suffer from a blocky appearance. These block regions are the lossy compression artefacts. This class of artefacts is often viewed as more serious than image noise [66] because of the loss of important object information in the scene.

2.5 Demosaicking Algorithms

From Figures 2.7 and 2.8, it can be observed that demosaicking is primarily an interpolation process [67], [68]. However, unlike conventional interpolation, the colour content in the various lattices is highly sporadic, depending on the scene. In addition each of these colour lattices, while handled independently at the sensor, jointly contribute to generate a single colour at each pixel point. This leads to the interpolation being both an intra-dependent and inter-dependent lattice problem. The process must be uniform in both dependencies. With this in mind, demosaicking algorithms are classified, but are not limited to, four principle classes: traditional, heuristic, optimisation and image modelling.

2.5.1 Traditional methods

Algorithms in this class are used predominantly outside the field of demosaicking. An image's spatial pixel data in the three colour lattices is converted into a sequential polynomial form. Mathematical interpolation is then done. Bayer's initial design demosaicking algorithm was a simple linear interpolation [39] that was an extension of a design by Banning et al. [69]. This was due to the fact that the process involved a sequential scanning of the image line by line and the generation of the missing components followed the same mechanism. Subsequent algorithms have employed higher order fitting techniques to improve performance. Yu [70] proposed a cubic interpolation solution and Randhawa et al. [71] used splines.

2.5.2 Heuristic methods

This class of algorithms involve applying a filtering process either in the spatial or spectral domain while making several assumptions on the properties of the image. These algorithms seek to produce a sufficient approximation of the original scene in a reasonable time rather than attempt a full mathematical optimisation [68]. This is because a sufficient approximation and a fully optimised image would be indistinguishable to the human viewer. Heuristic algorithms are the most commonly documented demosaicking technique and are further divided into spatial, spectral and a hybridisation of the two.

(i) Spatial techniques

This sub-class operates by applying the filtering process directly on sensor pixel data in the spatial domain. *Non-adaptive techniques* such as Bi-linear (BI) and bi-cubic interpolation (BCI) are the simplest forms of spatial demosaicking. They are adequate in smooth image region. However, colour and edge abnormalities result in the detailed regions because the intra-lattice and inter-lattice dependencies is not considered.

Constant hue-based techniques assume that there is a strong inter-lattice correlation between different colour lattices. This enables luminance (intensity) information to be used to interpolate chrominance (colour) lattice data. Considering a Bayer filter, the green lattice is assumed to hold luminance data while the red and blue are chrominance lattices. The constant hue assumption uses pixel data in the green lattice to find missing data points in the blue and red chrominance lattices. An example of this is the constant difference based interpolation (CDBI) [68] illustrated in Figure 2.11. The more populous green lattice ***G***, is reconstructed to form ***G**$_r$*, by simple bilinear interpolation. The red lattice, ***R***, is then subtracted to create a difference lattice, ***D***. The difference lattice is then used to form the reconstructed red lattice, ***R**$_r$*, using equation (2.4) for a pixel point, *p*;

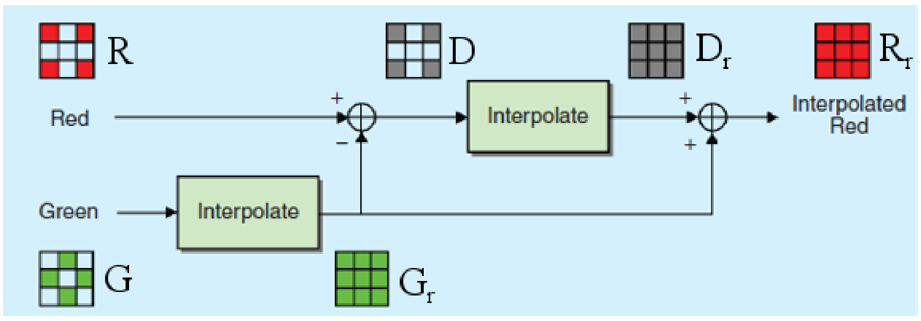


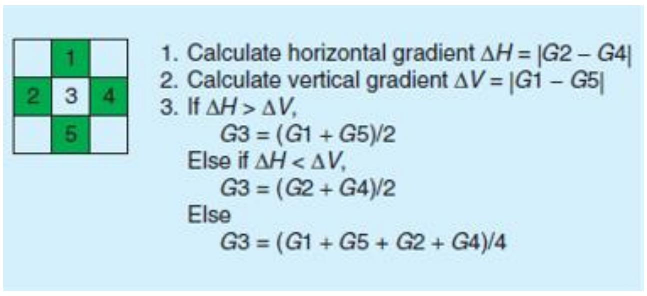
Figure 2.11 Constant difference-based interpolation in a Bayer CFA

$$\begin{aligned} D_{(p)} &= R_{(p)} - G_{r(p)} \\ R_{r(p)DIFF} &= D_{r(p)} + G_{r(p)} \end{aligned} \tag{2.4}$$

The similar constant ratio based interpolation (CRBI) [68] uses the relationship shown in equation (2.5) for any pixel point, *p*. Cok [72] proposed a logarithmic form of constant hue interpolation. Kimmel [73] and Chung et al. [74], [75] extended constant hue based interpolation by combining it with basic edge directed interpolation.

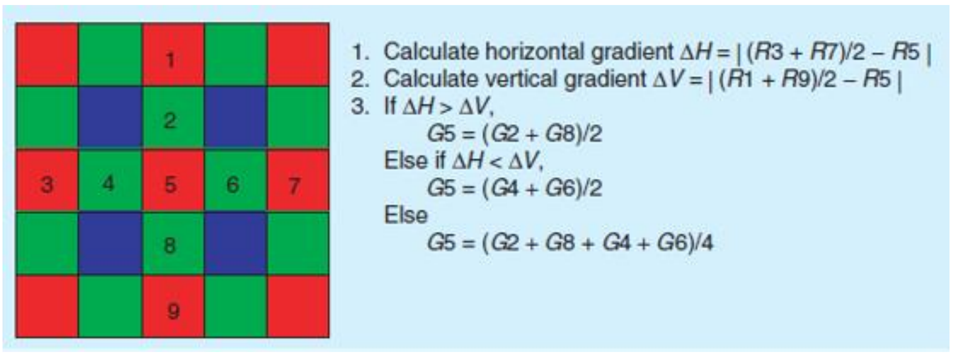
$$\begin{aligned} D_{(p)} &= R_{(p)} \div G_{r(p)} \\ R_{r(p)RATIO} &= D_{r(p)} \times G_{r(p)} \end{aligned} \tag{2.5}$$

Edge based adaptive techniques work on the principle of ensuring interpolation only occurs along object edges rather than across them. The determination of the direction of interpolation is established by querying neighbourhood pixel information. This is shown in Figure 2.12. Edge directed interpolation (EDI) and its variants [76]–[81] generate absolute difference measures in the cardinal directions; ΔH and ΔV. If one measure exceeds its complementary, the interpolation is done in the complementary direction. Otherwise, neighbourhood averaging is done. This demonstrates the lack of an edge in that particular pixel point. Figure 2.13 shows how edge directed demosaicking is extended when it incorporates inter-lattice correlation.



```
1. Calculate horizontal gradient $\Delta H = |G2 - G4|$
2. Calculate vertical gradient $\Delta V = |G1 - G5|$
3. If $\Delta H > \Delta V$,
        $G3 = (G1 + G5)/2$
   Else if $\Delta H < \Delta V$,
        $G3 = (G2 + G4)/2$
   Else
        $G3 = (G1 + G5 + G2 + G4)/4$
```

Figure 2.12 Sample edge directed interpolation using single lattice data



```
1. Calculate horizontal gradient $\Delta H = |(R3 + R7)/2 - R5|$
2. Calculate vertical gradient $\Delta V = |(R1 + R9)/2 - R5|$
3. If $\Delta H > \Delta V$,
        $G5 = (G2 + G8)/2$
   Else if $\Delta H < \Delta V$,
        $G5 = (G4 + G6)/2$
   Else
        $G5 = (G2 + G8 + G4 + G6)/4$
```

Figure 2.13 Sample edge directed interpolation using multiple lattice data

The performance of this demosaicking subset is wholly dependent on the difference measures generated to classify the presence or absence of an edge. The absolute difference measure is in fact a basic edge descriptor. Chang et al. [82] introduced finer edge descriptors leading to the development of *gradient based techniques*. Gradient based variants [83], [44], [84]–[91] make use of higher order absolute differences, termed gradients, that improve demosaicking. The gradient is inverted to act as an adaptive weighting factor because higher order gradients provide a more varied spread to fine tune demosaicking. Gradients can be improved further by working with residuals of gradients rather than absolute gradients. *Residual based techniques* were introduced in 2013-2016 by Kiku, Monno et al. [92]–[94].

Another way of further improving spatial demosaicking is introducing more interpolation directions. *Multidirectional weighted techniques* combine gradient or edge based methods with increased direction choice. Ordinal and oblique directions are used with the conventional cardinal directions. Some examples of this are presented in the algorithms found in [95]–[98].

(ii) Spectral techniques

This heuristic sub-class involves converting the sub-sampled spatial pixel information into the frequency domain and applying the filtering process in this domain. This form of demosaicking was first introduced by Alleysson et al. [99] and further developed by Dubois [100]. Frequency based demosaicking variants [101], [102] use simple low order filters to reconstruct the image. This leads to excessive smoothening of edge information. *Wavelet based techniques* proposed by Kolta et al. [103], Zhang et al. [104] and Komatsu et al. [105] use wavelet theory to sharpen the image during reconstruction. *Compressive sensing* has also been recently applied to the demosaicking problem by Gürbüz et al. [106] and Singh et al. [107]. Once the filtering is done, the image is restored to its spatial equivalent.

(iii) Spatio-spectral techniques

Spatio-spectral methods combine spatial and spectral filtering in their demosaicking process. In most instances, spectral filtering is performed followed by an edge or gradient based spatial technique to sharpen the image. The body of work by Hirakawa and Parks [46], [108], [109] highlights this class. Hirakawa et al. use wavelet filter banks to generate a unique edge demosaicking algorithm that searches for homogeneous regions to avoid edge misrepresentation.

2.5.3 Optimisation methods

This class of demosaicking algorithms treats the interpolation as a mathematical optimisation problem [67], [68] where colour correlations and other image properties are cost functions that can be iteratively minimised.

(i) Regularisation

The regularisation demosaicking sub-class minimises a cost function consisting of two primary terms: a colour correlation term and a spatial smoothness term. For the classical Bayer CFA, to write the cost function, a vicinity vector $\boldsymbol{v}$, is defined as given in equation 2.6.

$$\boldsymbol{v}(n_1,n_2) = \begin{bmatrix} R(n_1,n_2) - \bar{R} \\ G(n_1,n_2) - \bar{G} \\ B(n_1,n_2) - \bar{B} \end{bmatrix} \tag{2.6}$$

Where $\bar{R}, \bar{G}$ and $\bar{B}$ are colour averages in the vicinity of the pixel located at point(n_1, n_2). In addition, defining a colour covariance matrix $\boldsymbol{C}_{n1n2}$ and three directional derivatives: $S_{n1,n1}$, S_{n2n2} and $S_{n1,n2}$; the cost function X, is defined as:

$$\begin{aligned} X = & \iint \sum_{S=R,G,B} \left(S_{n1,n1}{}^2 + \ 2S_{n1,n2}{}^2 + S_{n2,n2}{}^2 \right) dn_1 dn_2 \\ & + \alpha \iint \boldsymbol{v}(n_1,n_2)^T \ \boldsymbol{C}_{n1n2}{}^{-1} \boldsymbol{v}(n_1,n_2) dn_1 dn_2 \end{aligned} \tag{2.7}$$

where α is a small positive constant. The minimisation process starts with a rough interpolation that is progressively refined. Examples of this method are found in literature [110]–[114]. Variations exist in the description of the covariance matrix and choice of directional derivatives.

(ii) Vector-based filtering

Vector based filtering demosaicking involves visualising each pixel as a vector of three or more colour values. The demosaicking algorithm shown by Yuk [115] and Lukac et al. [116], [117] then aims to fill in missing sample data such that the distance between vectors of neighbouring pixels are as small as possible. This process is repeated while updating vector distance values until an optimal solution is obtained.

(iii) Bayesian estimation and Projection onto Convex Sets

Bayesian estimation [68] works using probability theory to reconstruct the image. The recorded colour and noise statistics are modelled as probability distribution functions and a maximum a posteriori (MAP) formulation is performed to reconstruct an optimised estimate of the image. Projection onto convex sets is a demosaicking technique that involves using constrained set theory to map out the optimisation path [118].

2.5.4 Image Modelling and Training

This class involves exposing the algorithm to a large set of predefined pixel regions [119], [120]. The algorithm then uses a set of rules to compare sections of an image with the predefined regions. Once a match is found, a second set of rules determining the actual demosaicking process are then employed. Due to the large database that would be required to adequately define each possible scenario in an unknown image, this class of algorithms is hampered by long run times.

2.6 Knowledge Gaps

From the literature review, it was found that most demosaicking publications centred on the Bayer CFA. A simple search analysis was done to compare the occurrence of the Bayer CFA and panchromatic CFA in published literature.

Table 2.4 Search term statistics for the words 'bayer cfa' and 'panchromatic cfa'

Search Term	**IEEE Xplore Library**	**Springer Link Repository**	**SPIE Digital Library**
'bayer cfa'	139	607	881
'panchromatic cfa'	10	39	259*

** The term 'panchromatic cfa' in the SPIE repository contained references to analogue microfilm, telescope and laser technology that is not part of this study. To compensate, the terms 'RGBW cfa' and 'White-RGB cfa' were queried instead.*

Bayer noted that initial panchromatic devices performed poorly due to sensor design limitations [39] when comparing his method to panchromatic CFAs at the time. Panchromatic sensors such as the Gindele et al. [50] and Kodak panchromatic types [121] have overcome these limitations through improvements in CMOS production and miniaturisation.

Recent publications [23], [121], [122] have empirically proved that panchromatic CFAs have a better sensitivity characteristic especially in low light situations. Little work has been done developing robust algorithms to work with them. Most published demosaicking algorithms work with the older Bayer sensor developed in 1975 [39]. The work presented here seeks to make a contribution in the field of panchromatic CFA demosaicking design.

From the literature review it was also found that all the visual artefact studies were done using Bayer CFAs. This work also studies how artefacts present themselves in panchromatic CFAs and whether the proposed algorithm mitigates them sufficiently. The behaviour of demosaicking artefacts is considered solely due to their device invariance.

Finally most published work, to assume uniformity in analysis, adopts use of the classic Kodak image set [123]. Other image sets referenced are the McMaster-IMAX image set [124] and the Condat image set [125] but to a lesser degree. However, all these image sets were of a low resolution (under 1000×1000 pixels in dimension). To mimic images taken by real modern integrated DSCs, a custom image set database was created for medium resolution images. In addition, a high image database was also selected for supplementary analysis. These custom image sets are used in tandem with the established standard sets to allow the algorithm to be exposed to many resolution possibilities.

3 CONCEPTUAL FRAMEWORK

From the knowledge gaps highlighted in the literature review in the previous chapter, the following deficiencies were observed:

i. There is a tendency in literature to primarily focus on the traditional Bayer CFA rather than the panchromatic CFA class despite empirically proven superiority observed in the latter.

ii. Objective analysis of algorithm performance focuses on using at most two image sets: Kodak and McMaster-IMAX. Attributes inherently present in these image sets (such as type of scene content, light saturation levels, camera resolution among others) are not considered to play a role. This is not the case because an algorithm may exhibit better performance over another due to an image attribute, say type of scene content. To fully assess an algorithm, a larger and a more variable image test bed profile is desirable along with some understanding of the image attributes that are present.

iii. Demosaicking artefacts are primarily analysed using the traditional image assessment metrics such as the mean square error (MSE) and colour peak signal-to-noise ratio (CPSNR). These measures, while commonly referenced, are not suitable for analysing demosaicking artefacts such as Moiré and zipper effects [126], [127].

To ensure these deficiencies are adequately addressed in line with the main objective of the research work, the conceptual framework presented in Figure 3.1 was developed.

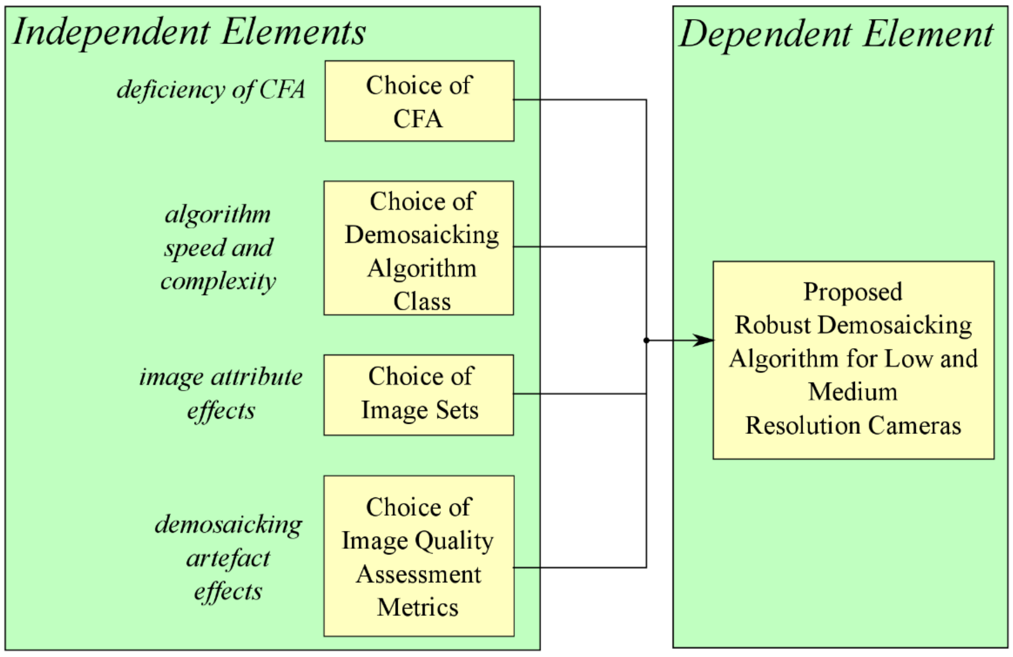

Independent Elements
deficiency of CFA
Choice of CFA
algorithm speed and complexity
Choice of Demosaicking Algorithm Class
image attribute effects
Choice of Image Sets
demosaicking artefact effects
Choice of Image Quality Assessment Metrics
Dependent Element
Proposed Robust Demosaicking Algorithm for Low and Medium Resolution Cameras

Figure 3.1 Conceptual framework to formulate proposed demosaicking algorithm

From the conceptual framework, four independent design choices can be considered to create and ensure the robustness of the proposed demosaicking algorithm: the colour filter array, the algorithm class, the image sets forming the test bed and the assessment metrics. Each design choice helps mitigate a deficiency or undesirable effect extrapolated from the knowledge gaps.

3.1 Colour Filter Array Selection

The proposed algorithm was designed to work in the panchromatic class of colour filter arrays. From the literature review, there are several documented panchromatic colour filter arrays that have been developed. In this study, the RGBW CFA proposed by Gindele and Sugiyama [50], [51] was selected for the reasons presented below:

i. ***The $\mathbf{2 \times 2}$ nature of the mosaic cell***: A simpler mosaic cell arrangement allows for the use of a wider class of demosaicking methods while simultaneously ensuring fewer demosaicking inaccuracies due to its simpler design. This has led many manufacturers to design cameras that use the smallest 2×2 mosaic cell arrangement. From Table 2.2, the CFAs meeting this criteria are the Bayer, Hamilton, Bean, Bawolek and RGBW (Gindele/Sugiyama) CFAs.

ii. ***The presence of a white pixel point:*** The additive and subtractive class of CFAs record individual light colour or ***chroma*** data at the expense of overall light intensity or ***luma*** data. From Appendix D, it is shown that overall light intensity is not a pure addition of the individual light intensities sampled. A true representation of the scene requires a recording both colour and intensity. This is provided by the panchromatic class of CFAs. Of those provided in Table 2.2, only the Bawolek and RGBW CFAs possess a white pixel point to sample intensity (luma) data and a 2×2 mosaic cell profile.

iii. ***A balanced luma-chroma distribution***: The recorded chroma and luma content should be done in such a manner as to ensure sufficient sampling of both parameters. From equation 2.1, it is noted that the Bawolek CFA only records primarily blue and green content devoting half its mosaic cell for light intensity. However, the RGBW CFA records red, blue and green colours leaving one quarter of the mosaic cell to capture light intensity. Comparing the two, the RGBW CFA luma-chroma distribution is closer to the human visual system of three colour cone photoreceptor types and one rod photoreceptor type.

From all the CFAs shown in Table 2.2, the RGBW CFA was selected using the above criteria. Working in the RGBW mosaic cell domain allows the findings presented herein to have wide application primarily in the target category of cameras with low to medium resolution.

3.2 Choice of Demosaicking Algorithm Class

The four classes of demosaicking algorithms presented in the literature review are the traditional, heuristic, optimisation-based and image modelling/training classes. Table 3.1 presents these aforementioned classes with the following associated properties:

i. **Speed**: an indicator of the ease of implementation and time taken to undergo the image demosaicking process, determined from real time testing.
ii. **Local adaptability**: an indicator to illustrate the ability of the algorithm class to compensate for and properly reconstruct localised areas in an image with large amounts edge variations. It is an indirect measure of algorithm complexity.
iii. **Image reconstruction acuity**: an indicator highlighting how accurately the reconstructed image represents the original scene.
iv. **Popularity**: an indicator showing how often the demosaicking class is researched and referenced in peer-reviewed literature.

Table 3.1 Some properties of the various demosaicking algorithm classes

Demosaicking Algorithm Class	**Speed**	**Local Adaptability**	**Image Reconstruction Acuity**	**Popularity**
Traditional	Very High	Non-adaptive	Low to Medium	Low
Heuristic	High	Non-adaptive and Adaptive	Medium to High	Very High
Optimisation-Based	Low	Adaptive	Very High	Medium
Image Modelling and Training	Low to High	Adaptive	High to Very High	Medium to High

Using the above metrics, the heuristic demosaicking class of algorithms was chosen as the basis of the proposed algorithm design. It has a high speed of processing that is desirable for a low-to-mid resolution camera integrated in a mobile device as it ensures a low computational load on the mobile device processor. The popularity of the heuristic class also ensures there is a large set of established methods that can be used for comparison to the proposed design.

Heuristic algorithms are further sub-divided into spatial and spectral techniques. The proposed design was chosen to work in the spatial technique domain because they work directly on pixel data without the need for a transformation step noted in spectral techniques. This is an attractive property that should be actively exploited in integrated cameras. Of the spatial techniques, the modern gradient-based spatial-heuristic subclass was considered due its local adaptability feature making it more resistant to demosaicking artefacts.

Consequently, the proposed algorithm was chosen to be of the gradient-based spatial-heuristic class.

3.3 Image Sets Selection

To ensure a variable and robust test bed for the demosaicking algorithm, six different image sets described in Table 3.2 were selected. Each image set possesses a unique property desirable in analysis. The images are provided in Appendix C. Using the camera resolution chart provided for reference in Appendix C, the author designated the following dimensions to determine resolution:

i. **Low**: images of 1024 × 768 or smaller.
ii. **Medium**: images between 1280 × 960 (1 MP) and 2048 × 1536 (3MP).
iii. **High**: images of 2240 × 1680 (4MP) or higher.

Table 3.2 Selected Image Sets

Image Set	No. of Images	Image Dimensions	Resolution	Reason for Use
USC-SIPI (Classical) [128]	16	256 × 256	Low	Image Popularity
Kodak [123]	24	768 × 512	Low	Standard Reference
McMaster-IMAX [124]	18	500 × 500	Low	Analysis of Oversaturation
Condat Subset* [125]	30	720 × 540	Low	New Object Types in Scene and Light Variability
ARRI [129], [130]	12	2880 × 1620	High	
Custom	15	1918 × 1077	Medium	Algorithm Robustness
Total	***115***			

** The full Condat image set consists of 150 images. A subset of 30 randomly picked images was defined and used*

The USC-SIPI image set shown in Figure C.2 in Appendix C contains the classical set of analogue film images widely used and referenced in signal processing literature. Some examples include the Mandrill image (*sipi_im11*) and Lena image (*sipi_im12*). The use of this set allows ease of comparison and the results can be compared to a wider range of reported works, in particular those outside the area of demosaicking but still within the field of image enhancement.

The Kodak image set given in Figure C.3 contains digital images most commonly documented in demosaicking surveys. The Kodak set is attractive because of its set variability with images of persons, landscapes and objects. As it is the most popular image set used in demosaicking research, it is used in this work to ensure comparability with other demosaicking techniques.

The McMaster-IMAX digital image set shown in Figure C.4 is another popular demosaicking image set that introduces the effect of colour vibrancy in demosaicking analysis. Images in the set are oversaturated; which is an effect of some medium and many high resolution cameras. Comparing the proposed algorithm to established techniques using this set provides an analysis of the proposed method in such a camera scenario.

The Condat and ARRI digital image sets are used to test the robustness of the proposed algorithm in situations that are not catered for by previous image sets such as multiple persons in an image, man-made structures, abstract forms and light variations in a single image. These image sets are illustrated in Figures C.5 and C.6 in Appendix C. The Condat set is a low resolution database while the ARRI set is a high resolution database.

Finally, the author felt there was the need to analyse performance of the algorithm in medium resolution cameras. Low resolution images tend to be more blurred and colour muted when compared to higher resolution devices. Many integrated cameras fall in this category; however, the established image sets have been taken by older, low resolution standalone digital cameras. A custom image set presented in Figure C.7 is used to analysis performance in the medium resolution domain.

3.4 Image Quality Assessment Metrics Employed

Image quality assessment metrics fall in two categories: reference and no-reference (blind) assessment [127], [131]. Reference assessment requires a ground truth image. This ground truth is compared to the reconstructed image in the assessment to form an opinion on the acuity of the demosaicking process. Blind assessment does not require a ground truth image but uses luma and chroma information in the reconstructed image to assess whether the demosaicking process was performed with sufficient accuracy. In demosaicking algorithm design, literature is biased towards use of a reference assessment mechanism. Reference assessment methods are also faster than blind assessment. The use of a ground truth image also allows for a subjective comparison involving human viewers [132].

The common reference-based image quality assessment methods are the mean square error (MSE) and peak signal-to-noise ratio (PSNR). The MSE as shown in equation (3.1) for a monochromatic image,

$$MSE = \frac{1}{rc}\left(\sum_{i=1}^{r}\sum_{j=1}^{c}[I(i,j) - I_R(i,j)]^2\right) \tag{3.1}$$

where r is the total number of rows in the image, c is the total number of columns in the image, I is the ground truth original image and I_R is the reconstructed image from the demosaicking process. When reconstruction is perfect, the MSE is zero. The PSNR is mathematically defined as:

$$PSNR = 10\ log_{10}\left(\frac{Max^2}{MSE}\right) \tag{3.2}$$

where Max represents the largest pixel value possible in the image. In this work 8-bit images are used, hence the Max value is 255. If the demosaicking is ideal, the PSNR would have an infinite value.

A common variant of PSNR considers the averaging of the three colour channels: red, green and blue and is called the colour peak signal-to-noise ratio (CPSNR). This is because a colour image can be considered as a combination of three monochromatic images in the three colour channels; as outlined in Chapter 2. The averaging of the three colour channels is shown in equation (3.3) along with the resulting mathematical definition of CPSNR

$$MSE_{CPSNR} = \frac{1}{3rc}\left(\sum_{i=1}^{r}\sum_{j=1}^{c}\sum_{k=1}^{3}[I(i,j,k) - I_R(i,j,k)]^2\right)$$
$$CPSNR = 10\, log_{10}\left(\frac{Max^2}{MSE_{CPSNR}}\right) \qquad (3.3)$$

The MSE, PSNR and CPSNR reference assessment techniques work by performing a pixel-by-pixel comparison as a test for reconstruction acuity. Two modern reference assessment techniques are the Structural Similarity Index (SSIM), the Feature Similarity Index and its chrominance inclusive variant (FSIM/FSIM$_C$). These were developed based on the fact that the human visual system considers objects in a scene rather than absolute pixel values. Wang and Bovik [126] have shown that if a translational shift occurs in the pixel profile of the reconstructed image, then the MSE and PSNR metrics break down. The SSIM metric provides a measure more in line with subjective human evaluation by indicating the strength of reconstruction of whole objects in an image scene. Mathematically, the SSIM is based on the determination of three similarity terms: a ***structural*** term (SS), a ***luminance*** term (SL) and a ***contrast*** term (SC). This is shown in equation (3.4)

$$SS(x,y) = \left(\frac{\sigma_{xy} + k_1}{\sigma_x\sigma_y + k_1}\right)$$
$$SL(x,y) = \left(\frac{2\mu_x\mu_y + k_2}{\mu_x^2 + \mu_y^2 + k_2}\right) \qquad (3.4a)$$
$$SC(x,y) = \left(\frac{2\sigma_x\sigma_y + k_3}{\sigma_x^2 + \sigma_y^2 + k_3}\right)$$

$$SSIM = [SS(x,y)]^{\alpha} \cdot [SL(x,y)]^{\beta} \cdot [SC(x,y)]^{\gamma} \qquad (3.4b)$$

where μ and σ are ***mean*** and ***variance*** values of image x and y. The term σ_{xy} is the ***covariance*** between images. The terms k_1, k_2 and k_3 are small non-zero constants to prevent indeterminate results when the means and variances are close to zero. The exponent terms α, β and γ are used to provide intercomponent weighting of the three SSIM metric components and are usually all set to 1. As the SSIM metric works on monochromatic images and this work uses colour images; the SSIM is calculated as the average of the SSIM in each of the colour planes. This is shown in equation (3.5) as:

$$SSIM = \frac{(SSIM_{Red} + SSIM_{Green} + SSIM_{Blue})}{3} \tag{3.5}$$

The FSIM measure extends SSIM concepts by considering low level sections of objects rather than entire structures. The FSIM metric is composed of two similarity terms: a ***gradient magnitude*** (G) term and a ***phase congruency*** (P) term that are both variants of the luminance term of equation (3.4).

Initially gradient magnitude and phase congruency maps are determined in turn for two images, say x and y respectively. An individual pixel located at point (i, j) in these two maps over the two images will have four values that can be denoted as pc_x, gm_x, pc_y and gm_y. The similarity terms G and P are determined for each pixel point using equation (3.6a) that resembles the form of the luminance term in equation (3.4).

The overall FSIM is determined by adding all the pixel point similarities. This is presented in equation (3.6b) as a summation over the entire image region denoted as Ω.

$$\begin{gathered} G_{ij}(x,y) = \left(\frac{2gm_x gm_y + k_1}{gm_x^2 + gm_y^2 + k_1}\right) \\ P_{ij}(x,y) = \left(\frac{2pc_x pc_y + k_2}{pc_x^2 + pc_y^2 + k_2}\right) \\ SL_{ij}(x,y) = \left[G_{ij}(x,y)\right]^{\alpha} \cdot \left[P_{ij}(x,y)\right]^{\beta} \end{gathered} \tag{3.6a}$$

$$FSIM = \left\{ \sum\nolimits_{\Omega} \left(SL_{ij}(x,y) \cdot pc_{max}\right) \Big/ \sum\nolimits_{\Omega} pc_{max} \right\} \tag{3.6b}$$

The k terms are small non-zero terms to avoid indeterminate results. The exponential terms α and β are to weight similarity terms and pc_{max} is the maximum value when comparing pc_x and pc_y numerically.

The main drawback of SSIM and FSIM is that they were primarily designed for application with grayscale images and do not yield information on colour quality. This is addressed in the FSIM$_C$ variant. In this metric, the images are converted from the RGB (Red Green Blue) colour space to the YIQ (Luminance In-phase Quadrature) colour space where Y matrix holds the luminance information and I and Q matrices hold chrominance information. Consequently, the FSIM$_C$ has three components: a ***luminance similarity*** term (SY), ***in-phase similarity*** term (SI) and a ***quadrature similarity*** term (SQ).

The Y (luminance) matrix has its gradient magnitude and phase congruency maps determined and its similarity takes the form given in equation (3.6a). The pixel points in the chrominance I and Q matrices are denoted as pi and pq respectively as FSIM$_C$ is calculated using pixel points. For any two images x and y, the FSIM$_C$ is then given as shown in equation (3.7) where the k terms are small non-zero terms for error prevention. The terms $\alpha,\ \beta$ and γ are exponential terms to provide weighting of the similarity terms and pc_{max} refers to the maximum pixel point phase congruency value over the two images.

$$\begin{aligned} SY_{ij}(x,y) &= \left[G_{ij}(x,y)\right]^{\alpha} \cdot \left[P_{ij}(x,y)\right]^{\beta} \\ SI_{ij} &= \left(\frac{2pi_x pi_y + k_1}{pi_x^2 + pi_y^2 + k_1}\right) \\ SQ_{ij} &= \left(\frac{2pq_x pq_y + k_2}{pq_x^2 + pq_y^2 + k_2}\right) \end{aligned} \tag{3.7a}$$

$$FSIMc = \left\{ \sum_{\Omega}\left(SY_{ij}(x,y) \cdot \left[SI_{ij}(x,y) SQ_{ij}(x,y)\right]^{\gamma} \cdot pc_{max}\right) \Big/ \sum_{\Omega} pc_{max} \right\} \tag{3.7b}$$

To assess the proposed algorithm along with current and popular established demosaicking methods, the author selected MSE, CPSNR, SSIM and FSIM$_C$ as the image quality metrics. These metrics, along with their reasons for usage, are presented in Table 3.3.

Table 3.3 Selected Image Quality Assessment Metrics

Assessment Metric	**Units**	**Range (Min. to Max.)**	**Improvement**	**Operation**	**Reason for Use**
Mean Square Error (MSE)	-	0 to ∞	Low Value	Pixel Statistics	To analyse fidelity of ***reconstruction in a single plane***
Colour Peak Signal-to-Noise Ratio (CPSNR)	dB	0 to ∞	High Value	Pixel Statistics	To analyse fidelity of ***colour reconstruction in whole image***
Structural Similarity Index (SSIM)	-	0 to 1	High Value	Large Objects	To analyse fidelity of ***coarse object reconstruction***
Feature Similarity Index with chrominance included (FSIM$_C$)	-	0 to 1	High Value	Finer Detail Objects	To analyse fidelity of ***fine object reconstruction*** with ***colour consideration***

3.5 Proposed Algorithm Parameters

To fulfil the main design objective, the conceptual framework yielded the following design parameters for the proposed algorithm:

i. it belongs to the gradient-based spatial-heuristic class of algorithms
ii. it employs the RGBW panchromatic CFA
iii. it can be exposed to different image sets with varying image attributes
iv. it can be analysed using the MSE, CPSNR, SSIM and $FSIM_C$ image quality assessment metrics

4 ALGORITHM DESIGN

The development of the proposed demosaicking algorithm is presented in this chapter. From Chapter 3, the proposed algorithm was selected to be of the gradient-based spatial-heuristic type and to operate on panchromatic CFAs. This chapter outlines the design of this algorithm, taking into account white pixel processing that is a feature of panchromatic CFAs. The chapter also presents interpolation path determination that is a feature of gradient based algorithms. In addition, several novel concepts and contributions were proposed and incorporated into the designed algorithm. This ensures the creation of a robust algorithm.

4.1 Bayer and RGBW Design Comparison

The proposed algorithm is designed to interpolate missing colour data in a recorded CFA image using neighbourhood information. In Bayer demosaicking, the CFA imposes a homogeneous one colour-per-pixel regime. This is illustrated in Figure 4.1 for a 5×5 filter grid and it is noted that in each pixel location only one colour is recorded.

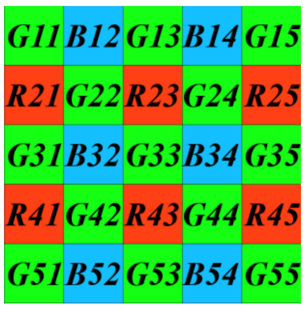


Figure 4.1 A 5×5 Bayer grid

The RGBW CFA, shown in Figure 4.2, by virtue of its white pixel points has a heterogeneous pixel arrangement. The red, green and blue pixel points all record a single colour, moreover this single colour occurs at a specific wavelength dictated by the filter itself. Conversely, the white pixel points allow the full spectrum of visible light to pass through. From the description of panchromatic sensors, in any white pixel point, say *W11*, the sensor does not only record the discrete wavelengths of the red, green and blue colour filters; it also records red, green and blue wavelengths outside the colour filter values.

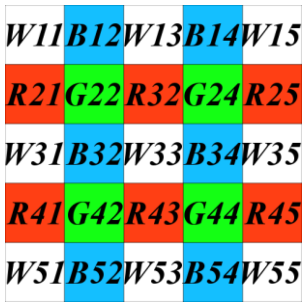


Figure 4.2 A 5×5 RGBW grid

This panchromatic concept can be mathematically expressed as given in equation (4.1):

$$W = R_{fil} + G_{fil} + B_{fil} + \left\{ \sum_{k \in \Omega} (R_k + G_k + B_k + \varepsilon_k) \right\} \tag{4.1}$$

Where R_{fil}, G_{fil} and B_{fil} are the specific wavelengths of the red, green and blue colour filters respectively; R_k, G_k and B_k are the red, green and blue wavelengths outside the filter specifications and ε_k denotes any other colour wavelength for the entire visible spectrum denoted by Ω.

4.2 White Pixel Processing

Due to the heterogeneous nature of the panchromatic class of colour filter arrays, in any panchromatic demosaicking process special attention must be paid to handling the recorded data from the white pixel point. This additional step, unique to panchromatic CFA regimes, must be considered before applying a demosaicking algorithm.

4.2.1 The Separation Process Technique

In the few heuristic panchromatic methods cited in literature [23], [133], [134] the white (W) pixel is usually handled in a separately from demosaicking. These methods, predominantly spectral-heuristic, consider the white pixel points as a purely light intensity (luma) component. From this treatment, the white pixel is observed to offer no contribution to the colour (chroma) component and is ignored during demosaicking. Consequently, the demosaicking process becomes wholly chroma-driven working in the non-white colour planes: red, green and blue. The additional luma information is incorporated in a latter process. The author has termed this reduction technique a ***Separation process***.

In the spatial-heuristic domain, the separation process is handled by replacing the white pixel points with green equivalents through a neighbourhood averaging process [23]. This is done using the immediate green pixel point neighbours of every white pixel point. Considering pixel $W33$ in Figure 4.2, the equivalent $\hat{G}_{W33}$ is generated using equation (4.2)

$$\hat{G}_{W33} = 0.25(G22 + G24 + G42 + G44) \tag{4.2}$$

The averaging process can be expressed as a filter, h_{basic}, shown in equation (4.3):

$$h_{basic} = \begin{bmatrix} 2 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 2 \end{bmatrix} /8 \tag{4.3}$$

All the required green equivalent pixels are generated in this manner. Green pixels on the image border are either discarded or approximated. The RGBW CFA representation of the image is consequently reduced to a Bayer CFA equivalent of the form illustrated in Figure 4.1. Demosaicking is then performed in the reduced Bayer equivalent CFA representation. The light intensity information of the white pixels is then handled separately and added to the image after demosaicking.

4.2.2 The Bayerisation Process Technique

A newer alternative method of handling the white pixel points was proposed by Chen et al. [135] based on the Malvar-He-Cutler algorithm [79]. The RGBW CFA representation of the image is exposed to the following modified averaging filter, h_{alt}, given in equation (4.4):

$$h_{alt} = \begin{bmatrix} 0 & 0 & -3/2 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ -3/2 & 0 & 6 & 0 & -3/2 \\ 0 & 2 & 0 & 2 & 0 \\ 0 & 0 & -3/2 & 0 & 0 \end{bmatrix} /8 \tag{4.4}$$

By using this filter, the RGBW CFA data is converted to an equivalent Bayer representation while simultaneously encoding the light intensity information within. All the RGBW information is encoded in the reduced Bayer equivalent. The author has termed this unnamed combined conversion and encoding technique a ***Bayerisation process***.

Analysing the effect of the Bayerisation process, it is noted that the white pixel coefficients of h_{alt} introduce a light intensity term into equation (4.2). If pixel $W33$ in Figure 4.2 is significantly brighter or darker than its white pixel neighbours, this intensity is additively factored into the green pixel average. If all the white pixels in the filter region are of equal value, h_{alt} reduces to h_{basic}. This light intensity term is denoted as δ and is given in equation (4.5).

$$\hat{G}_{W33} = 0.25(G22 + G24 + G42 + G44) + \delta \tag{4.5}$$

The Bayerisation process is attractive as it combines colour demosaicking and light intensity encoding in a single step thus reducing the overall algorithm complexity. This is useful when the algorithm is to run in an integrated camera on a mobile device. It is also a valid assumption to consider that the light intensity term, δ, is often significantly smaller than the green neighbourhood estimates in practice for the white pixels to adversely affect the green plane approximation. As such, the proposed algorithm employs the Bayerisation process to handle the white pixel points in the RGBW CFA data.

4.3 Gradient Based Demosaicking

Any spatial-heuristic demosaicking algorithm works on the principle of finding sufficient heuristic values that can be used to fill in the missing information in the raw CFA image. Rough initial estimates are derived through an analysis of neighbourhood information. Interpolation descriptors are then defined and used as interpolation weights to refine the initial estimates and generate the desired heuristic values. An accurate generation of interpolation descriptors is directly related to the efficacy of the algorithm.

In the gradient-based class of spatial-heuristic demosaicking algorithms, the descriptors are directional gradients obtained from a sum-of-difference (SOD) calculation mechanism during interpolation. Gradient-based demosaicking process can be generically defined in four steps expressed from equations (4.6) to equation (4.9).

Consider a general pixel, D_P; where D is the desired colour plane, N is a separate neighbouring colour plane, P is the pixel point location and k is the interpolation direction.

Step 1: The initial rough estimates of the desired pixel colour are established, as follows:

$$\widetilde{D}_P^k = \left(D_{P-1}^k\right) + c_1(N_P^k - N_{P-2}^k) \tag{4.6}$$

Where c_1 is a fractional constant to minimise neighbour effects.

Step 2: Interpolation descriptors in the form of directional gradients are derived based on a sum-of-differences calculation,

$$\Phi_P^k = \Phi_{Red,P}^k + \Phi_{Green,P}^k + \Phi_{Blue,P}^k + c_2 \tag{4.7}$$

Where Φ indicates a sum-of-differences and c_2 is a small, positive value corrective constant.

Step 3: The directional gradients from equation (4.7) are then converted into weighting factors as shown in equation (4.8),

$$\varphi_P^k = \frac{1}{\Phi_P^k} \tag{4.8}$$

Step 4: Finally, the missing colour component in the general pixel D_P is found from the calculated estimates and the associated weighting factors using equation (4.9),

$$\widehat{D}_P = {\sum_k \left(\varphi_P^k \widetilde{D}_P^k\right)} \Big/ {\sum_k \left(\varphi_P^k\right)} \tag{4.9}$$

Variations and additions to this generic four-phase algorithm process exist in different gradient based algorithms but they all follow the above trend. A common modification is to add a refinement step after $\widehat{D}_P$ has been calculated. In practice, however, a proper choice of the interpolation descriptors tends to reduce the efficacy of this refinement.

4.4 Novel Concepts and Contributions

The demosaicking process is highly dependent on choosing pixels that can accurately estimate missing colour content with minimal error.

In the case of gradient-based demosaicking, from equations (4.6) and (4.7), it is apparent that the selected pixels should be sufficient enough to form good initial estimates and interpolation descriptors. Particular care must be taken in establishing the interpolation descriptors. If the descriptors are generated with too few pixels, they lead to an overly smooth image where regions of fine detail and texture are blurred. If too many pixels are used in the descriptor formation, the outlier information from distant pixels may lead to inaccuracies in reconstruction. Distant pixels also introduce unnecessary complexity to the algorithm. A sufficient balance in pixel selection is required to maximise algorithm performance.

With the above consideration, this work introduces several new ideas to maximise performance:

i. An ***ordinal-direction driven exploitation*** of the quincuncial green plane of a Bayer or converted equivalent array
ii. The use of ***combinatorial geometry in the form of polyominoes*** to generate robust interpolation descriptors
iii. Concept of ***variable plane factors*** to prioritise weighting

It should be noted that these contributions are applied in the designed algorithm after the Bayerisation process and form part of the gradient-based demosaicking technique. As such they are considered in a Bayer or converted equivalent CFA environment.

4.4.1 The Ordinal Nature of the Green Plane

Assuming a Bayer or converted equivalent CFA arrangement, there are three colour planes forming the CFA – the red, green and blue plane. In this three plane arrangement shown in Figure 2.7, the green plane is the most populous with almost 50% of the total recorded CFA data. It is for this reason that proper demosaicking of the green plane will significantly improve overall image reconstruction. Established gradient-based methods use a cardinal direction interpolation system. A visual inspection, however, reveals that more pixels in the green plane lie in the ordinal directions than in the cardinal directions for the same bounding window size. This is shown in Figure 4.3 that illustrates the green pixel distributions in the North and North-East directions arising from the quincuncial arrangement of the green plane.

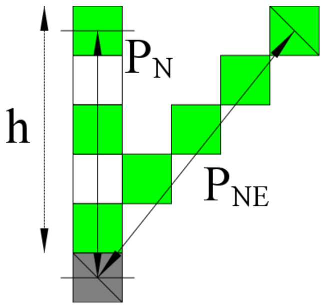


Figure 4.3 Paths in the North (N) and North-East (NE) directions in the green quincunx plane

For any window of size $(2h + 1)$ pixels, there are more ordinal-directed pixels present than cardinal ones. This results in a higher pixel packing along the ordinal directions. This is shown in Figure 4.4 for two different grid sizes, where P_{Car} represents the cardinal pixel count and P_{Ord} represents the ordinal pixel count.

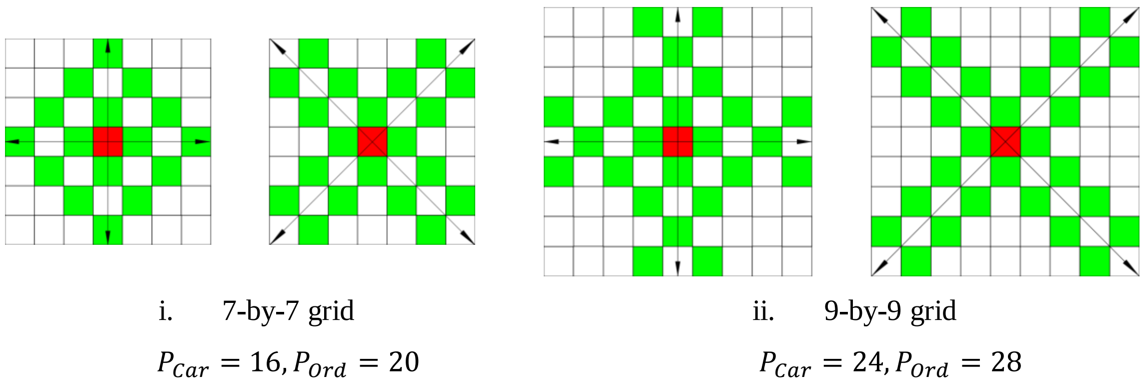


Figure 4.4 A comparison of cardinal and ordinal directed pixels over different Bayer grid sizes

This observation motivated the author to implement an ordinal-only interpolation process in the proposed algorithm.

4.4.2 Combinatorial Geometry, Polyominoes and Pentomino Inspired Paths

Any CFA is defined in combinatorial geometry as a square tessellation in two dimensional Euclidean space with sets of square blocks 'fitting together' to form a whole. Polyominoes are shapes made by connecting a certain number of equal-sized squares, each connected to another square along an edge [136]. From these definitions, polyominoes can be viewed as sub-sets of a Bayer CFA arrangement. This relationship and the concept of 'fitting together' motivated the author to study and use polyomino theory to generate the heuristic interpolation descriptors.

Polyominoes occur in various sizes depending on the number of interconnecting squares. The most recognisable use of polyominoes is in the popular game of *Tetris* – a puzzler using randomly generated tetrominoes (4-square polyominoes).

A subset of polyominoes must be determined to ensure that a sufficient number of pixels to form heuristic interpolation descriptors can be generated. Using the assumption that the CFA under consideration is a reduced Bayer equivalent of the RGBW CFA, the author imposed that for any general *n*-square polyomino set; the following conditions hold:

i. The maximum number of green pixels bounded by the polyomino, g, for odd or even valued *n*;

$$g_{odd} = \left(\frac{n+1}{2}\right) \\ g_{even} = \left(\frac{n}{2}\right) \tag{4.10}$$

ii. The number of paths found within the polyomino, p;

$$p = \frac{g!}{(g-2)!\,2!} \tag{4.11}$$

Exceptions exist to the above rules. From equations (4.10) and (4.11), an optimal value allowing for the generation of a sufficient number of unique descriptors in the reduced Bayer equivalent CFA was found to be at *n* = 5. This value was established by experimentally testing values of *n* from *n* = 1 to *n* = 6 and using equations (4.10) and (4.11) [91]. This subset of polyominoes are called pentominoes.

From polyomino theory, there are a total of 18 one-sided pentominoes (assuming rotations are not considered unique). All the possible pentomino blocks are shown in Figure 4.5 and blocks that form chirals (non-superimposable mirror images) are denoted by a letter having a prime symbol.

Each pentomino block of Figure 4.5 is used to generate two paths of the three possible. These paths, in turn, are used to select the sum-of-difference (SOD) term pairs to form the directional gradients used as the heuristic interpolation descriptors. The following rules were set as guidelines to ensure unique paths:

i. Two blocks forming a chiral pair will have their paths forming a chiral pair as well

ii. The pixel of interest must share a vertex with an edge square of the pentomino construct

The paths generated are shown in Figure 4.6 with the pixel of interest in grey. The figure considers the path generation is being performed in the green colour plane.

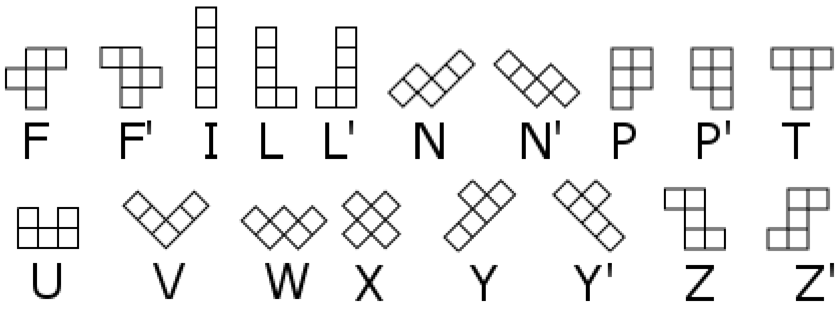


Figure 4.5 The 18 possible pentomino blocks using the Golomb letter naming system

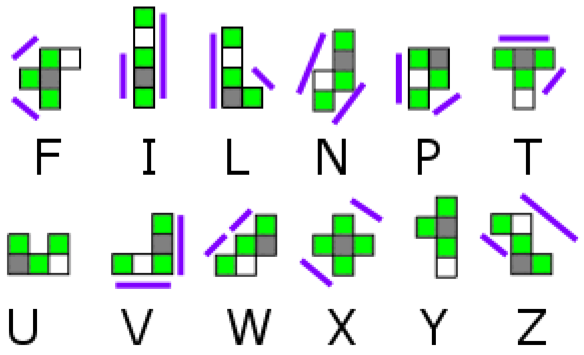


Figure 4.6 Generated pentomino paths

Considering the ordinal argument from Section 4.4.1 and based on the paths generated, the pentomino blocks ***N, P, T, V, W, X*** or ***Z*** are potential constructs for use. This is because blocks ***I, L*** and ***P*** have cardinal paths; ***F*** has paths in two different ordinal directions thus it has no preferred direction and the ***U*** and ***Y*** pentominoes generated no new paths that cannot be generated from ***F, P*** and ***T***.

The author selected the ***N, W*** and ***Z*** pentominoes to form the gradient selection paths due to their preferred and significant ordinal bias.

4.4.3 Variable Plane Factors

A reduced Bayer equivalent of the RGBW CFA will be of the form depicted in Figure 4.1 with three colour planes present. The inter-plane relationship between the three planes expanded in Figure 4.7 can take three distinct forms given in equation (4.12a):

$$\begin{gathered} Case\ 1: k_2 = k_3 = k_1 \\ Case\ 2: k_2 = k_3 \neq k_1 \\ Case\ 3: k_2 \neq\ k_3 \neq k_1 \end{gathered} \tag{4.12a}$$

Many established gradient-based methods treat these planes as equal when this in fact is not the case [86], [96], [135].

Consider the decomposed reduced Bayer equivalent RGBW CFA shown in Figure 4.7. The grey pixel point is a missing pixel point. This grey pixel point is located in the green plane but has a record of red colour data. When determining the missing colour content in this pixel of interest, it can be noted from inspection that the green plane has the most influence since the pixel physically resides in this plane. The red plane is of secondary importance as it is the plane containing a record of data in the CFA arrangement. The blue plane, in turn, offers no positional or colour record information as is deemed the least important contributor of information plane-wise.

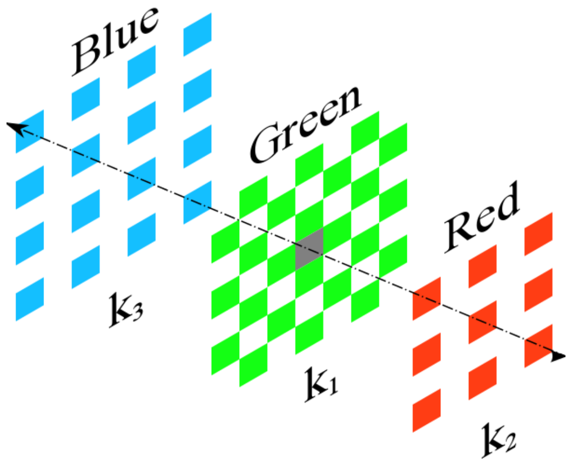


Figure 4.7 A plane-wise decomposition of the reduced Bayer equivalent of the RGBW CFA

From this intuitive reasoning, this variable inter-plane weighting can be reduced to a specific subset of Case 3:

$$k_3 < k_2 < k_1 \tag{4.12b}$$

Consequently, the three colour planes can be denoted in rank as follows, for the above case: the green plane has the largest impact (rank 1), the red plane follows (rank 2) and the blue plane that offers no contribution in position or colour has the least importance (rank 3). The author encoded this ranking system in the proposed algorithm using three variable factors: $k_1 = 1$, $k_2 = 0.8$ and $k_3 = 0.7$ where the subscript denotes the plane rank. This empirical determination is provided in Chapter 5.

The above factor values are used when determining missing colour content in the green colour plane. A similar treatment is applied when determining colour content in the red and blue colour planes.

4.5 The Proposed Algorithm

The proposed algorithm is divided into the following sections:

i. Reduction of the RGBW CFA to a Bayer equivalent
ii. Green Content Interpolation
iii. Blue and Red Content Interpolation

A flowchart depicting the processing of this sections is presented in Figure 4.8.

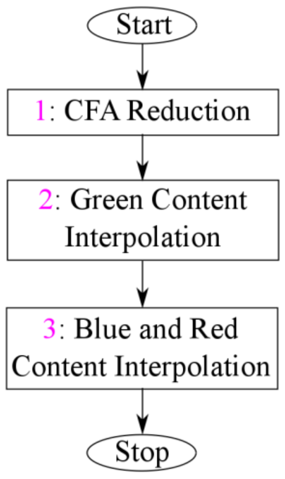

Start
1: CFA Reduction
2: Green Content Interpolation
3: Blue and Red Content Interpolation
Stop

Figure 4.8 Proposed algorithm flowchart

4.5.1 RGBW CFA Reduction

This phase of the proposed algorithm is performed by using the filter presented in equation (4.4) directly on the RGBW CFA data. This results in the reduced Bayer-equivalent representation that contains the white pixel data encoded in green equivalent pixels located at the white pixel points. The result of this process is shown in Figure 4.9 for a 7×7 segment. For ease of analysis in later stages of the algorithm, no distinction is made in the green plane between the original green pixel points and the green equivalent pixels that now populate the former white pixel points. This process is shown as a flowchart in Figure 4.10.

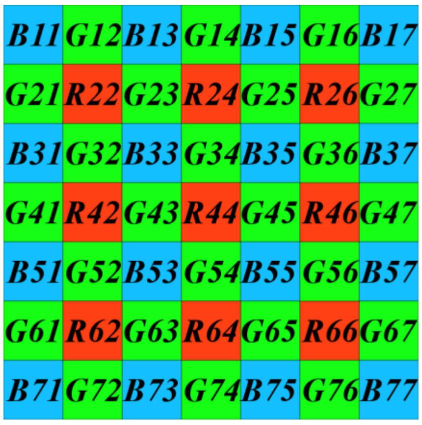

B11 G12 B13 G14 B15 G16 B17
G21 R22 G23 R24 G25 R26 G27
B31 G32 B33 G34 B35 G36 B37
G41 R42 G43 R44 G45 R46 G47
B51 G52 B53 G54 B55 G56 B57
G61 R62 G63 R64 G65 R66 G67
B71 G72 B73 G74 B75 G76 B77

Figure 4.9 A 7×7 segment of the reduced Bayer equivalent of the RGBW CFA

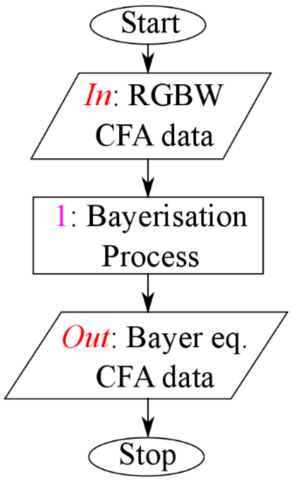


Figure 4.10 CFA Reduction flowchart

4.5.2 Green Plane Reconstruction

Consider the process of determining the green colour content present in the pixel *R44* in Figure 4.9. The initial estimates are established for the four ordinal directions as follows:

$$\begin{aligned}
\tilde{G}_{R44}^{NW} &= 0.5(G34 + G43) + \ (k_2k_3)(R44 - R22) \\
\tilde{G}_{R44}^{SW} &= 0.5(G54 + G43) + \ (k_2k_3)(R44 - R62) \\
\tilde{G}_{R44}^{SE} &= 0.5(G45 + G54) + \ (k_2k_3)(R44 - R66) \\
\tilde{G}_{R44}^{NE} &= 0.5(G45 + G34) + \ (k_2k_3)(R44 - R26)
\end{aligned} \tag{4.13}$$

The directional gradient in each ordinal direction is determined using all three colour planes in the manner highlighted in equation (4.7). Using the generation of the South-West (SW) directional gradient as an example and using Figure 4.11 as a reference for the associated path generation,

$$\Phi_{R44}^{SW} = \Phi_{Green,R44}^{SW} + \Phi_{Red,R44}^{SW} + \Phi_{Blue,R44}^{SW} + \varepsilon \tag{4.14}$$

Where ε is a small positive non-zero number to prevent a zero gradient result. The components of equation (4.13) are as follows:

$$\begin{aligned}\Phi_{Green,R44}^{SW} &= |path_N| + |path_W| + |path_Z| \\ \Phi_{Red,R44}^{SW} &= k_2(|R44 - R62|) \\ \Phi_{Blue,R44}^{SW} &= k_3(|B53 - B71|)\end{aligned} \tag{4.15}$$

And from Figure 4.9,

$$\begin{aligned}path_N &= (|G43 - G52| + |G45 - G52|) \\ path_W &= (|G34 - G43| + |G43 - G52|) \\ path_Z &= (|G54 - G63| + |G45 - G63|)\end{aligned} \tag{4.16}$$

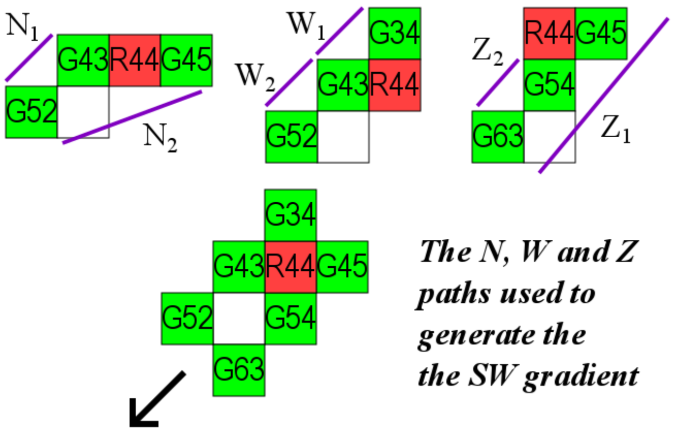


Figure 4.11 The South-West paths from pixel R44 using the N, W and Z pentomino blocks

By superimposing the ***N, W*** and ***Z*** pentomino blocks as shown in Figure 4.11, the proposed algorithm enforces an ordinal-only directed mechanism to generate the gradients using the sum-of-difference calculations shown in equation (4.16). The variable plane factor weighting proposed in Section 4.4.3 is implemented in equation (4.15) for the red and blue gradient terms.

The North-West (Φ_{R44}^{NW}), North-East (Φ_{R44}^{NE}) and South-East (Φ_{R44}^{SE}) terms are generated in a similar manner by rotating the pentomino paths in Figure 4.10 clockwise 90°, 180° and 270° respectively and applying equations (4.15) and (4.16). Additionally, to generate the red and blue terms of equation (4.15) for these three directions, the pixels lying flush in the ordinal direction are selected. For example, *R44, B55, R66* and *B77* all lie in the South-East direction line and are the terms to be adopted in equation (4.15).

Once all the directional terms are established, the proposed algorithm determines the weighting factors. This is done by applying equation (4.8) to the results of equation (4.14) for all ordinal directions and mathematically expressed as follows:

$$\begin{aligned} \varphi_{R44}^{SW} &= \frac{1}{\Phi_{R44}^{SW}} \\ \varphi_{R44}^{NW} &= \frac{1}{\Phi_{R44}^{NW}} \\ \varphi_{R44}^{NE} &= \frac{1}{\Phi_{R44}^{NE}} \\ \varphi_{R44}^{SE} &= \frac{1}{\Phi_{R44}^{SE}} \end{aligned} \tag{4.17}$$

After determining the weighting factor, the proposed algorithm uses polling maps similar to those used by Chen et al. [137] to establish the final missing colour value. The polling maps help determine whether interpolation is predominant in a particular ordinal direction or not. Equations (4.18) and (4.19) are used when the polling map shows preference to a NW-SE or a SE-NW direction respectively. Otherwise, equation (4.20) is used.

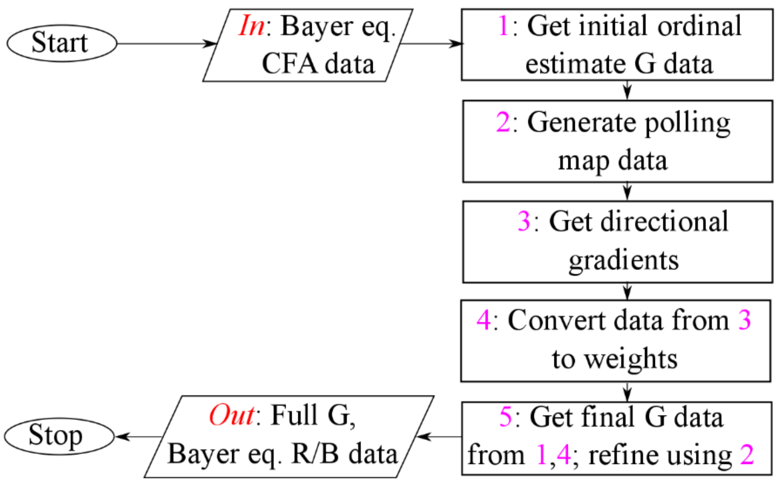
$$\hat{G}_{R44} = \left. \sum_{k\in(NW,SE)} \left\{\varphi_{R44}^{k} \tilde{G}_{R44}^{k}\right\} \middle/ \sum_{k\in(NW,SE)} \left\{\varphi_{R44}^{k}\right\} \right. \tag{4.18}$$

$$\hat{G}_{R44} = \left. \sum_{k\in(NE,SW)} \left\{\varphi_{R44}^{k} \tilde{G}_{R44}^{k}\right\} \middle/ \sum_{k\in(NE,SW)} \left\{\varphi_{R44}^{k}\right\} \right. \tag{4.19}$$

$$\hat{G}_{R44} = \left. \sum_{k\in(NW,SE,NE,SW)} \left\{\varphi_{R44}^{k} \tilde{G}_{R44}^{k}\right\} \middle/ \sum_{k\in(NW,SE,NE,SW)} \left\{\varphi_{R44}^{k}\right\} \right. \tag{4.20}$$

The final result $\hat{G}_{R44}$ is considered to be the missing colour content of the green plane for pixel local *R44*. The above steps are repeated to establish all the missing green colour content in the red pixel locations.

The proposed algorithm then works on the CFA data in a similar way to determine the green colour content in the blue pixel locations. During this step, the positions of the blue and red pixels in equations (4.13) though to (4.20) are switched without loss of generality. Once complete, the entire green plane is fully reconstructed. The entire process of reconstruction of green plane data is shown as a flowchart in Figure 4.12.



Start
In: Bayer eq. CFA data
1: Get initial ordinal estimate G data
2: Generate polling map data
3: Get directional gradients
4: Convert data from 3 to weights
5: Get final G data from 1,4; refine using 2
Out: Full G, Bayer eq. R/B data
Stop

Figure 4.12 Green Plane Reconstruction flowchart

4.5.3 Red and Blue Plane Reconstruction

Unlike the green plane, the red and blue colour planes are not quincuncial in nature. Consequently, reconstruction is divided into a two phase process. The first phase involves establishing content in an opposing colour planes (that is red colour content in the blue pixel record locations and vice versa) and the second involves finding missing red or blue colour content within the green locations.

(i) Opposing Plane Reconstruction in the Blue Plane

Consider the problem of establishing blue content in pixel *R44* in Figure 4.9. The proposed algorithm takes advantage of the fact that at this stage the green content in each red pixel has been established. The algorithm slightly modifies the initial estimation step of the generic gradient demosaicking shown in equation (4.6) instead using a difference based solution. The initial ordinal estimates are calculated using equation (4.21) provided:

$$\begin{aligned}\tilde{\rho}_{R44}^{SW} &= B53 - \hat{G}_{B53}\\ \tilde{\rho}_{R44}^{NW} &= B33 - \hat{G}_{B33}\\ \tilde{\rho}_{R44}^{NE} &= B35 - \hat{G}_{B35}\\ \tilde{\rho}_{R44}^{SE} &= B55 - \hat{G}_{B55}\end{aligned} \tag{4.21}$$

Directional gradients are found using a variant of equation (4.7) shown in equation (4.22) below where both actual green colour content from the CFA and those determined from Section 4.5.2 are used.

$$\Gamma^{k} = \Gamma^{k}_{G,actual} + \Gamma^{k}_{G,calculated} + \Gamma^{k}_{B} + \varepsilon \tag{4.22}$$

Where ε is a small positive non-zero constant. Considering the North-West (NW) direction for pixel *R44* in Figure 4.9, the components of equation (4.22) are:

$$\begin{aligned}\Gamma^{NW}_{R44(G,actual)} &= (|G43 - G32| + |G34 - G23|)\\ \Gamma^{NW}_{R44(G,calculated)} &= (|\hat{G}_{R44} - \hat{G}_{B33}| + |\hat{G}_{B33} - \hat{G}_{R22}|)\\ \Gamma^{NW}_{R44(B)} &= (|B33 - B55|)\end{aligned} \tag{4.23}$$

The remaining gradients Γ^{NE}_{R44}, Γ^{SE}_{R44} and Γ^{SW}_{R44} are found by repeating the process from equation (4.22) to form the weights. Once established, the reciprocals of the gradients are determined using the relationship in equation (4.24):

$$\gamma^{k}_{R44} = \frac{1}{\Gamma^{k}_{R44}} \tag{4.24}$$

Finally, the weights are applied to the initial estimates of equation (4.21) and added to $\hat{G}_{R44}$ to counter the difference relationship set in equation (4.21). This will result in a final estimate for the blue colour content in pixel location *R44* as shown in equation (4.25). To simplify the proposed algorithm, no polling maps were used in this phase.

$$\hat{B}_{R44} = \hat{G}_{R44} + \left\{ \sum_{k\in(NW,SW,SE,NE)} \{\gamma^{k}_{R44}\tilde{\rho}^{k}_{R44}\} \Big/ \sum_{k\in(NW,SW,SE,NE)} \{\gamma^{k}_{R44}\} \right\} \tag{4.25}$$

(ii) Opposing Plane Reconstruction in the Red Plane

The process outlined in equations (4.21) through to equation (4.25) is repeated to establish all the red colour content in blue pixel locations. The proposed algorithm switches the placing of the red and blue in equations (4.21) through to (4.25) to perform the inverse process of determining red colour content in blue pixel locations. Consider, for example, the process of determining red colour content in the pixel *B55* in Figure 4.9. The initial estimates for this pixel will take the form given in equation (4.26),

$$\begin{aligned}\tilde{\rho}^{SW}_{B55} &= R64 - \hat{G}_{R64}\\ \tilde{\rho}^{NW}_{B55} &= R44 - \hat{G}_{R44}\\ \tilde{\rho}^{NE}_{B55} &= R46 - \hat{G}_{R46}\\ \tilde{\rho}^{SE}_{B55} &= R66 - \hat{G}_{R66}\end{aligned} \tag{4.26}$$

The directional gradient general equation was be as follows:

$$\Gamma^{k} = \Gamma^{k}_{G,actual} + \Gamma^{k}_{G,calculated} + \Gamma^{k}_{R} + \varepsilon \tag{4.27}$$

And for the pixel under analysis, *B55*, interpolation in the SE direction would use a gradient formulation shown in equation (4.28):

$$\begin{gathered}\Gamma^{SE}_{B55(G,actual)} = (|G65 - G76| + |G56 - G67|) \\ \Gamma^{SE}_{B55(G,calculated)} = (|\hat{G}_{B55} - \hat{G}_{R66}| + |\hat{G}_{R66} - \hat{G}_{B77}|) \\ \Gamma^{SE}_{B55(R)} = (|R44 - R66|)\end{gathered} \tag{4.28}$$

After all the gradients, namely Γ^{SE}_{B55}, Γ^{NE}_{B55}, Γ^{NW}_{B55} and Γ^{SW}_{B55}, are found using equation (4.27) the weights are found using equation (4.29) before the final value is established using equation (4.30). These equations are given below:

$$\gamma^{k}_{B55} = \frac{1}{\Gamma^{k}_{B55}} \tag{4.29}$$

$$\hat{R}_{B55} = \hat{G}_{B55} + \left\{ {\sum_{k\in(NW,SW,SE,NE)}\{\gamma^{k}_{B55}\tilde{\rho}^{k}_{B55}\}} \Big/ {\sum_{k\in(NW,SW,SE,NE)}\{\gamma^{k}_{B55}\}} \right\} \tag{4.30}$$

Equation (4.26) through to equation (4.30) are recursively applied to all the blue pixel points to generate the associated red colour content. Once complete, the blue and red colour planes both observe a quincuncial profile.

(iii) Reconstruction in Green Pixel Locations

At this point in the processing, the proposed algorithm has a fully constructed green plane and quincuncial blue and red planes. The remaining undetermined content are the blue and red colour values in green pixel locations. The proposed algorithm uses a procedure similar to the green plane reconstruction. This is because the missing colour blue or red content now occurs in a quincuncial plane – a situation similar to that in Section 4.5.2.

Consider the problem of establishing red content in pixel *G45*. An initial estimate is generated and is of a simpler form than equation (4.13) due to the availability of previously missing colour content data. This is shown in equation (4.31) using the North-West (NW) direction as an example:

$$\tilde{R}^{NW}_{G45} = 0.25(\hat{R}_{35} + R24 + \hat{R}_{B33} + R44) \tag{4.31}$$

Estimates in the three remaining ordinal directions are generated in the same way. The gradients, associated weights and final colour content values are then established. Equations (4.32), (4.33) and (4.34) illustrate the proposed algorithm's workflow in the North-West direction in establishing the red colour content in pixel *G45*. This is done in the remaining ordinal directions and the final colour value is determined using equation (4.35).

$$\Phi_{G45}^{NW} = \Phi_{Red,G45}^{NW} + \Phi_{Green,G45}^{NW} + \varepsilon \tag{4.32}$$

Where ε is a small positive non-zero constant and

$$\begin{aligned} \Phi_{Red,G45}^{NW} &= \left(\left|\hat{R}_{B35} - R24\right| + \left|\hat{R}_{B33} - R44\right|\right) \\ \Phi_{Green,G45}^{NW} &= \left(|G45 - G34| + |G34 - G23|\right) \end{aligned} \tag{4.33}$$

$$\varphi_{G45}^{k} = \frac{1}{\Phi_{G45}^{k}} \tag{4.34}$$

$$\hat{R}_{G45} = {\sum_{k\in(NW,NE,SE,SW)}\left\{\varphi_{G45}^{k}\tilde{R}_{G45}^{k}\right\}} \Big/ {\sum_{k\in(NW,NE,SE,SW)}\left\{\varphi_{G45}^{k}\right\}} \tag{4.35}$$

The process outline in this is repeated to establish all the red colour content in the green pixel locations. In the same manner, the proposed algorithm works on determining the missing blue content in the green pixel locations. Equations (4.36), (4.37), (4.38) and (4.39) outline the steps taken in establishing the blue colour content in pixel *G34* over a North-East interpolation direction.

$$\tilde{B}_{G34}^{NE} = 0.25(\hat{B}_{R24} + B15 + \hat{B}_{R26} + B35) \tag{4.36}$$

$$\Phi_{G34}^{NE} = \Phi_{Red,G34}^{NE} + \Phi_{Green,G34}^{NE} + \varepsilon \tag{4.37}$$

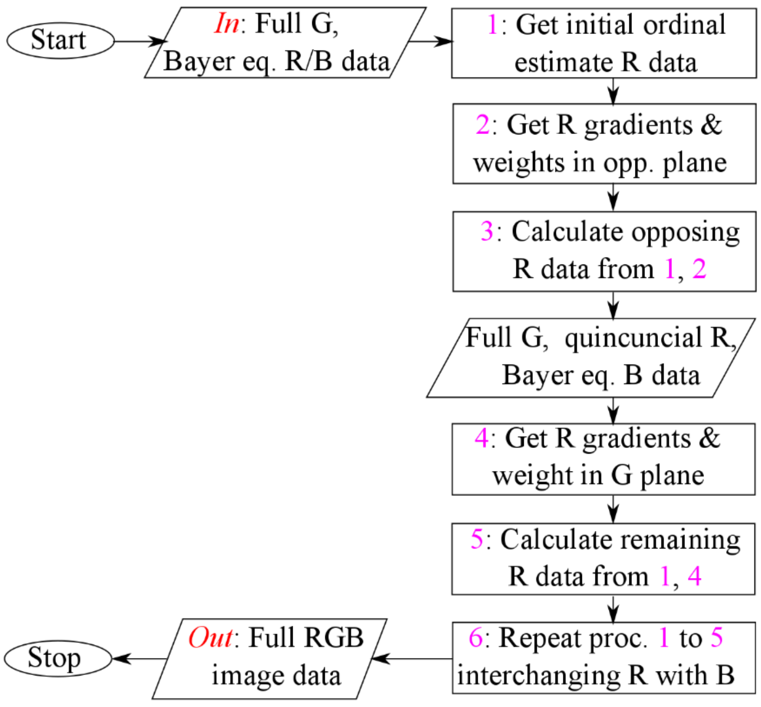
$$\begin{aligned} \Phi_{Blue,G34}^{NE} &= \left(\left|\hat{B}_{R24} - B15\right| + \left|\hat{B}_{R26} - B35\right|\right) \\ \Phi_{Green,G34}^{NE} &= \left(|G34 - G25| + |G25 - G16|\right) \end{aligned} \tag{4.38}$$

$$\varphi_{G34}^{k} = \frac{1}{\Phi_{G34}^{k}} \tag{4.39}$$

The final estimate for the pixel G34 is given by equation (4.40):

$$\hat{B}_{G34} = \sum_{k\in(NW,NE,SE,SW)}\{\varphi_{G34}^{k}\tilde{B}_{G34}^{k}\} \Big/ \sum_{k\in(NW,NE,SE,SW)}\{\varphi_{G34}^{k}\} \tag{4.40}$$

This process is repeated until all the missing blue colour content in the green pixel points is found. When all the missing colour points have been reconstructed, the demosaicking process is complete. The red and blue pixel processing is depicted in Figure 4.13.



Start
In: Full G, Bayer eq. R/B data
1: Get initial ordinal estimate R data
2: Get R gradients & weights in opp. plane
3: Calculate opposing R data from 1, 2
Full G, quincuncial R, Bayer eq. B data
4: Get R gradients & weight in G plane
5: Calculate remaining R data from 1, 4
6: Repeat proc. 1 to 5 interchanging R with B
Out: Full RGB image data
Stop

Figure 4.13 Red and Blue Plane Reconstruction flowchart

5 SIMULATION PROCEDURE AND RESULTS

The experimental phase of this research work was carried out fully in simulation through the use of *MATLAB*® (matrix laboratory) software platform. *MATLAB*® is a numerical computational tool with built-in graphical, simulation modelling and programming functionality. This software environment was used because of:

i. the lack of physical real world Bayer and RGBW CFA CMOS sensors in the desired pixel dimensions and arrangement to perform the experiment
ii. the need for a ground-truth reference image to perform a comparative analysis with the demosaicked equivalent to assess algorithm performance
iii. the fact that several established state-of-the-art algorithms are not available in the public domain due to strict confidentiality and non-disclosure agreements and must themselves be modelled from their journal and/or patent information

Modelling the experiment in the *MATLAB*® platform allowed for the simulation of the image acquisition as well as the colour processing and interpolation (demosaicking) shown in Figure 2.2. It also allowed for the modelling of a comparison between the original ground-truth reference image and the final reconstructed demosaicked image.

It should be noted that the associated process shown in Figure 2.2 that form part of the image processing pipeline such as noise filtering, white balance adjustment and post processing enhancement techniques are not modelled because:

i. this work is primarily focused on the efficacy of demosaicking process
ii. these associated processes are considered independently of the demosaicking stage
iii. some processes such as noise tend to be device-specific due to design and choice of materials of the physical camera and modelling them would constrain the applicability of the proposed algorithm
iv. some enhancement processes, such as gamut correction, are purely subjective and image quality assessment would vary amongst observers

5.1 Simulation Process

Each of the aforementioned phases of image acquisition, demosaicking and comparison were modelled as *MATLAB*® algorithm function blocks in line with the physical equivalent shown in Figure 2.7. The functional *MATLAB*® code blocks created and forming part of this research study are provided in their entirety in Appendix A.

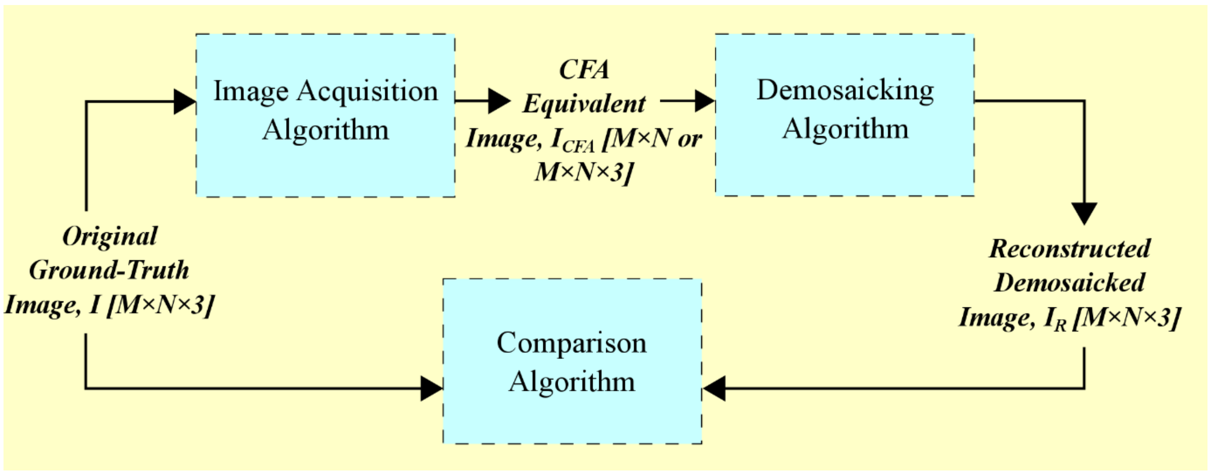


Image Acquisition Algorithm
CFA Equivalent Image, I_{CFA} [M×N or M×N×3]
Demosaicking Algorithm
Original Ground-Truth Image, I [M×N×3]
Reconstructed Demosaicked Image, I_R [M×N×3]
Comparison Algorithm

Figure 5.1 The experimental MATLAB simulation process

The ***image acquisition algorithm block*** takes the three dimensional M×N×3 original ground truth image, $\boldsymbol{I}$, subsampling it to the CFA equivalent image representation, $\boldsymbol{I_{CFA}}$. In this work, this block has two functional variants depending on whether the algorithm under consideration is driven using a Bayer or an RGBW colour filter array. If the algorithm uses a Bayer CFA, the resulting CFA equivalent image is two dimensional. However, in the RGBW case, the image acquisition block would produce a three dimensional CFA equivalent image. Code blocks detailing these processes are provided in Section A.2.

The ***demosaicking algorithm block*** is responsible for the conversion of the CFA representation, $\boldsymbol{I_{CFA}}$, to a fully reconstructed demosaicked image, $\boldsymbol{I_R}$. This block is the main area of study and a functional block variant was created for the proposed algorithm as well as each established state-of-the-art demosaicking algorithm that formed part of the overall test bed. The main aim of this block is to model equation (2.2). The proposed algorithm is given in Section A.1 while the test bed algorithms are provided in Section A.4. It should be noted, due to confidentiality agreement constraints in industry, the author created and implemented *MATLAB®* versions of the state-of-the-art test bed demosaicking methods from their initial journal, conference periodical or patent formulation.

The final simulation process is the ***comparison algorithm block*** detailed in Section A.3. Its function is to provide a quantitative and qualitative analysis of the overall reconstruction by comparing the demosaicked image, $\boldsymbol{I_R}$, to the original image, $\boldsymbol{I}$. A quantitative analysis is achieved by implementing a comparison using the established image quality assessment techniques laid out in the conceptual framework in Chapter 3. These are the mean square error (MSE), colour peak signal-to-noise ratio (CPSNR) and the structural and feature similarity indices (SSIM/FSIM). Each assessment method is developed as *a MATLAB®* function and combined to form the algorithm functional block. A qualitative analysis was implemented by having the comparison algorithm block saving both the original and reconstructed image in computer storage for the purposes of display to a human observer at a later time.

5.2 Empirical Determination of Corrective Terms (*k*₂, *k*₃ and *ε*)

Prior to using the proposed algorithm; a determination of corrective terms was performed. From equations (4.6) and (4.7), any gradient based demosaicking algorithm may have up to two corrective terms; denoted as c_1 and c_2. Many of the established algorithms either set these terms to zero or give arbitrary values to these terms to fulfil the fractional requirement of c_1 and the small positive value requirement of c_2.

However, due to the fact that these values are themselves used in the initial estimate stage, the choice of value may impact demosaicking performance. Consequently, the author felt it was necessary to empirically determine these corrective terms. Comparing equation (4.6) to (4.13) and equation (4.7) to (4.14), (4.22), (4.27), (4.32) and (4.37) the proposed algorithm designed the corrective terms to be:

$$\begin{gathered} c_1 = (k_2 k_3) \\ c_2 = \varepsilon \end{gathered} \tag{5.1}$$

Where the variable plane factors k_2 and k_3 constitute the first corrective term and the small positive constant ε is the second. To analyse the effect of these corrective terms, a base gradient demosaicking algorithm was created and applied to the reconstruction of Bayer CFA content using the simulation process highlighted in Figure 5.1. This base algorithm was created by using simplified variants of equations (4.13) and (4.14) that operate in the cardinal directions and primarily focus on the green colour plane. These are shown below in equation (5.2) through to equation (5.4):

$$\begin{aligned} \tilde{G}^N_{R44} &= G34 + (k_2 k_3)(R44 - R24) \\ \tilde{G}^W_{R44} &= G43 + (k_2 k_3)(R44 - R42) \\ \tilde{G}^S_{R44} &= G54 + (k_2 k_3)(R44 - R64) \\ \tilde{G}^E_{R44} &= G45 + (k_2 k_3)(R44 - R46) \end{aligned} \tag{5.2}$$

$$\Phi^k_{R44} = \Phi^k_{Green,R44} + \varepsilon \tag{5.3}$$

Where:

$$\begin{aligned} \Phi^N_{R44} &= (|G34 - G23| + |G34 - G25|) + \varepsilon \\ \Phi^W_{R44} &= (|G43 - G32| + |G43 - G52|) + \varepsilon \\ \Phi^S_{R44} &= (|G54 - G63| + |G54 - G65|) + \varepsilon \\ \Phi^E_{R44} &= (|G45 - G36| + |G45 - G56|) + \varepsilon \end{aligned} \tag{5.4}$$

The weights of this base algorithm and the final value for missing pixels in the green colour plane are established using equations (5.5) and (5.6) that are cardinal variants of equations (4.17) and (4.20) respectively.

$$\varphi_{R44}^{k} = \frac{1}{\Phi_{R44}^{k}} \tag{5.5}$$

$$\hat{G}_{R44} = \sum\nolimits_{k\in(N,S,E,W)}\{\varphi_{R44}^{k}\tilde{G}_{R44}^{k}\} \Big/ \sum\nolimits_{k\in(N,S,E,W)}\{\varphi_{R44}^{k}\} \tag{5.6}$$

This treatment explaining the synthesis of this base algorithm is also provided in the author's paper outlining corrective term usage [138]. The three forms of equation (4.12a) were tested using several images from the Kodak and McMaster-IMAX picked at random. Normalising k_1 to 1, k_2 and k_3 were varied from 0 to 2 in steps of 0.1. Their CPSNR, SSIM and FSIM values were recorded. The optimal values of k_2 and k_3 for each form of equation (4.12a) were determined and their values noted and these are shown in Table 5.1. The observed the optimal values were at k_2 = 0.8 and k_3 =0.7. These were the values used and indicated in Chapter 4.

For the second corrective term denoted by ε, a preliminary estimation of value was performed by exposing two images from two different image sets (*kodim21* from the Kodak Image Set and *mcm04* from the McMaster-IMAX Image Set) to the same base gradient algorithm provided in [138] and varying the value of ε logarithmically. Using equation (4.7), only the green colour plane was considered during the ε analysis. The results of the reconstruction using the same base gradient algorithm but differing values of ε is shown in Figures 5.2 and 5.3. From simple visual inspection, the optimal value was between ε=1 and ε=100.

Table 5.1 Performance metric variations for different k_2 and k_3 combinations

Set	Case 1: $\mathbf{k_2 = k_3 = 1}$	Case 2: $\mathbf{k_2 = k_3 = 0.8}$	Case 3: $\mathbf{k_2 = 0.8, k_3 = 0.7}$
		CPSNR	
Kodak	38.469	41.111	**41.356**
McMaster-IMAX	37.661	39.407	**39.756**
		SSIM	
Kodak	0.9721	0.9813	**0.9818**
McMaster-IMAX	0.9512	0.9652	**0.9676**
		FSIM	
Kodak	0.9690	0.9720	**0.9723**
McMaster-IMAX	0.9655	0.9684	**0.9688**

Refinement of the ε value was achieved by exposing each image in the Kodak set to 33 different ε values in the range of ε=0 to ε=100. The values of PSNR, SSIM and FSIM were calculated and recorded at each step point and the average result for each metric over the entire Kodak image set was determined.

The resultant trend is plotted in Figure 5.4. This process was repeated in three other image sets: McMaster-IMAX, McGill University Calibrated Color Image Set [139] and the default local images found on a Windows 7 computer.

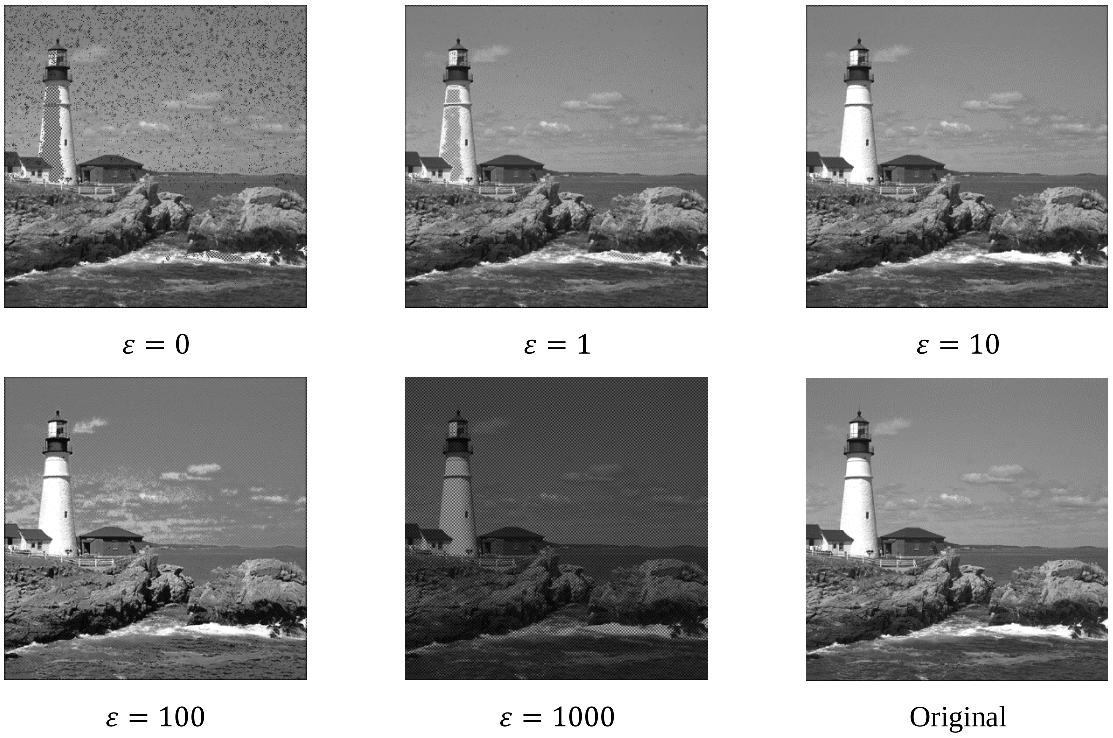


Figure 5.2 A comparison of various ε values using kodim21 of the Kodak Image Set

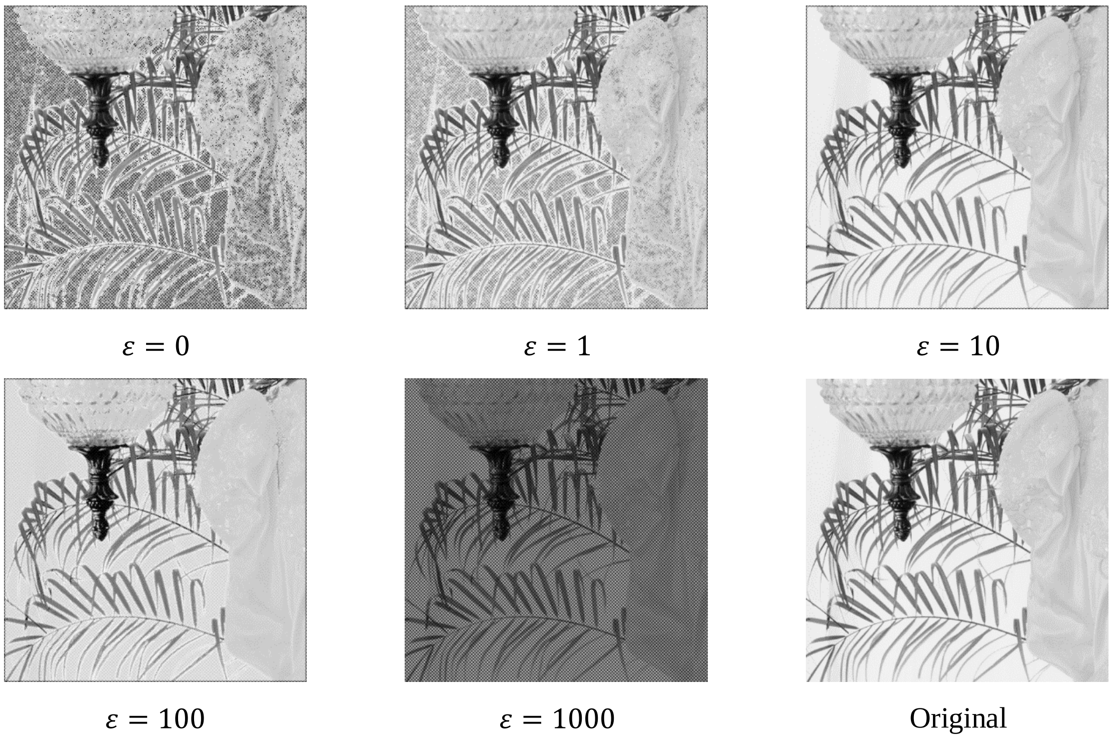
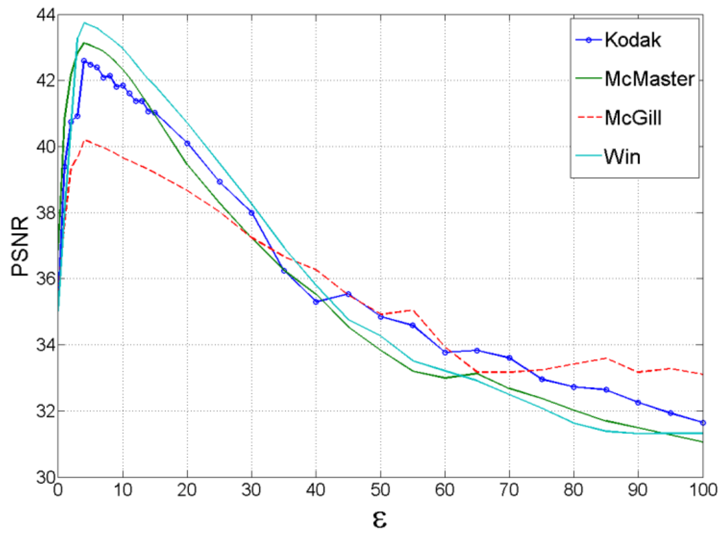


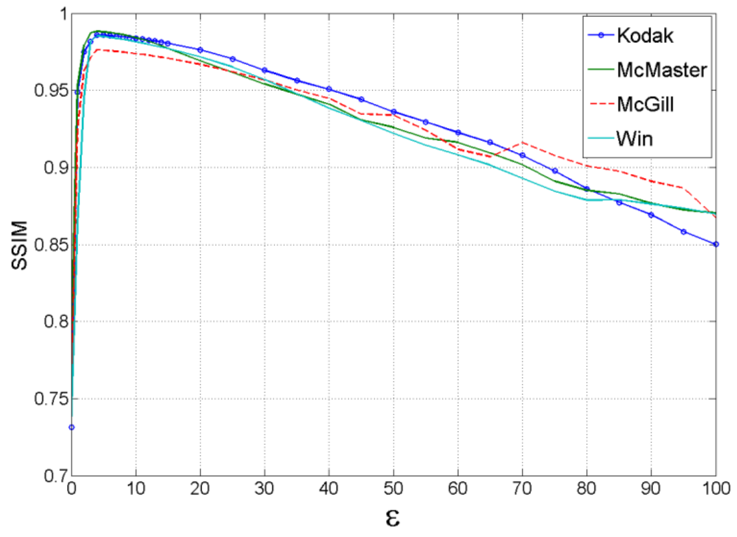
Figure 5.3 A comparison of various ε values using mcm03 of the McMaster-IMAX Image Set

From Figure 5.4, the optimal value over all performance metrics was determined to occur at ε = 4. This was the value used in the proposed algorithm outlined in Chapter 4 and Section A.1 of Appendix A.

(a) PSNR Variation



(b) SSIM Variation



(c) FSIM Variation

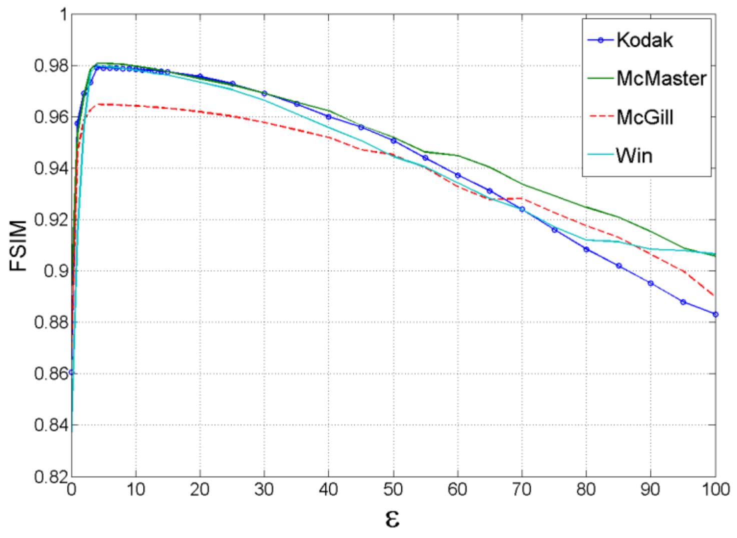


Figure 5.4 Variation of performance metrics for different values of ε

5.3 Simulation Methodology and Testing Procedure

The simulation of image acquisition, demosaicking and comparison processes was performed using *MATLAB*® R2015b running on an Intel® Core ™ i5-6200 CPU @ 2.39 GHz processor.

The testing procedure was as follows. For each image set defined in Table 3.2:

i. Every image was decomposed to a Bayer CFA equivalent (or RGBW CFA equivalent in the case of some of the RGBW methods lacking a Bayerisation process) by passing it through the image acquisition algorithm block.

ii. The CFA equivalent image was then passed through all the algorithms forming the experimental test bed in turn. There were nine (9) algorithms considered in total, comprising the proposed method along with the established state-of-the-art current methods. Each method is provided in Appendix A and described below in Table 5.1. Each algorithm is realised as a demosaicking algorithm block and results in a unique demosaicked image.

Table 5.2 List of algorithms forming experimental test bed

S/n	Algorithm	Heuristic Class	CFA	Year Developed
1	Constant Difference Based Interpolation (CDBI) [68]	Constant Hue Based	Bayer	n/a
2	Edge Directed Interpolation (EDI) [76]	Edge	Bayer	2006
3	Malvar-He-Cutler algorithm (MHC) [79]	Edge	Bayer	2004
4	Wang algorithm (Wang) [81]	Gradient	Bayer	2012
5	Edge Strength Filter Based Interpolation (ESFBI) [83]	Gradient	Bayer	2012
6	Multi-Gradient Based Interpolation (MGBI) [86]	Gradient	Bayer	2013
7	Average Colour Ratio algorithm (ACR) [140]	Edge	RGBW	2014
8	Edge Directed Colour Ratio algorithm (EDCR) [140]	Edge	RGBW	2014
9	*Proposed algorithm*	*Gradient*	*Bayerised RGBW*	*2017*

iii. In step (ii), a maximum bounding window region of five pixels-width was used as padding and ignored in the reconstruction phase so that the results over different algorithms could be normalised. This was done as different algorithms in the experimental test bed employ different path descriptor lengths to establish edge or gradient information.

iv. Each reconstructed version of the image under consideration was then passed in turn to the comparison algorithm block for quantitative determination of the four image quality assessment metrics; namely MSE, CPSNR, SSIM and FSIM$_C$. The values obtained were recorded. The reconstructed version was then stored for qualitative presentation.

v. The above steps were repeated in sequence until all the images in an image set were processed. A geometric mean was then taken for each of the algorithms. A geometric mean was chosen because it is more resistant to perturbation from outlier information than an arithmetic mean. These perturbations were inherent due to the random nature of the images.

vi. An overall rank was generated from the geometric mean data to show the position of the proposed algorithm relative to all the test bed algorithms. These values and the associated geometric mean data are highlighted in the detailed tabulation of simulation results found in Appendix B.

vii. After all the six selected image sets had been processed in the manner outlined in steps (i) through (vi), a compilation of the geometric means was performed and the data is presented in Tables 5.3 through to 5.6.

5.4 Compiled Experimental Simulation Results

Tables 5.3 – 5.6 below present the geometric mean value data and associated ranking of each algorithm over each image set collated from the values provided in Appendix B. A two decimal point resolution for the MSE and CPSNR (both range from 0 to infinity) and a three decimal point resolution for the SSIM and FSIM$_C$ (both range from 0 to 1) was considered adequate. As previously mentioned, geometric mean rather than arithmetic mean evaluation was used to ensure average results were more resistant to outlier data effects.

Table 5.3 below highlights the average MSE values for each of the nine test bed algorithms when exposed to the 115 images from the six image databases. The proposed algorithm achieved a median performance rank of 6 overall. The raw image data values are tabulated in Tables B.1 through to B.6 of Appendix B.

Table 5.3 Geometric Average MSE evaluation values and associated ranking

Image Set (No. of Images)	**CDBI**	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Proposed**	**Rank**
USC-SIPI(16)	6.99	6.07	**5.39**	5.72	7.76	6.96	11.72	13.21	6.58	4
Kodak (24)	8.42	7.19	3.62	4.95	3.13	**2.74**	11.57	10.40	5.91	5
McMaster-IMAX (18)	6.39	5.27	**4.71**	4.88	6.44	5.87	9.61	12.72	5.86	4
Condat (30)	7.84	6.12	5.42	**5.27**	6.45	5.96	11.72	7.43	6.82	6
ARRI (12)	2.14	1.53	1.63	**1.29**	2.40	2.72	2.54	4.32	2.85	8
Custom (15)	4.66	4.17	1.55	2.47	**1.27**	1.28	6.38	5.48	3.13	5
Average (115)	5.56	4.56	**3.29**	3.64	3.81	3.64	7.93	8.22	4.90	6

Table 5.4 gives the average CPSNR evaluation values for the entire algorithm test bed over the entire image databases chosen. From the compilation of the raw data from Tables B.7 to B.12 in Appendix B, the proposed algorithm achieves the premier CPSNR geometric mean performance of 40.78 dB over the six chosen image sets.

Table 5.4 Geometric Average CPSNR evaluation values (in dB) and associated ranking

Image Set (No. of Images)	**CDBI**	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Proposed**	**Rank**
USC-SIPI(16)	33.27	38.12	38.11	38.31	37.60	37.44	37.48	36.76	**39.54**	1
Kodak (24)	32.80	38.84	39.72	40.12	41.23	**41.70**	36.75	36.61	41.39	2
McMaster-IMAX (18)	33.71	38.89	38.94	**38.96**	38.44	38.43	37.64	36.92	38.94	2
Condat (30)	32.34	38.43	38.29	38.44	38.18	38.09	36.75	36.84	**38.90**	1
ARRI (12)	38.44	**43.68**	43.06	42.50	41.57	40.73	42.44	41.11	42.61	3
Custom (15)	37.29	41.48	43.32	43.39	**44.99**	44.45	39.04	38.84	43.55	3
Average (115)	34.56	39.86	40.18	40.24	40.25	40.07	38.30	37.81	**40.78**	1

The average SSIM performance is presented in Table 5.5 for the algorithm testbed. The compiled data from the raw image values given in Section B.3 of Appendix B shows that the proposed algorithm performs second best, only behind the CDBI technique by a value of 0.002 (or 0.2%)

Table 5.5 Geometric Average SSIM evaluation values and associated ranking

Image Set (No. of Images)	**CDBI**	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Proposed**	**Rank**
USC-SIPI(16)	0.956	0.948	0.949	0.947	0.944	0.937	0.947	0.937	**0.961**	1
Kodak (24)	0.979	0.975	0.979	0.979	**0.981**	0.972	0.954	0.942	0.979	2
McMaster-IMAX (18)	**0.979**	0.973	0.970	0.969	0.961	0.956	0.969	0.958	0.972	3
Condat (30)	**0.983**	0.978	0.976	0.976	0.969	0.956	0.969	0.941	0.976	3
ARRI (12)	**0.998**	0.996	0.998	0.998	0.997	0.954	0.998	0.989	0.997	5
Custom (15)	**0.997**	0.995	0.994	0.995	0.988	0.971	0.994	0.985	0.995	2
Average (115)	**0.982**	0.977	0.977	0.977	0.973	0.958	0.971	0.958	0.980	2

From the raw image data of the selected image sets provided in Section B.4 of Appendix B, Table 5.6 is generated. This table provides the geometric mean data for the $FSIM_C$ image quality assessment metric for all the algorithms in the demosaicking test bed over the selected image databases. From Table 5.6, it is noted that the proposed algorithm ties for first place in performance with the CDBI and EDI methods with an average value of 0.991.

Table 5.6 Geometric Average FSIM$_C$ evaluation values and associated ranking

Image Set (No. of Images)	**CDBI**	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Proposed**	**Rank**
USC-SIPI(16)	**0.990**	0.987	0.973	0.978	0.968	0.934	0.975	0.961	0.988	2
Kodak (24)	**0.994**	0.993	0.982	0.986	0.980	0.948	0.976	0.962	0.989	3
McMaster-IMAX (18)	0.991	0.993	0.980	0.984	0.977	0.953	0.979	0.967	**0.993**	1
Condat (30)	0.990	**0.991**	0.979	0.984	0.975	0.946	0.976	0.940	0.984	3
ARRI (12)	0.990	0.994	**0.999**	0.998	0.998	0.972	0.999	0.988	0.998	3
Custom (15)	0.990	0.988	0.995	**0.997**	0.992	0.970	0.993	0.982	0.995	2
Average (115)	**0.991**	**0.991**	0.985	0.988	0.981	0.954	0.983	0.966	**0.991**	1

6 ANALYSIS AND DISCUSSION OF RESULTS

From the simulation process outlined in Chapter 5, the 115 test images exposed to the nine different demosaicking algorithms of the test bed and four image quality assessment metrics resulted in a data set of 4,140 values. To make logical inferences from this data set, the author subdivided the analysis into three main sections:

i. *RGBW CFA domain analysis*: to test the efficacy of the proposed algorithm's Bayerisation process and establish whether it is of any benefit
ii. *Single plane reconstruction analysis*: to test the performance of the proposed algorithm's reconstruction in the green colour plane – the most significant of the three colour planes from a human physiological viewpoint
iii. *Full colour and object reconstruction analysis*: to measure the overall performance of the algorithm when compared to the entire test bed

The RGBW only analysis operates on three of the nine algorithms – the Average-based Colour Reconstruction algorithm (ACR) [140], the Edge Detection-based Colour Reconstruction algorithm (EDCR) [140] and the proposed method. In this category, all of the image quality metrics are considered. The single and full colour analysis techniques add the Bayer based methods – that is the Constant Difference Based Interpolation (CDBI) [68], Edge Directed Interpolation (EDI) [76], Malvar-He-Cutler algorithm (MHC) [79], Wang algorithm (Wang) [81], Edge Strength Filter Based Interpolation (ESFBI) [83] and the Multi-scale Gradient Based Interpolation (MGBI) [86].

In the single plane reconstruction analysis, the MSE data established on the green plane is compared over all algorithms. Finally, the full colour and object fidelity analysis makes use of the CPSNR, SSIM and $FSIM_C$ data. It should be noted here, as stated in Table 3.3, that an improvement in performance in an algorithm is observed when the MSE is lower and the CPSNR, SSIM and $FSIM_C$ are higher than comparative methods.

For the purpose of analysis, all graphs plotted in this section are oriented in such a manner that the Bayer-based algorithms increase in descriptor complexity as one moves from left to right: that is from constant hue-based descriptors (CDBI); to edge-based descriptors (EDI, MHC) and finally to gradient-based descriptors (Wang, ESFBI, MGBI). Furthermore, the complexity within each descriptor class increases from left to right. For example, MHC is a more complex algorithm when compared to EDI. By the same token the MGBI algorithm is the most complex of its descriptor class.

6.1 RGBW CFA Domain Reconstruction Analysis

Extracting the geometric mean data from algorithms working in the RGBW CFA domain from the results tables in Chapter 5, it is observed that the proposed method has a superior performance to the two other RGBW algorithms in the test bed. This is because the proposed algorithm exhibits the lowest MSE and highest CPSNR, SSIM and FSIM_C values of the subset. This is shown in Table 6.1.

Table 6.1 RGBW CFA domain algorithm data

	ACR	EDCR	Proposed
Average MSE	7.93	8.22	**4.90**
Average CPSNR (dB)	38.30	37.81	**40.78**
Average SSIM	0.971	0.958	**0.980**
Average FSIMc	0.983	0.966	**0.991**

From the geometric average MSE data from Table 6.1, it is noted that the proposed algorithm improves the green plane reconstruction by a factor of approximately 1.6 over both the ACR and EDCR algorithms. This was attributed to the fact that the Bayerisation process introduces more green values into the CFA data prior to demosaicking. Comparing the demosaicking processes outlined in Figures 2.7 and Figure 2.8, the Bayer CFA has twice the number of green samples per unit than the equivalent RGBW (panchromatic) CFA of the same dimensional size. Theoretically, therefore, a reconstruction of this green plane would experience half the number of estimation errors in the Bayer CFA compared to the panchromatic RGBW case if the demosaicking process were considered equal. This is approximately what it observed and the author attributes the factor to be at 1.6 rather than 2 because the Bayerisation process itself introduces estimation errors.

In the CPSNR geometric mean data, the proposed method also exceeds the established ACR method by 2.97 dB and the EDCR method by 2.48 dB. The author attributed this to:

i. the *larger green sample size available* to the demosaicking algorithm
ii. the *ordinal nature* of the proposed demosaicking algorithm

The larger number of pixels available in the green plane increases CPSNR performance because its value is largely dependent on pixel statistics. The larger the number of samples taken the better the CPSNR. Concurrently, the ordinal nature of the demosaicking algorithm allows the exploitation of the tighter pixel packing along these directions, leading to more accurate estimates in all three colour planes.

It is this ordinal demosaicking regime that the author also attributes to the proposed algorithm yielding higher SSIM and FSIM_C values than the established panchromatic demosaicking techniques. The SSIM and FSIM operate on analysing object reconstruction. Using Figure 4.3 as reference, by demosaicking using ordinal path descriptors in the green plane, there is a lower likelihood of over-smoothing an object edge. This is particularly important in regions of object edge transition.

6.2 Single Plane Colour Reconstruction Analysis

The single plane colour reconstruction analysis is performed by considering the MSE values obtained from the reconstruction of the green plane. The RGBW MSE values were considered in the previous section. Expanding the MSE value analysis to include all the demosaicking algorithms in the test bed; as shown in Figure 6.1, it is observed that the proposed method performs poorly when compared to the Bayer CFA based techniques.

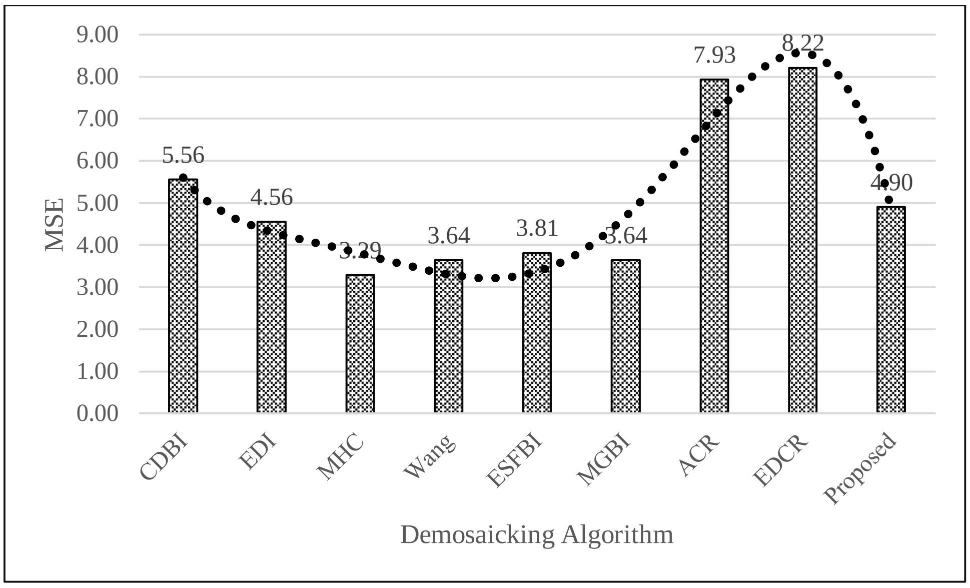


Figure 6.1 Graph of average geometric mean MSE values over all test bed demosaicking algorithms in all selected image sets

This is a departure from the variation observed in the RGBW only methods. However, the author believes this is to be expected. This is because while shifting the sensor data from the RGBW to the Bayer domain prior to demosaicking, the Bayerisation process does so using estimate data. As such, the green plane does not have a 50% exact sample data profile but a 25% exact sample data plus 25% estimate data profile instead. In the RGBW domain, only 25% of the sensor contains exact sample data used for reconstruction. This median property is quantitatively proven by calculating the geometric mean values of the algorithms in the two domains while isolating the proposed method. This process is outlined in equation (6.1).

$$\begin{gathered} MSE_{Bayer} = \sqrt[6]{(MSE_{CDBI} \times MSE_{EDI} \times MSE_{MHC} \times MSE_{Wang} \times MSE_{ESFBI} \times MSE_{MGBI})} \\ MSE_{RGBW} = \sqrt{(MSE_{ACR} \times MSE_{EDCR})} \end{gathered} \tag{6.1}$$

This results in the following values:

$$MSE_{Bayer} = 4.43$$
$$MSE_{RGBW} = 8.07 \qquad (6.2)$$
$$MSE_{Proposed} = 4.90$$

Equations (6.1) and (6.2) provide an explanation to the fact that the proposed algorithm ranks in the near median position (6th) in the overall MSE evaluation of Table 5.3. This issue of estimate data leads to inaccuracies in reconstruction, especially in the case of the ARRI image set that has a very large pixel count due to its resolution characteristics.

6.3 Full Colour Reconstruction and Object Fidelity Analysis

The main aim of any demosaicking process is to ensure that colour reconstruction from subsampled sensor data is effective and provides a true representation of the scene. This, however, is sometimes achieved at the expense of edge definition of an object in an image scene. A demosaicking algorithm may use a large number of descriptors to accurately interpolate missing colour information. However, the large number of descriptors may adversely affect object edge in a scene resulting in colour shifting or blurring. Some of these errors are shown in Figure E.7: (b) Colour Shifting (d) Blurring. In both cases, the colour is a true representation of the scene but the sharp edge definition is lost.

The converse is true; one may use fewer descriptors to maintain edge fidelity. However, at high frequency colour regions (such as changes between different objects), these descriptors are inadequate to provide a proper colour transition. This results in demosaicking errors such as Moiré and zipper effects also depicted in Figure E.7: (a) Zipper effect and (c) Moiré effect. Here the edges are quite sharp; however, either spurious colours are generated or granulation of the edge occurs.

Consequently, a balance must be observed and maintained by any demosaicking algorithm. This is particularly pertinent in images taken by low to medium resolution cameras. This is because they have a lower sub-sample data count and must rely more heavily on descriptor information.

In the full reconstruction analysis, the author selected the CPSNR to provide an indication of colour quality and the SSIM/FSIM$_C$ to provide edge quality indication.

6.3.1 Colour Reconstruction Analysis from CPSNR data

From the data set generated from the experimental simulation, the geometric mean CPSNR data was extracted and plotted in Figure 6.2. The proposed algorithm exhibited the best overall CPSNR value of

40.78 dB that exceeded by best performing Bayer algorithm in the test bed, that is ESFBI, by 0.53dB. It also exceeded the best performing RGBW algorithm that is the ACR method by 2.48 dB.

The author attributes the high performance over Bayer methods to:

i. the use of *gradient descriptors that operate in ordinal directions*
ii. the *encoding of white pixel (luminosity) information* through the Bayerisation process

The high performance over RGBW methods is attributed to the *increase in green colour sample data* through the Bayerisation process.

From the 6th-order polynomial curve fitted trend line provided in Figure 6.2, it is observed that as algorithm complexity increases in the Bayer based algorithms, there is a rise, plateauing and fall of the CPSNR values. This is indicative of a 'maximum threshold region' beyond which increasing descriptor complexity does not benefit demosaicking. By using ordinal directed descriptors coupled with the encoding of luminosity information; the proposed method exceeds this threshold and generates, on average, a more representative facsimile of the original scene.

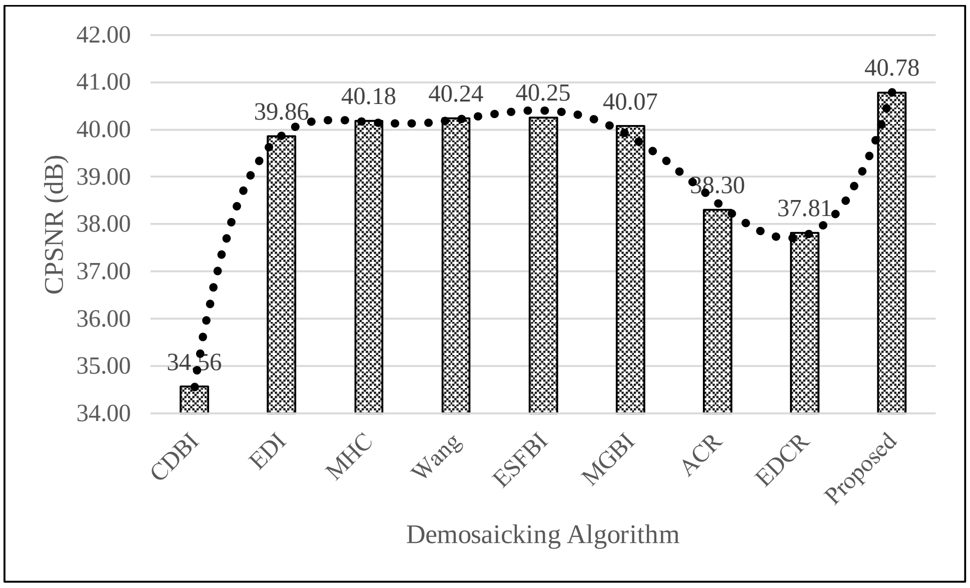


Figure 6.2 Graph of average geometric mean CPSNR values (in dB) over all test bed demosaicking algorithms in all selected image sets

Comparing the RGBW counterparts, an increase in descriptor complexity results in a reduction in CPSNR. The author believes this is due to the fact that the RGBW domain has a low sub-sample data count in the green plane. Therefore, increasing the descriptor complexity at a low sub-sample count leads to an increase in estimation errors rather than a reduction. This inference is strengthened by the fact that the more complex EDCR exhibits a higher MSE in Figure 6.1 than the simpler ACR method. The proposed method, however, increases its low green sub-sample count through the Bayerisation process and consequently operates as a Bayer algorithm. As such its increased complexity is not hampered in the same way as the EDCR method.

6.3.2 Fidelity of Object Reconstruction Analysis from SSIM and FSIM data

The SSIM and FSIM$_C$ data generated from the simulation is plotted in Figures 6.3 and 6.4 respectively. The proposed algorithm exhibits the best FSIM$_C$, of 0.9911 (rounded off to 0.991) followed by CDBI at 0.9909 and EDI at 0.9908 (both also rounded off to 0.991). For the purposes of analysis, the FSIM$_C$ is considered the same over all three algorithms. The proposed algorithm has an SSIM value of 0.980 which the second best performance value after the 0.982 value of the CDBI algorithm.

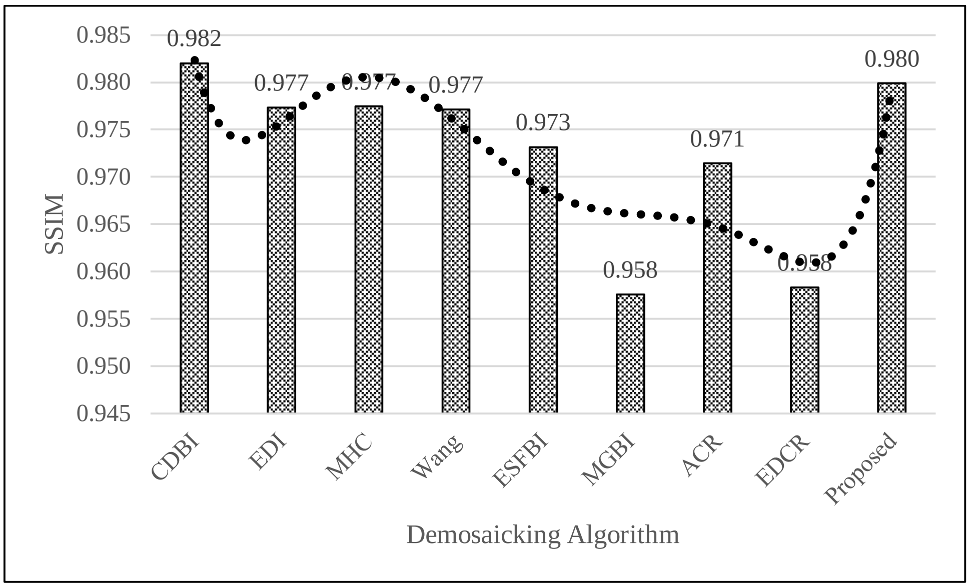


Figure 6.3 Graph of average geometric mean SSIM values over all test bed demosaicking algorithms in all selected image sets

The author attributes the high values of SSIM and FSIM$_C$ from:

i. the use of a *wholly ordinal-directed* descriptor generation system uses pentomino inspired paths
ii. the *non-symmetry of individual N, W and Z pentomino paths* with emphasis on shorter descriptor paths

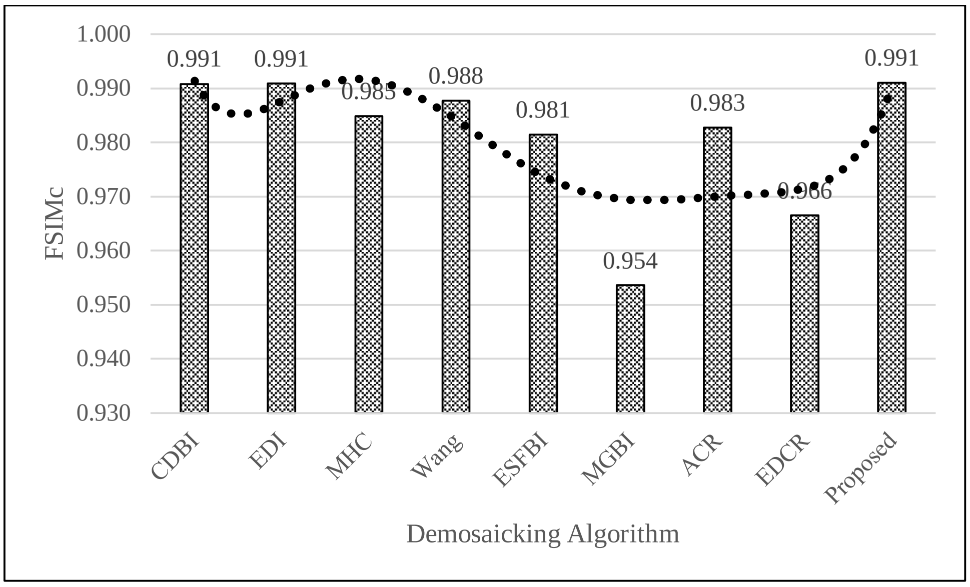


Figure 6.4 Graph of average geometric mean FSIM$_C$ values over all test bed demosaicking algorithms in all selected image sets

Observation of the trend lines and bar graph values of both graphs in Figures 6.3 and 6.4 reveals a similar trend over both the SSIM and FSIM metrics. This is to be expected as both metric are similar in definition and vary only in application size. In both the Bayer-based and RGBW-based comparison algorithms, it is observed that as the algorithm descriptor complexity increases, the SSIM and FSIM$_C$ decreases. EDI and CDBI perform well due to their simple interpolation descriptors.

The reason for this inverse relationship arises from the fact that all the algorithms in the test bed with the exception of the proposed method use a cardinal direction for establishing descriptors. An increased descriptor complexity results in the use of a larger pixel grid. Consider the different grid sizes shown in Figure 4.4, and the associated process of reconstruction. In a 7×7 grid, a Bayer-based algorithm has a pool of 16 pixels with which to generate a sufficient set of descriptors for interpolation. Increase to a 9×9 grid, the pool has grown substantially to 24 pixels.

This larger size occurs in a loosely packed grid and the introduction of the far outlier pixels, denoted O_i in equation 6.3, during descriptor generation results in an overall smoothening effect at the pixel of interest due the large distance traversed from the neighbour pixels, N_i. A larger distance often means larger path difference, d_{path}:

$$d_{path} = \sum_{i=1}^{n} |(N_i - O_i)| > 0 \tag{6.3}$$

A large path difference value can be interpreted as a higher likelihood of smoothening because of the inverse relationships descriptor paths have with their edges/gradients, as shown in equation (4.8). This smoothening results in loss of object edge information, lowering the SSIM and FSIM$_C$ values.

The proposed algorithm works solely on an ordinal directed environment. From Figure 4.3 and 4.4, it is noted that the tight packing ensures a smaller comparative distance is traversed by the descriptor paths generated. As such, the overall path difference value generated from equation (6.3) will be comparatively smaller that the cardinal driven algorithms of similar complexity. This in turn will allow for sharper edges and fidelity of object reconstruction overall.

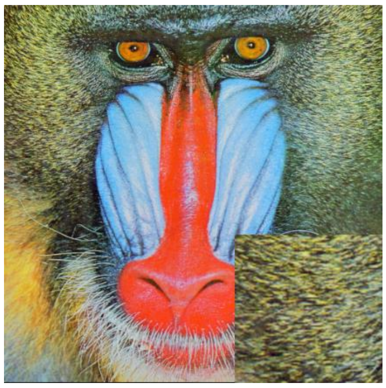
The additional property of the N, W and Z paths having a larger number of shorter paths than longer paths reinforces the edge sharpening property of the algorithm via equation (6.3). A large number of shorter paths is better than a small number of large paths due to the fact that small paths have a higher likelihood of producing near null values compared to larger paths, resulting in a smaller path difference.

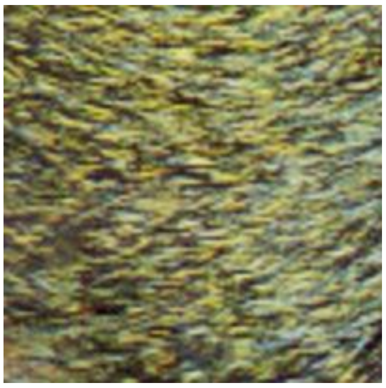
6.3.3 Reconstruction Analysis from Visual Inspection

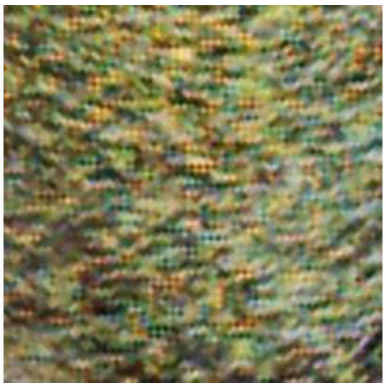
To augment the quantitative data established in Figures 6.2, 6.3 and 6.4, three images were selected from the 115 tested – each from a different image set. In each image, a region of interest (ROI) was selected after completing the demosaicking process and enlarged for observation of inconsistencies in image reconstruction. These regions of interest were also visually inspected to determine the test bed algorithms' resistance to demosaicking artefacts.

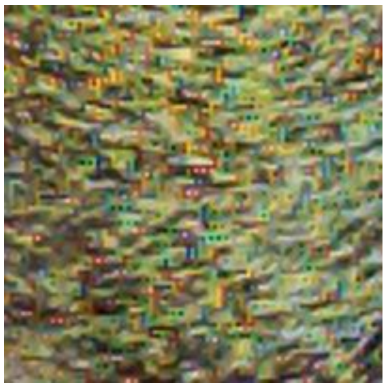
The first image under visual inspection was the *sipi_im11* or *Mandrill* image from the USC-SIPI database. The region of interest is the right hand side cheek of the mandrill. The image ROI contains fine detail regions due to the fur of the mandrill. From the visual data presented in Figure 6.5, color shifting is observed to occur appreciably in the CDBI, EDI, MHC and Wang methods and to a lesser degree in the ESFBI method. This shifting takes the form of small blue-orange or purple-green pixel blocks in the ROI. The ACR and EDCR methods are more resistant to color shifting. However, both these methods instead suffer from blurring effects due to the oversmoothening nature of the algorithms. In particular, the oversmoothening property of the EDCR algorithm begins to introduce a wash-out effect on the ROI.

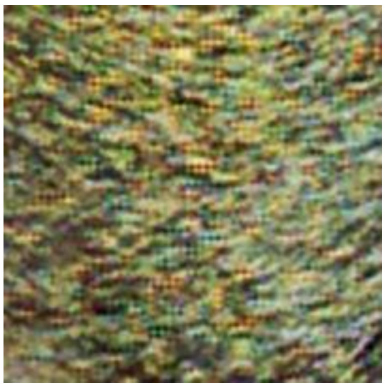
From the *sipi_im11* image inspection of Figure 6.5, only the MGBI and the proposed methods result in reconstructions of the ROI that exhibit no real demosaicking artefacts.

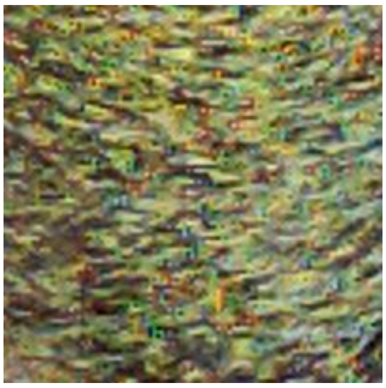

(a) Original image with enlarged section expanded for reference

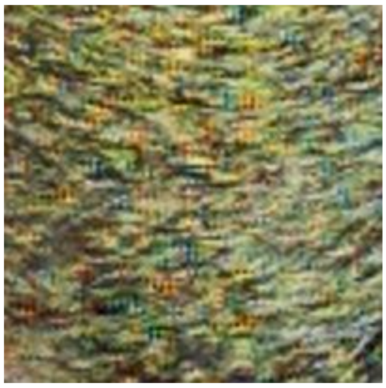

(b) Original expanded

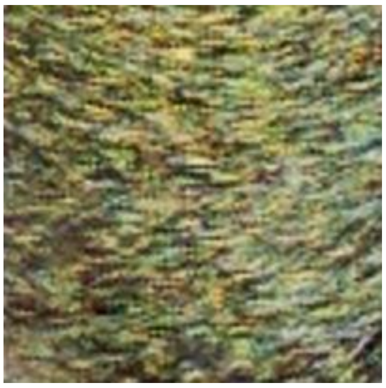

(c) CDBI expanded

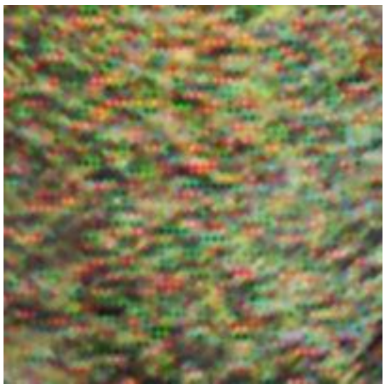

(d) EDI expanded

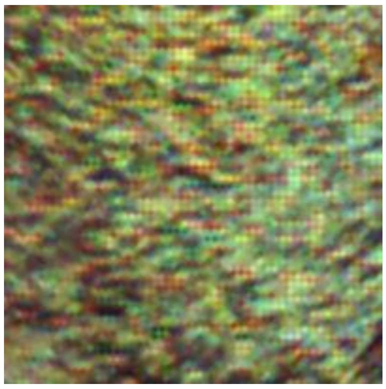

(e) MHC expanded


(f) Wang expanded


(g) ESFBI expanded


(h) MGBI expanded


(i) ACR expanded


(j) EDCR expanded

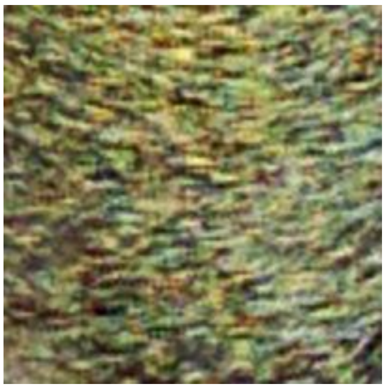

(k) Proposed expanded

Figure 6.5 A visual comparison of the right cheek section of the sipi_im11 image from the USC-SIPI Image Set over the different demosaicking schemes in the test bed

The second image selected for a qualitative visual inspection was the *kodim19* image, also referred to as the '*Lighthouse*'. It is a member of the Kodak Image Set. This particular image exhibits high frequency colour transition in the fence section of the image due to the narrow distance between individual fence posts. Enlarging the fence section near the coin-operated binoculars, in the right of the image, to act as the region of interest; the reconstruction results are given in Figure 6.6. It is noted that the CDBI, MHC and ESFBI methods suffer from discernable blue-orange Moiré patterns. The EDI and proposed methods suffer from the same artefact but to a lesser degree. The ACR and EDCR methods experience a yellow-purple Moiré effect by virtue of demosaicking in the RGBW domain. The Moiré pattern in the EDCR is so severe that the bands constituting the artefact are clearly observed. The MGBI and Wang methods suffer the least from the colour Moiré effect. However, a closer inspection reveals that this resistance comes at the expense of a granulation of the fence edges. This granulation is also observed in the EDI reconstruction of the ROI. The proposed method also suffers from colour Moiré in the fence post region. This is because the posts themselves are vertically placed reducing the efficacy of the ordinal gradients selected.

The third image selected from the various images was the *mcm11* image from the McMaster-IMAX Image Set. This image is of a striped tea cloth towel surrounded by a batch of freshly washed assorted vegetables. Selecting the ROI are the tea cloth towel and the nearby green leaves, an enlargement of this region was extracted from the original image. The reconstruction of the image was done with all the demosaicking algorithms of the test bed, the ROI extracted and the results are shown in Figure 6.7. It is observed that the CDBI, MHC, ESFBI and MGBI algorithms result in a ROI reconstruction with 'jaggies' present. This demosaicking artefact is observed primarily on the leaves section in the region of interest. The EDI and Wang methods suffer from the introduction of black spots in the leaves sections and a smotthening of the leaf edges. The ACR and EDCR reconstructions suffer from excessive blurring of the leaves and towel sections of the ROI. The EDCR method also introduces color shifts in the bands of the towel.

In the *mcm11* image ROI reconstructions of Figure 6.7, the proposed method exhibits the closest approximation to the reference original.

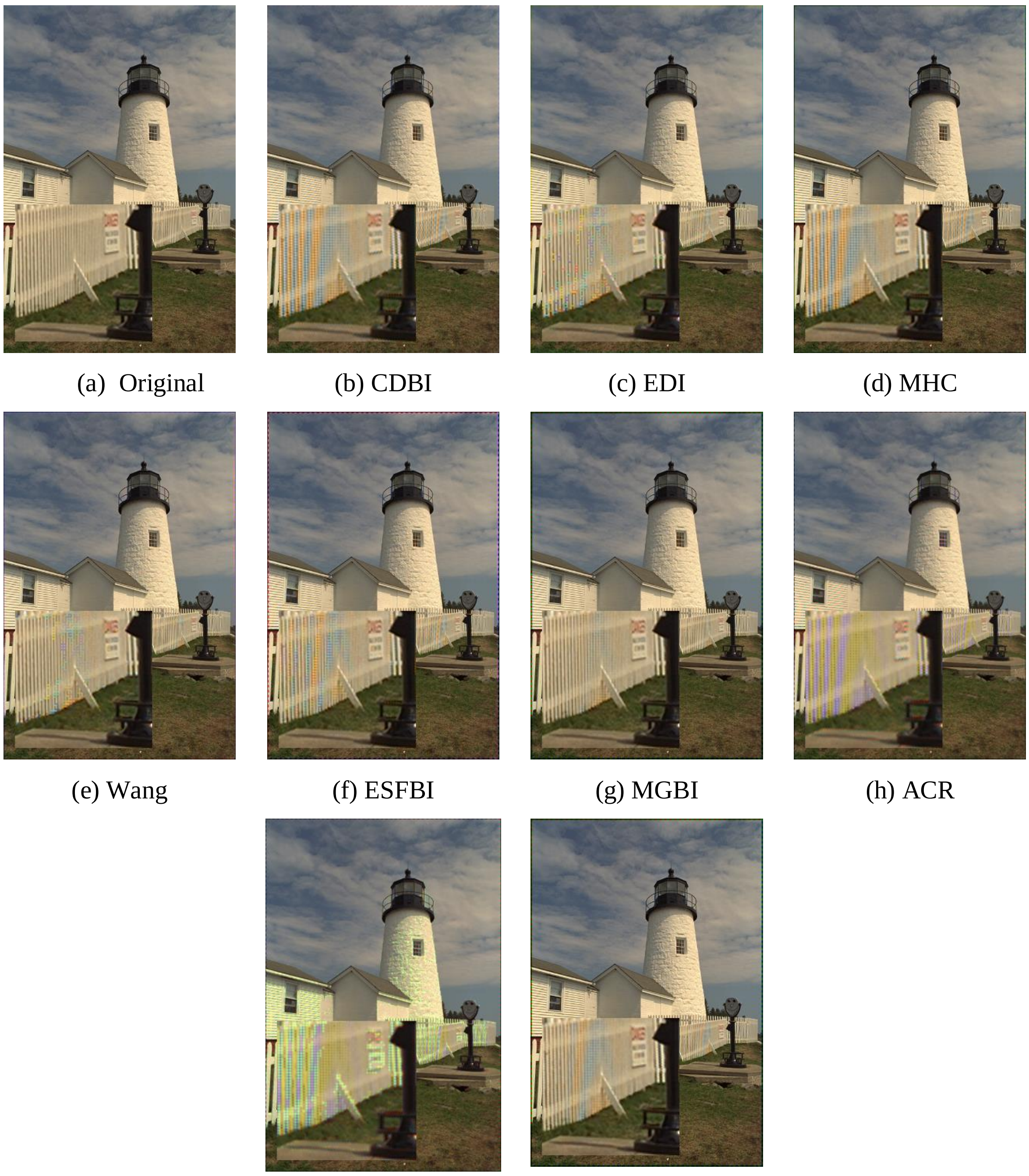
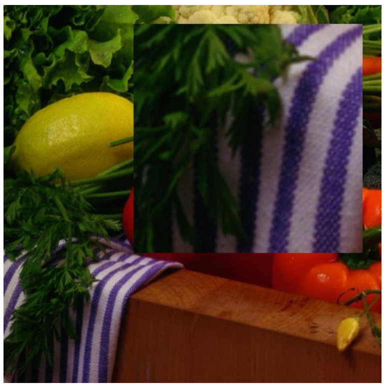
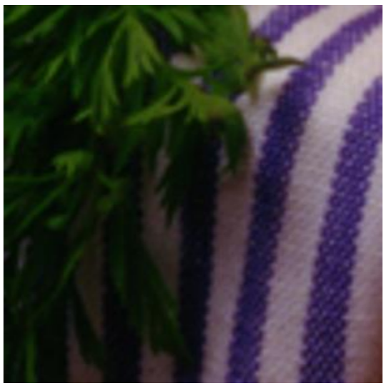
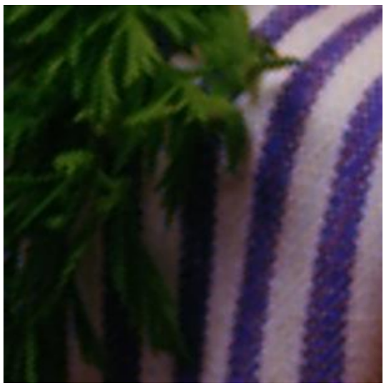
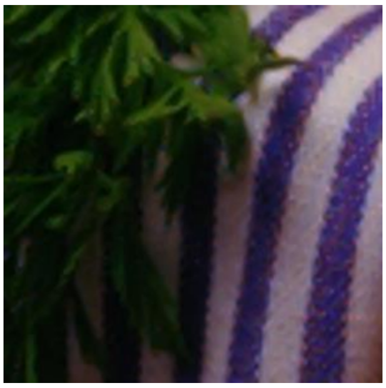


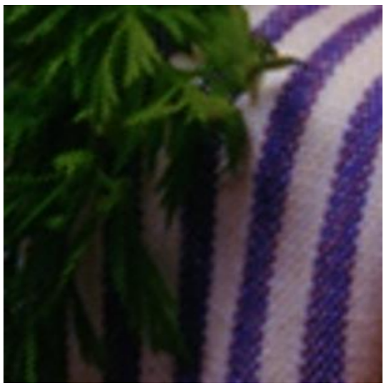
Figure 6.6 A visual comparison of the fence section of the kodim19 image from the Kodak Image Set over the different demosaicking schemes in the test bed

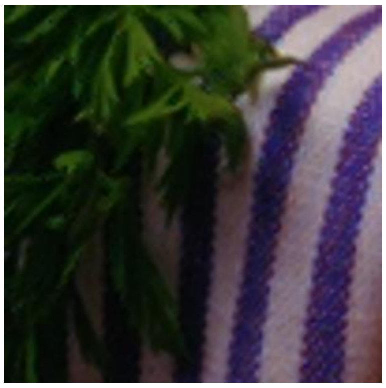

(a) Original image with enlarged section expanded for reference

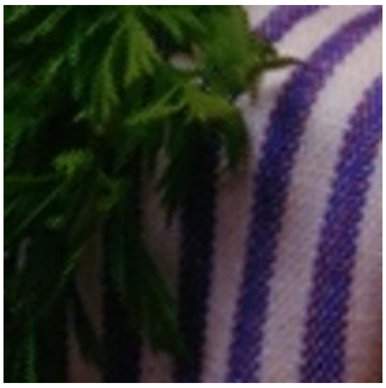

(b) Original expanded

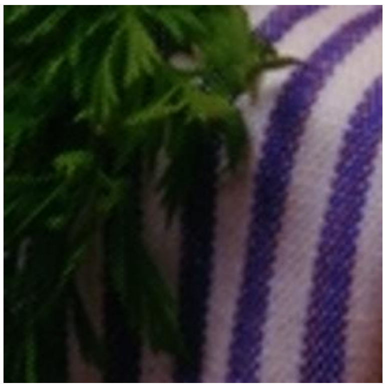

(c) CDBI expanded

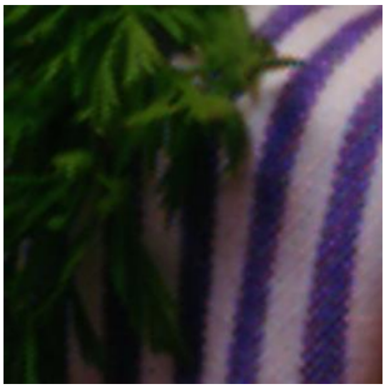

(d) EDI expanded

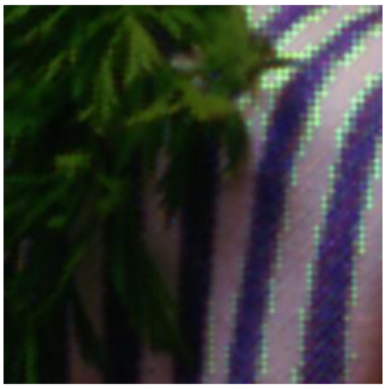

(e) MHC expanded


(f) Wang expanded


(g) ESFBI expanded


(h) MGBI exanded


(i) ACR expanded


(j) EDCR expanded

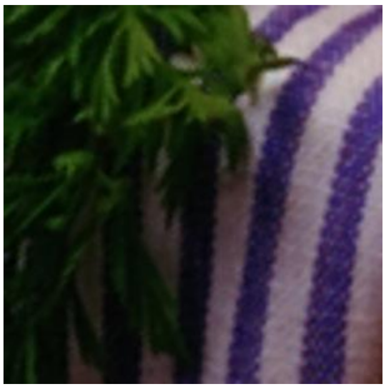

(k) Proposed expanded

Figure 6.7 A visual comparison showing the expanded towel cloth section of the mcm11 image from the McMaster-IMAX Image Set over the different demosaicking schemes in the test bed

From the brief visual inspection, it was noted that the proposed algorithm was largely invariant to most of the documented demosaicking effects namely colour shifting, blurring and granulation jaggies. However, it was found to be somewhat weak to colour Moiré in high frequency transition regions. The proposed algorithm, in line with its design, also ensured that object fidelity was not sacrificed for colour accuracy. This was observed visually in Figure 6.6 when comparing the proposed method to the MGBI method.

6.4 Supplementary Observations from the Different Image Sets

The performance rank values for the proposed algorithm over all image sets, established from simulation, is provided in Table 6.2. In all but one of the image sets, the proposed algorithm ranks in the top three best performing algorithms over CPSNR, SSIM and FSIM$_C$ assessment metrics.

Table 6.2 Proposed algorithm's ranking over CPSNR, SSIM and FSIM$_C$ metrics

	CPSNR	**SSIM**	**FSIM$_C$**
USC-SIPI(16)	1	1	2
Kodak (24)	2	2	3
McMaster-IMAX (18)	2	3	1
Condat (30)	1	3	3
ARRI (12)	3	5	3
Custom (15)	3	2	2
Average (115)	**1**	2	**1**

From the conceptual framework section, in particular Table 3.2, each of the image sets had been chosen for a particular property that is exhibited uniformly in all of its images. From the resolution designations set up in the framework, it is observed that the proposed algorithm had a very robust CPSNR over the low resolution image sets (USC-SIPI, Kodak, McMaster-IMAX and Condat) ranking in the top two algorithms. In the medium image set (Custom) and high resolution image set (ARRI), the proposed algorithm had the third ranking CPSNR value.

In the SSIM performance, the proposed algorithm achieves a robust edge preservation profile. This is because it obtains top three level performance in both the low and medium resolution sets; the only exception overall is in the high resolution ARRI image set where it achieves a median rank. Finally, in the FSIM$_C$ affirms the robust nature of the proposed algorithm where, on average, it is the best performing demosaicking algorithm.

From the average information over all image sets, the proposed algorithm is found to be invariant to the properties of various image sets such as object number and types in a scene, light intensity variations and saturation effects.

7 CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK

7.1 Concluding Remarks

A novel heuristic-based localised area demosaicking algorithm for panchromatic colour filter arrays has been proposed, outlined and developed in this work. It is a two stage process that operates by first using a Bayerisation process to convert RGBW panchromatic CFA sensor data to an equivalent Bayer representation. The Bayerisation process allows for the capture and encoding of luminance information from the RGBW domain into the Bayer domain for use in the demosaicking. The proposed algorithm's second stage then employs a unique ordinal-directed gradient interpolation mechanism, founded on pentomino constructs, to generate a sufficient number of path descriptors that allow for robust colour reconstruction without compromising on object edge information. The gradient-based interpolation process was strengthened by additional novel concepts such as the quincuncial exploitation of the ordinal nature of the green colour plane and the introduction of corrective variable plane (k_2 *and* k_3) and path (ε) terms.

In line with the study's objectives of operation for low and medium resolution cameras, a conceptual framework was established yielding an appropriate grouping of image sets. Four of the six image sets were of low resolution, one of medium resolution and one of high resolution. The image sets were selected from both standard and custom and each set provided an opportunity to test a different image property.

To test the efficacy of the proposed algorithm in a structured manner, a robust test bed of well documented and state-of-the art demosaicking algorithms both in the Bayer and panchromatic domain was created. Together with the proposed algorithm, the entire test bed was passed through the aforementioned image sets and a quantitative assessment (through four standard image assessment metrics) and qualitative assessment (through visual inspection) was made. It was determined that the proposed algorithm, from average metric data achieved a high colour reconstruction with maintaining object edge acuity. From a visual inspection, it was found the proposed algorithm was mostly resistant to the expected visual artefacts reviewed in literature.

7.2 Note on Publications

This study on demosaicking algorithm design resulted in the publication of the following internationally peer-reviewed papers:

i. K. Wachira, E. Mwangi, and G. Jeon, “An Ordinal Direction Driven Gradient-Based RGBW CFA Demosaicking Technique Using A Bayerisation Process and Polyomino Theory,” in IEEE AFRICON2017, 2017, pp. 1–6. DOI:10.1109/AFRCON.2017.8095483 [95].

ii. K. Wachira, E. Mwangi, and G. Jeon, “A pentomino-based path inspired demosaicking technique for the bayer color filter array,” in IEEE AFRICON2015, 2015, pp. 1–5. DOI:10.1109/AFRCON.2015.7331959 [91].

iii. K. Wachira, “Corrective term usage in the improvement of gradient-based bayer CFA demosaicking algorithms,” in IEEE EUROCON 2015 - International Conference on Computer as a Tool (EUROCON), 2015, pp. 1–6. DOI:10.1109/EUROCON.2015.7313800 [138].

iv. K. Wachira and E. Mwangi, “A multi-variate weighted interpolation technique with local polling for bayer CFA demosaicking,” in 2015 International Conference on Information and Communication Technology Research (ICTRC), 2015, pp. 76–79. DOI:10.1109/ICTRC.2015.7156425 [90].

v. K. Wachira and E. Mwangi, “A cardinal-direction quincunx based interpolation technique with non-uniform inter-plane weighting for bayer CFA demosaicking,” in 2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015, pp. 1–5. DOI:10.1109/SPICES.2015.7091363 [89].

7.3 Further Work

Consequently, it is the opinion of the author, that the main and specific objectives of this study were resolved. However, from the analysis of the proposed algorithm, several new questions arose such as:

i. Why does the proposed algorithm’s performance deteriorate at higher resolutions despite the advantages of tighter pixel packing in the ordinal directions?

ii. Can the Bayerisation process be improved by employing a filter, h_{alt}, that uses dynamic rather than static coefficients?

iii. The panchromatic arrangement considered was the RGBW domain. Are the qualities of the proposed algorithm transferable to other panchromatic arrangements while maintaining the same level of image reconstruction performance?

iv. Through both simulation and real world implementation, a complexity vs. speed analysis can be done to compare the proposed algorithm with other state-of-the art demosaicking techniques.

v. How would compressive sensing theory be employed in demosaicking and would the results generated be significantly higher than established techniques? Secondly, how would compressive sensing affect the commonly observed demosaicking artefacts?

These questions are the author’s recommendations for future work in this area.

REFERENCES

[1] P. Hall, *Indonesia, Malaysia & Singapore Handbook*. Trade & Trade & Travel Publications ; New York, NY, 1993.

[2] U. W. H. Centre, “Cueva de las Manos, Río Pinturas,” *UNESCO World Heritage Centre*. [Online]. Available: http://whc.unesco.org/en/list/936/. [Accessed: 28-Mar-2017].

[3] J. Mott, “A Journey to the Oldest Cave Paintings in the World,” *Smithsonian*, vol. 46, no. 10, Jan-2016.

[4] C. Sagan, L. S. Sagan, and F. Drake, “A Message from Earth,” *Science*, vol. 175, no. 4024, pp. 881–884, Feb. 1972.

[5] D. Goldsmith and T. Owen, *The Search for Life in the Universe*, 3rd edition. Sausalito, Calif: University Science Books, 2001.

[6] “Number of mobile phone users worldwide 2013-2019,” *Statista*. [Online]. Available: https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/. [Accessed: 27-Mar-2017].

[7] B. Sanou, “ICTFacts & Figures: The world in 2015,” ITU, ITU Telecommunication Development Bureau, May 2015.

[8] M. Tumbleson, “19 Text Messaging Facts | Teckst | Incredible Textable Tech,” *Texting for Customer Service*, 03-Oct-2015. [Online]. Available: https://teckst.com/19-text-messaging-stats-that-will-blow-your-mind/. [Accessed: 27-Mar-2017].

[9] J. Schneider, “Infographic: There Will Be One Trillion Photos Taken in 2015,” *Resource Magazine Online*. [Online]. Available: http://resourcemagonline.com/2014/12/infographic-there-will-be-one-trillion-photos-taken-in-2015/45332/. [Accessed: 27-Mar-2017].

[10] H. Bajaj and R. Jindal, “Thinking beyond WhatsApp,” in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2015, pp. 1443–1447.

[11] J. Dunn, “Here’s how slowly Twitter has grown compared to Facebook, Instagram, and Snapchat,” *Business Insider*. [Online]. Available: http://www.businessinsider.com/twitter-vs-facebook-snapchat-user-growth-chart-2017-2. [Accessed: 27-Mar-2017].

[12] “Our Story,” *Instagram*, 29-Nov-2016. [Online]. Available: https://instagram-press.com/our-story/. [Accessed: 27-Mar-2017].

[13] M. Minervini, H. Scharr, and S. A. Tsaftaris, “Image Analysis: The New Bottleneck in Plant Phenotyping [Applications Corner],” *IEEE Signal Process. Mag.*, vol. 32, no. 4, pp. 126–131, Jul. 2015.

[14] V. Menezes, V. Patchava, and M. S. D. Gupta, “Surveillance and monitoring system using Raspberry Pi and SimpleCV,” in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 2015, pp. 1276–1278.

[15] H.-Q. Nguyen, T. T. K. Loan, B. D. Mao, and E.-N. Huh, “Low cost real-time system monitoring using Raspberry Pi,” in *2015 Seventh International Conference on Ubiquitous and Future Networks*, 2015, pp. 857–859.

[16] R. W. G. Hunt, *The Reproduction of Colour*. John Wiley & Sons, 2005.

[17] D. H. Hubel and T. N. Wiesel, *Brain and Visual Perception: The Story of a 25-Year Collaboration*, 1 edition. New York, N.Y: Oxford University Press, 2004.

[18] K. R. Gegenfurtner, M. Bloj, and M. Toscani, “The many colours of ‘the dress,’” *Curr. Biol.*, vol. 25, no. 13, pp. R543–R544, Jun. 2015.

[19] R. Lafer-Sousa, K. L. Hermann, and B. R. Conway, “Striking individual differences in color perception uncovered by ‘the dress’ photograph,” *Curr. Biol.*, vol. 25, no. 13, pp. R545–R546, Jun. 2015.

[20] A. D. Winkler, L. Spillmann, J. S. Werner, and M. A. Webster, “Asymmetries in blue–yellow color perception and in the color of ‘the dress,’” *Curr. Biol.*, vol. 25, no. 13, pp. R547–R548, Jun. 2015.

[21] D. H. Brainard and A. C. Hurlbert, “Colour Vision: Understanding #TheDress,” *Curr. Biol.*, vol. 25, no. 13, pp. R551–R554, Jun. 2015.

[22] "Gartner Says By 2018, More Than 50 Percent of Users Will Use a Tablet or Smartphone First for All Online Activities." [Online]. Available: http://www.gartner.com/newsroom/id/2939217. [Accessed: 27-Mar-2017].

[23] J. Li, C. Bai, Z. Lin, and J. Yu, "Automatic Design of High-Sensitivity Color Filter Arrays With Panchromatic Pixels," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 870–883, Feb. 2017.

[24] S. K. Shevell, *The Science of Color*. Elsevier, 2003.

[25] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson Education, 2011.

[26] G. Wald, "The Photoreceptor Process in Vision*," *Am. J. Ophthalmol.*, vol. 40, no. 5, pp. 18–41, Nov. 1955.

[27] B. H. Zeavin and G. Wald, "Rod and Cone Vision in Retinitis Pigmentosa*," *Am. J. Ophthalmol.*, vol. 42, no. 4, pp. 253–269, Oct. 1956.

[28] Alhazen and A. I. Sabra, *The optics of Ibn al-Haytham: Books I-III : on direct vision*. Warburg Institute, University of London, 1989.

[29] K. Keller *et al.*, "Photography," in *Ullmann's Encyclopedia of Industrial Chemistry*, Wiley-VCH Verlag GmbH & Co. KGaA, 2000.

[30] N. Waltham, "CCD and CMOS sensors," in *Observing Photons in Space*, M. C. E. Huber, A. Pauluhn, J. L. Culhane, J. G. Timothy, K. Wilhelm, and A. Zehnder, Eds. Springer New York, 2013, pp. 423–442.

[31] G. C. Holst and T. S. Lomheim, *CMOS/CCD Sensors and Camera Systems*, 2 edition. Winter Park, FL : Bellingham, Wash: SPIE--The International Society for Optical Engineering, 2011.

[32] R. Ramanath, W. E. Snyder, Y. Yoo, and M. S. Drew, "Color Image Processing Pipeline," *IEEE*, vol. 22, no. 1, pp. 34–43, Jan. 2005.

[33] R. Lukac, *Single-Sensor Imaging: Methods and Applications for Digital Cameras*. CRC Press, 2008.

[34] R. Lukac, *Computational Photography: Methods and Applications*. CRC Press, 2010.

[35] J. Nakamura, *Image Sensors and Signal Processing for Digital Still Cameras*. CRC Press, 2016.

[36] J. Pach and P. K. Agarwal, *Combinatorial Geometry*, 1 edition. New York: Wiley-Interscience, 1995.

[37] H. H. Crapo and G.-C. Rota, *On The Foundations of Combinatorial Theory: Combinatorial Geometries*, Prelim. ed edition. Cambridge, Mass: The MIT Press, 1970.

[38] J. A. McKee, "Pixel cells in a honeycomb arrangement," US7511323 B2, 31-Mar-2009.

[39] B. E. Bayer, "Color Imaging Array," 3971065, Jul-1976.

[40] S. Yamanaka, "Solid state color camera," US4054906 A, 18-Oct-1977.

[41] T. Watanabe and S. Miyatake, "Color imaging array and color imaging device," US4500914 A, 19-Feb-1985.

[42] J. E. Roddy, R. J. Zolla, N. A. Blish, and L. Horvath, "Four color image sensing apparatus," US7057654 B2, 06-Jun-2006.

[43] A. Morimura and H. Tanaka, "Color solid-state imager with color filter having an overlapping segmented filter arrangement," US4630106 A, 16-Dec-1986.

[44] J. F. Hamilton, J. E. Adams, and D. M. Orlicki, "Particular pattern of pixels for a color filter array which is used to derive luminance and chrominance values," US6330029 B1, 11-Dec-2001.

[45] J. J. Bean, "Cyan-magenta-yellow-blue color filter array," US6628331 B1, 30-Sep-2003.

[46] K. Hirakawa and P. J. Wolfe, "Spatio-Spectral Color Filter Array Design for Optimal Image Recovery," *IEEE*, vol. 17, no. 10, pp. 1876–1890, Oct. 2008.

[47] P. L. P. Dillon, "Color imaging array," US4047203 A, 06-Sep-1977.

[48] T. Yamagami, T. Sasaki, and A. Suga, "Image signal processing apparatus having a color filter with offset luminance filter elements," US5323233 A, 21-Jun-1994.

[49] E. J. Bawolek, Z.-F. Li, and R. D. Smith, "Magenta-white-yellow (MWY) color system for digital image sensor applications," US5914749 A, 22-Jun-1999.

[50] E. B. Gindele and A. C. Gallagher, "Sparsely sampled image sensing device with color and luminance photosites," US6476865 B1, 05-Nov-2002.

[51] T. Sugiyama, "Image-capturing apparatus," US20050231618 A1, 20-Oct-2005.

[52] G. Luo, "A novel color filter array with 75% transparent elements," 2007, vol. 6502, p. 65020T–65020T–8.

[53] N. Mansurov, “What is Spherical Aberration?,” *Photography Life*. [Online]. Available: https://photographylife.com/what-is-spherical-aberration/. [Accessed: 06-Apr-2017].

[54] S. Borodin, “Lens Aberrations in Photography: Good or Bad?” [Online]. Available: http://allphotolenses.com/articles/item/c_28.html. [Accessed: 06-Apr-2017].

[55] D. H. Marimont and B. A. Wandell, “Matching color images: the effects of axial chromatic aberration,” *JOSA A*, vol. 11, no. 12, pp. 3113–3122, Dec. 1994.

[56] “Achromats Information | Engineering360.” [Online]. Available: http://www.globalspec.com/learnmore/optics_optical_components/optical_components/achromats. [Accessed: 06-Apr-2017].

[57] M. Hullin, E. Eisemann, H.-P. Seidel, and S. Lee, “Physically-based Real-time Lens Flare Rendering,” in *ACM SIGGRAPH 2011 Papers*, New York, NY, USA, 2011, p. 108:1–108:10.

[58] “Digital Camera Image Noise: Concept and Types.” [Online]. Available: http://www.cambridgeincolour.com/tutorials/image-noise.htm. [Accessed: 06-Apr-2017].

[59] D. M. John and A. Thomas, “Combined denoising and demosaicing of CFA images,” in *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, 2015, pp. 1–6.

[60] X. Zhang, M.-T. Sun, L. Fang, and O. C. Au, “Joint Denoising and demosaicking of noisy CFA images based on inter-color correlation,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 5784–5788.

[61] I. Amidror, *The Theory of the Moiré Phenomenon*. Springer Science & Business Media, 2012.

[62] L. Condat and S. Mosaddegh, “Joint demosaicking and denoising by total variation minimization,” in *2012 19th IEEE International Conference on Image Processing (ICIP)*, 2012, pp. 2781–2784.

[63] L. Condat, “A simple, fast and efficient approach to denoisaicking: Joint demosaicking and denoising,” in *2010 17th IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 905–908.

[64] E. Reuss, “Beyond the Limits of Visual Acuity: The Real Reason for 4K and 8K Image Resolution,” *SMPTE Motion Imaging J.*, vol. 126, no. 2, pp. 33–39, Mar. 2017.

[65] R. Lukac, *Perceptual Digital Imaging: Methods and Applications*. CRC Press, 2012.

[66] S. Fang, Q. Shi, and Y. Cao, “Adaptive removal of real noise from a single image,” *J. Electron. Imaging*, vol. 22, no. 3, pp. 033014–033014, 2013.

[67] X. Li, B. Gunturk, and L. Zhang, “Image demosaicing: a systematic survey,” 2008, vol. 6822, p. 68221J–68221J–15.

[68] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, “Demosaicking: Color Filter Array Interpolation,” *IEEE*, vol. 22, no. 1, pp. 44–54, Jan. 2005.

[69] T. Banning Jr., “Color television and the like,” US2683769 A, 13-Jul-1954.

[70] W. Yu, “Adaptive cubic convolution interpolation and sequential filtering for color demosaicing of Bayer pattern image sensors,” in *Proc. SPIE*, 2005, vol. 5909, pp. 5–11.

[71] S. Randhawa and J. S. J. Li, “Adaptive Order Spline Interpolation for Edge-Preserving Colour Filter Array Demosaicking,” in *2011 International Conference on Digital Image Computing: Techniques and Applications*, 2011, pp. 666–671.

[72] D. R. Cok, “Signal Processing Method and Apparatus For Producing Interpolated Chrominance Values In A Sampled Color Signal,” 4642678, Feb-1987.

[73] R. Kimmel, “Demosaicing: Image Reconstruction from color CCD Samples,” *IEEE*, vol. 8, no. 9, pp. 1221–1228, Sep. 1999.

[74] K.-H. Chung and Y.-H. Chan, “An Adaptive Color Filter Array Interpolation Algorithm for Digital Camera,” in *2006 IEEE International Conference on Image Processing*, 2006, pp. 2697–2700.

[75] K. H. Chung and Y. H. Chan, “Color Demosaicing Using Variance of Color Differences,” *IEEE*, vol. 15, no. 10, pp. 2944–2955, Oct. 2006.

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

[77] X. Wang, W. Lin, and P. Xue, “Demosaicing with improved edge direction detection,” in *2005 IEEE International Symposium on Circuits and Systems*, 2005, p. 2048–2051 Vol. 3.

[78] R. Lukac, K. N. Plataniotis, D. Hatzinakos, and M. Aleksic, “A novel cost effective demosaicing approach,” *IEEE Trans. Consum. Electron.*, vol. 50, no. 1, pp. 256–261, Feb. 2004.

[79] H. S. Malvar, L. He, and R. Cutler, "High-quality linear interpolation for demosaicing of Bayer-patterned color images," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004, vol. 3, pp. 485–488.

[80] D. Su and P. Willis, "Demosaicing of color images using pixel level data-dependent triangulation," in *Proceedings of Theory and Practice of Computer Graphics, 2003.*, 2003, pp. 16–23.

[81] W. Jin, "Improved Color Interpolation Method Based On Bayer Image," in *Proc. SPIE 8420, 6th International Symposium on Advanced Optical Manufacturing and Testing Technologies*, 2012.

[82] H.-A. Chang and H. Chen, "Directionally weighted color interpolation for digital cameras," in *2005 IEEE International Symposium on Circuits and Systems*, 2005, p. 6284–6287 Vol. 6.

[83] I. Pekkucuksen and Y. Altunbasak, "Edge Strength Filter Based Color Filter Array Interpolation," *IEEE*, vol. 21, no. 1, pp. 393–397, Jan. 2012.

[84] C.-Y. Su and Y.-H. Chen, "Low-complexity demosaicing via multiscale gradients," 2014, vol. 9069, pp. 906910-906910–5.

[85] D.-C. Sung and H.-W. Tsao, "A Gradient Based Edge Sensing Scheme for Color Filter Array Demosaicking," in *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, Tokyo, Japan, 2013.

[86] I. Pekkucuksen and Y. Altunbasak, "Multiscale Gradients-Based Color Filter Array Interpolation," *IEEE*, vol. 22, no. 1, pp. 157–165, Jan. 2013.

[87] I. Pekkucuksen and Y. Altunbasak, "Gradient based threshold free color filter array interpolation," in *2010 17th IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 137–140.

[88] K. L. Chung, W. J. Yang, W. M. Yan, and C. C. Wang, "Demosaicing of Color Filter Array Captured Images Using Gradient Edge Detection Masks and Adaptive Heterogeneity-Projection," *IEEE Trans. Image Process.*, vol. 17, no. 12, pp. 2356–2367, Dec. 2008.

[89] **K. Wachira and E. Mwangi, "A cardinal-direction quincunx based interpolation technique with non-uniform inter-plane weighting for bayer CFA demosaicking," in *Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015 IEEE International Conference on*, 2015, pp. 1–5.**

[90] **K. Wachira and E. Mwangi, "A multi-variate weighted interpolation technique with local polling for bayer CFA demosaicking," in *Information and Communication Technology Research (ICTRC), 2015 International Conference on*, 2015, pp. 76–79.**

[91] **K. Wachira, E. Mwangi, and G. Jeon, "A pentomino-based path inspired demosaicking technique for the bayer color filter array," in *AFRICON, 2015*, 2015, pp. 1–5.**

[92] Y. Monno, S. Kikuchi, M. Tanaka, and M. Okutomi, "A Practical One-Shot Multispectral Imaging System Using a Single Image Sensor," *IEEE Trans. Image Process.*, vol. 24, no. 10, pp. 3048–3059, Oct. 2015.

[93] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Minimized-Laplacian Residual Interpolation for Color Image Demosaicking," in *Proc. SPIE-IS&T 9023, Digital Photography X*, 2014.

[94] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Residual interpolation for color image demosaicking," in *2013 20th IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 2304–2308.

[95] **K. Wachira, E. Mwangi, and G. Jeon, "An Ordinal Direction Driven Gradient-Based RGBW CFA Demosaicking Technique Using A Bayerisation Process and Polyomino Theory," in *IEEE AFRICON2017 Proceedings*, Cape Town, South Africa, 2017, pp. 1–6.**

[96] X. Chen, G. Jeon, J. Jeong, and L. He, "Multi-Directional Weighted Interpolation and Refinement Method for Bayer Pattern CFA Demosaicking," *IEEE*, 2014.

[97] L. Wang and G. Jeon, "Bayer Pattern CFA Demosaicking Based on Multi-Directional Weighted Interpolation and Guided Filter," *Signal Process. Lett. IEEE*, vol. 22, no. 11, pp. 2083–2087, Nov. 2015.

[98] T. y Jung, S. Yang, and J. Jeong, "Multi-directional Demosaicing for Digital Still Cameras," in *2009 WRI World Congress on Computer Science and Information Engineering*, 2009, vol. 7, pp. 374–378.

[99] D. Alleysson, S. Süsstrunk, and J. Hérault, "Linear Demosaicing Inspired By The human Visual System," *IEEE*, vol. 14, no. 4, pp. 1–12, Apr. 2005.

[100] E. Dubois, "Frequency-domain methods for demosaicking of Bayer-sampled color images," *IEEE Signal Process. Lett.*, vol. 12, no. 12, pp. 847–850, Dec. 2005.

[101] W. Tang, O. C. Au, Y. Yang, X. Wen, and L. Fang, “Frequency selection and merging with universal matrices for color filter array demosaicking,” in *2009 IEEE International Symposium on Circuits and Systems*, 2009, pp. 2369–2372.

[102] E. Dubois, “Filter Design for Adaptive Frequency-Domain Bayer Demosaicking,” in *2006 International Conference on Image Processing*, 2006, pp. 2705–2708.

[103] R. W. B. Kolta, H. A. Aly, and W. Fakhr, “A hybrid demosaicking algorithm using frequency domain and wavelet methods,” in *2011 International Conference on Image Information Processing (ICIIP)*, 2011, pp. 1–6.

[104] D. Zhang, X. Wu, and D. Zhang, “Color Reproduction From Noisy CFA Data of Single Sensor Digital Cameras,” *IEEE Trans. Image Process.*, vol. 16, no. 9, pp. 2184–2197, Sep. 2007.

[105] T. Komatsu and T. Saito, “Sharpening-demosaicing with the shift-invariant Haar wavelet transform,” in *2009 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, 2009, pp. 252–255.

[106] H. İlbeği and A. C. Gürbüz, “Demosaicking with compressive sensing,” in *2012 20th Signal Processing and Communications Applications Conference (SIU)*, 2012, pp. 1–4.

[107] T. Singh and M. Singh, “Disregarding spectral overlap #x2014; A unified approach for demosaicking, compressive sensing and color filter array design,” in *2011 18th IEEE International Conference on Image Processing*, 2011, pp. 3161–3164.

[108] K. Hirakawa and T. W. Parks, “Joint demosaicing and denoising,” *IEEE Trans. Image Process.*, vol. 15, no. 8, pp. 2146–2157, Aug. 2006.

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

[110] X. Wu, S. Tang, L. Huang, W. Shao, P. Liu, and Z. Wei, “Robust color demosaicking via vectorial hessian frobenius norm regularization,” in *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*, 2016, pp. 161–165.

[111] C. Hu, L. Cheng, and Y. M. Lu, “Graph-based regularization for color image demosaicking,” in *2012 19th IEEE International Conference on Image Processing*, 2012, pp. 2769–2772.

[112] T. Ma and S. J. Reeves, “An iterative regularization approach for Color Filter Array image restoration,” in *2011 IEEE International Conference on Industrial Technology (ICIT)*, 2011, pp. 332–335.

[113] D. Menon and G. Calvagno, “Regularization Approaches to Demosaicking,” *IEEE Trans. Image Process.*, vol. 18, no. 10, pp. 2209–2220, Oct. 2009.

[114] O. A. Omer and T. Tanaka, “Image demosaicking based on chrominance regularization with region-adaptive weights,” in *2007 6th International Conference on Information, Communications Signal Processing*, 2007, pp. 1–5.

[115] C. K. M. Yuk, O. C. Au, R. Y. M. Li, and S. Y. Lam, “Soft-Decision Color Demosaicking with Direction Vector Selection,” in *2007 IEEE 9th Workshop on Multimedia Signal Processing*, 2007, pp. 449–452.

[116] R. Lukac and K. N. Plataniotis, “Vector Concepts-Based Spectral Modelling,” in *2006 Canadian Conference on Electrical and Computer Engineering*, 2006, pp. 2009–2012.

[117] R. Lukac, B. Smolka, K. Martin, K. N. Plataniotis, and A. N. Venetsanopoulos, “Vector filtering for color imaging,” *IEEE Signal Process. Mag.*, vol. 22, no. 1, pp. 74–86, Jan. 2005.

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

[119] H. J. Trussell, E. Saber, and M. Vrhel, “Color image processing [basics and special issue overview],” *IEEE Signal Process. Mag.*, vol. 22, no. 1, pp. 14–22, Jan. 2005.

[120] H. J. Trussell and R. E. Hartwig, “Mathematics for demosaicking,” *IEEE Trans. Image Process.*, vol. 11, no. 4, pp. 485–492, Apr. 2002.

[121] M. Rafinazari and E. Dubois, “Demosaicking algorithm for the Kodak-RGBW color filter array,” 2015, p. 939503.

[122] K. S. Song, C. H. Park, J. Kim, and M. G. Kang, “Color interpolation algorithm for an RWB color filter array including double-exposed white channel,” *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, p. 58, Dec. 2016.

[123] “True Color Kodak Images,” *True Color Kodak Image Set*, 11-Jul-2017. [Online]. Available: http://r0k.us/graphics/kodak/. [Accessed: 09-Jul-2017].

[124] "CDM Dataset." [Online]. Available: http://www4.comp.polyu.edu.hk/~cslzhang/CDM_Dataset.htm. [Accessed: 09-Apr-2017].

[125] "Index of /~laurent.condat/download." [Online]. Available: https://www.gipsa-lab.grenoble-inp.fr/~laurent.condat/download/. [Accessed: 09-Apr-2017].

[126] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009.

[127] Z. Wang and A. C. Bovik, *Modern Image Quality Assessment*. Morgan & Claypool Publishers, 2006.

[128] "SIPI Image Database - Misc," *The USC-SIPI Image Set*, 11-Jul-2017. [Online]. Available: http://sipi.usc.edu/database/database.php?volume=misc. [Accessed: 10-Jul-2017].

[129] S. Andriani, H. Brendel, T. Seybold, and J. Goldstone, "Beyond the Kodak image set: A new reference set of color image sequences," in *2013 IEEE International Conference on Image Processing*, 2013, pp. 2289–2293.

[130] "ARRI Imageset." [Online]. Available: ftp://imageset@ftp.arri.de. [Accessed: 23-May-2017].

[131] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: A Feature SImilarity Index for Image Quality Assessment," *IEEE*, vol. 20, no. 8, pp. 2378–2386, Aug. 2011.

[132] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-Reference Image Quality Assessment in the Spatial Domain," *IEEE Trans. Image Process.*, vol. 21, no. 12, pp. 4695–4708, Dec. 2012.

[133] S. Mihoubi, O. Losson, B. Mathon, and L. Macaire, "Multispectral demosaicing using pseudo-panchromatic image," *IEEE Trans. Comput. Imaging*, vol. PP, no. 99, pp. 1–1, 2017.

[134] M. Aghagolzadeh, A. Abdolhosseini Moghadam, M. Kumar, and H. Radha, "Bayer and panchromatic color filter array demosaicing by sparse recovery," 2011, vol. 7876, pp. 787603-787603–11.

[135] X. Chen, L. He, J. Tang, and Y. S. Lee, "A Low-Complexity Interpolation Method for Single-Sensor Camera Imaging with White-RGB Color Filter Array," in *2015 11th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, 2015, pp. 560–565.

[136] S. W. Golomb, *Polyominoes: Puzzles, Patterns, Problems and Packings*, 2nd ed. Princeton, New Jersey: Princeton University Press, 1994.

[137] X. Chen, G. Jeon, and J. Jeong, "Voting-Based Directional Interpolation Method and Its Application to Still Color Image Demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 2, pp. 255–262, Feb. 2014.

[138] **K. Wachira, "Corrective term usage in the improvement of gradient-based bayer CFA demosaicking algorithms," in *EUROCON 2015 - International Conference on Computer as a Tool (EUROCON), IEEE*, 2015, pp. 1–6.**

[139] "McGill Calibrated Colour Image Database." [Online]. Available: http://tabby.vision.mcgill.ca/html/browsedownload.html. [Accessed: 04-Oct-2017].

[140] C. Wang and J. Chong, "An Improved White-RGB Color Filter Array Based CMOS Imaging System for Cell Phones in Low-Light Environments," *IEICE Trans. Inf. Syst.*, vol. E97.D, no. 5, pp. 1386–1389, 2014.

[141] "IEEE Xplore Digital Library." [Online]. Available: http://ieeexplore.ieee.org/Xplore/home.jsp. [Accessed: 09-Apr-2017].

[142] "Home - Springer." [Online]. Available: https://link.springer.com/. [Accessed: 09-Apr-2017].

[143] "Home | SPIE." [Online]. Available: http://spiedigitallibrary.org/index.aspx. [Accessed: 09-Apr-2017].

Appendix A: MATLAB Code Blocks

The MATLAB code implementations of the following demosaicking algorithms are provided below:

A.1 Proposed Algorithm Block

```matlab
%==========================================================================
%   Name:       ordinal5Tris_v1.m
%   Author:     Kinyua Wachira, Univ. of Nairobi
%   Date:       30-04-2017 (Completion)
%   Desc:       an ordinal pentomino inspired path demosaicking algorithm,
%                working under a Bayer equivalent CFA
%
%==========================================================================

%==========================================================================
%   Preamble
%==========================================================================
% utility fcns
% clc; clear all; close all hidden;

% load image and generate Bayer CFA representation - in particular rgbg
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\USC-SIPI\sipi_im16.tiff');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\Kodak\kodim24.png');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\McM\mcm18.tif');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\Condat\codim30.tif');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\ARRI\arri_im12.tif');
img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\Custom1\cusim15.jpg');

imgBayer = fcn_bayerisation(img); %used to perform the bayerisation process


imgBayer = double(imgBayer); %need to make this a double during calculation
%imgBayer = img;
[R,C] = size(imgBayer);
[imgRed,imgGrn,imgBlu] = deal(double(zeros(R,C)));
imgPTI = deal(double(zeros(R,C,3)));
%also set a border padding size
pad=3; %during interpolation - the maximum width
padBorder = 4; %when calculating PSNR, CPSNR

for i=1:1:R;
    for j=1:1:C;
        if (mod(i,2)==1 && mod(j,2)==1)
            imgRed(i,j) = imgBayer(i,j);
        elseif (mod(i+j,2)==1)
            imgGrn(i,j) = imgBayer(i,j);
        else
            imgBlu(i,j) = imgBayer(i,j);
        end;
    end;
end;

%==========================================================================
%   Algorithm
%==========================================================================
% SECTION I: GREEN CHANNEL INTERPOLATION

%-----------------------------------------------------------------
% step 0: set the 5 variables and perform the interpolation process
%-----------------------------------------------------------------
```

```
%variable set 1: choice of epsilon
err = 4; %using my epsilon analysis information over 4 image sets

%variable set 2: green plane overemphasis and no emphasis factors
noemp = 1;
empGrn = 3;
empOpp = 2;
%variable set 3: non uniform interplane weighting
k1 = 0.8;
k2 = 0.7;

%-----------------------------------------------------
% step 1: set up maps to be used for polling
%-----------------------------------------------------
[HRmap,VRmap,HBmap,VBmap] = deal(double(zeros(R,C)));
[crmap,cbmap] = deal(double(zeros(R,C)));

%------------------------------------------------------
% step 2: populate the Hmaps and Vmaps
%------------------------------------------------------
for i=1+2:1:R-2;
    for j=1+2:1:C-2;
        if (mod(i,2)==1 && mod(j,2)==1) %red pixel locations
            HRmap(i,j) = 0.25*abs(2.*imgBayer(i,j) - imgBayer(i,j-2) - 
imgBayer(i,j+2)) ...
                        + 0.5*abs(imgBayer(i,j-1) - imgBayer(i,j+1));
            VRmap(i,j) = 0.25*abs(2*imgBayer(i,j) - imgBayer(i-2,j) - 
imgBayer(i+2,j)) ...
                        + 0.5*abs(imgBayer(i-1,j) - imgBayer(i+1,j));
        end;
        if (mod(i,2)==0 && mod(j,2)==0) %blue pixel locations
            HBmap(i,j) = 0.25.*abs(2.*imgBayer(i,j) - imgBayer(i,j-2) - 
imgBayer(i,j+2)) ...
                        + 0.5*abs(imgBayer(i,j-1) - imgBayer(i,j+1));
            VBmap(i,j) = 0.25*abs(2*imgBayer(i,j) - imgBayer(i-2,j) - 
imgBayer(i+2,j)) ...
                        + 0.5*abs(imgBayer(i-1,j) - imgBayer(i+1,j));
        end;
    end;
end;

%---------------------------------------------------------
% step 3: populate the cmaps - the choice/polling maps
%---------------------------------------------------------
for i=1:1:R;
    for j=1:1:C;
        if (HRmap(i,j) > 2*VRmap(i,j))
            crmap(i,j) = 1;
        end;
        if (VRmap(i,j) > 2*HRmap(i,j))
            crmap(i,j) = -1;
        end;
        if (HBmap(i,j) > 2*VBmap(i,j))
            cbmap(i,j) = 1;
        end;
        if (VBmap(i,j) > 2*HBmap(i,j))
            crmap(i,j) = -1;
        end;
    end;
end;

%-------------------------------------------------------
% step 4: perform the green interpolation, with polling
%-------------------------------------------------------
for i=1+pad:1:R-pad;
    for j=1+pad:1:C-pad;
        if ~(mod(i+j,2)==1)
            %initial estimates
            GNWest = 0.5*(imgGrn(i-1,j) + imgGrn(i,j-1)) + (k1*k2).*(imgBayer(i,j) - imgBayer(i-2,j-2));
            GSWest = 0.5*(imgGrn(i+1,j) + imgGrn(i,j-1)) + (k1*k2).*(imgBayer(i,j) - imgBayer(i+2,j-2));
            GSEest = 0.5*(imgGrn(i,j+1) + imgGrn(i+1,j)) + (k1*k2).*(imgBayer(i,j) - imgBayer(i+2,j+2));
            GNEest = 0.5*(imgGrn(i,j+1) + imgGrn(i-1,j)) + (k1*k2).*(imgBayer(i,j) - imgBayer(i-2,j+2));

            %establish gradients
            gNW = abs(imgGrn(i,j-1) - imgGrn(i-1,j-2)) + ... %g43-g32 --N
                 abs(imgGrn(i,j+1) - imgGrn(i-1,j-2)) + ... %g45-g32 --N
                 abs(imgGrn(i,j+1) - imgGrn(i-2,j-1)) + ... %g45-g23 --Z
                 abs(imgGrn(i-1,j) - imgGrn(i-2,j-1)) + ... %g34-g23 --Z
                 abs(imgGrn(i+1,j) - imgGrn(i,j-1)) + ... %g54-g43 --W
                 abs(imgGrn(i,j-1) - imgGrn(i-1,j-2)) + ... %g43-g32 --W
                 k1.*abs(imgBayer(i,j) - imgBayer(i-2,j-2)) + ...%r44-r22
                 k2.*abs(imgBayer(i-1,j-1) - imgBayer(i-3,j-3)) + ...%b33-b11
                 err;

            gSW = abs(imgGrn(i,j+1) - imgGrn(i+2,j-1)) + ... %g45-g63 --Z
                 abs(imgGrn(i+1,j) - imgGrn(i+2,j-1)) + ... %g54-g63 --Z
                 abs(imgGrn(i-1,j) - imgGrn(i,j-1)) + ... %g34-g43 --W
                 abs(imgGrn(i,j-1) - imgGrn(i+1,j-2)) + ...%g43-g52 --N
                 abs(imgGrn(i,j+1) - imgGrn(i+1,j-2)) + ...%g45-g52 --N
                 abs(imgGrn(i,j-1) - imgGrn(i+1,j-2)) + ...%g43-g52 --W
                 k1.*abs(imgBayer(i,j) - imgBayer(i+2,j-2)) + ...%r44-r62
                 k2.*abs(imgBayer(i+1,j-1) - imgBayer(i+3,j-3)) + ...%b53-b71
                 err;

            gNE = abs(imgGrn(i,j-1) - imgGrn(i-1,j+2)) + ... %g43-g36 --N
                 abs(imgGrn(i,j+1) - imgGrn(i-1,j+2)) + ... %g45-g36 --N
                 abs(imgGrn(i+1,j) - imgGrn(i,j+1)) + ... %g54-g45 --W
                 abs(imgGrn(i,j+1) - imgGrn(i-1,j+2)) + ... %g45-g36 --W
                 abs(imgGrn(i,j-1) - imgGrn(i-2,j+1)) + ... %g43-g25 --Z
                 abs(imgGrn(i-1,j) - imgGrn(i-2,j+1)) + ... %g34-g25 --Z
                 k1.*abs(imgBayer(i,j) - imgBayer(i-2,j+2)) + ...%r44-r26
                 k2.*abs(imgBayer(i-1,j+1) - imgBayer(i-3,j+3)) + ...%b35-b17
                 err;

            gSE = abs(imgGrn(i,j+1) - imgGrn(i+1,j+2)) + ... %g45-g56 --N
                 abs(imgGrn(i,j-1) - imgGrn(i+1,j+2)) + ... %g43-g56 --N
                 abs(imgGrn(i-1,j) - imgGrn(i,j+1)) + ... %g34-g45 --W
                 abs(imgGrn(i,j+1) - imgGrn(i+1,j+2)) + ...%g45-g56 --W
                 abs(imgGrn(i,j-1) - imgGrn(i+2,j+1)) + ...%g43-g65 --Z
                 abs(imgGrn(i+1,j) - imgGrn(i+2,j+1)) + ...%g54-g65 --Z
                 k1.*abs(imgBayer(i,j) - imgBayer(i+2,j+2)) + ...%r44-r66
                 k2.*abs(imgBayer(i+1,j+1) - imgBayer(i+3,j+3)) + ...%b55-b77
                 err;

             %establish weights
             wNW = 1./gNW;
             wSE = 1./gSE;
             wSW = 1./gSW;
             wNE = 1./gNE;

             %set up the voting mechanism
             if (mod(i,2)==1 && mod(j,2)==1) %red pixel centre
                 F = crmap(i,j) + ...
                     cbmap(i-1,j-1) + cbmap(i+1,j+1) + ...
                     cbmap(i-1,j+1) + cbmap(i+1,j-1);
             end;
             if (mod(i,2)==0 && mod(j,2)==0) %blue pixel centre
                 F = cbmap(i,j) + ...
```

```
                     crmap(i-1,j-1) + crmap(i+1,j+1) + ...
                     crmap(i-1,j+1) + crmap(i+1,j-1);
             end;

             switch F
                 case {4,5} % predominantly vertical
                     imgGrn(i,j) = (wNW.*GNWest + wSE.*GSEest)/(wNW+wSE);
                 case {-4,-5} % predominantly horizontal
                     imgGrn(i,j) = (wSW.*GSWest + wNE.*GNEest)/(wSW+wNE);
                 otherwise % undetermined
                     imgGrn(i,j) = (wNW.*GNWest + wSE.*GSEest + ...
                                    wSW.*GSWest + wNE.*GNEest)/(wNW+wSE+wSW+wNE);
             end;
        end;
    end;
end;

% SECTION II: RED CHANNEL INTERPOLATION
% red content in blue pixel points
for i=1+pad:1:R-pad;
    for j=1+pad:1:C-pad;
        if (mod(i,2)==0 && mod(j,2)==0)
            %initial estimates in the NW, NE, SE, SW directions
            rNW = imgBayer(i-1,j-1) - imgGrn(i-1,j-1);
            rNE = imgBayer(i-1,j+1) - imgGrn(i-1,j+1);
            rSE = imgBayer(i+1,j+1) - imgGrn(i+1,j+1);
            rSW = imgBayer(i+1,j-1) - imgGrn(i+1,j-1);

            %gradient determination
            gNW = k2.*abs(imgBayer(i-1,j-1) - imgBayer(i+1,j+1)) + ... %opp plane
                  empGrn.*abs(imgGrn(i,j) - imgGrn(i-1,j-1)) + ... %green in-line
                  empGrn.*abs(imgGrn(i-1,j-1) - imgGrn(i-2,j-2)) + ...
                  noemp.*abs(imgGrn(i-1,j) - imgGrn(i-2,j-1)) + ...%green outlier
                  noemp.*abs(imgGrn(i,j-1) - imgGrn(i-1,j-2)) + ...
                  err;%desired

            gNE = k2.*abs(imgBayer(i-1,j+1) - imgBayer(i+1,j-1)) + ...
                  empGrn.*abs(imgGrn(i,j) - imgGrn(i-1,j+1)) + ...
                  empGrn.*abs(imgGrn(i-1,j+1) - imgGrn(i-2,j+2)) + ...
                  noemp.*abs(imgGrn(i-1,j) - imgGrn(i-2,j+1))+ ...
                  noemp.*abs(imgGrn(i,j+1) - imgGrn(i-1,j+2)) + ...
                  err;

            gSE = k2.*abs(imgBayer(i+1,j+1) - imgBayer(i-1,j-1)) + ...
                  empGrn.*abs(imgGrn(i,j) - imgGrn(i+1,j+1)) + ...
                  empGrn.*abs(imgGrn(i+1,j+1) - imgGrn(i+2,j+2)) + ...
                  noemp.*abs(imgGrn(i+1,j) - imgGrn(i+2,j+1)) + ...
                  noemp.*abs(imgGrn(i,j+1) - imgGrn(i+1,j+2)) + ...
                  err;

            gSW = k2.*abs(imgBayer(i+1,j-1) - imgBayer(i-1,j+1)) + ...
                  empGrn.*abs(imgGrn(i,j) - imgGrn(i+1,j-1)) + ...
                  empGrn.*abs(imgGrn(i+1,j-1) - imgGrn(i+2,j-2)) + ...
                  noemp.*abs(imgGrn(i+1,j) - imgGrn(i+2,j-1)) + ...
                  noemp.*abs(imgGrn(i,j-1) - imgGrn(i+1,j-2)) + ...
                  err;

            % weights
            wNW = 1./gNW;
            wNE = 1./gNE;
            wSE = 1./gSE;
            wSW = 1./gSW;
            w = wNW + wNE + wSE + wSW;

            r = imgGrn(i,j) + (wNW.*rNW + wNE.*rNE + wSE.*rSE +
wSW.*rSW)/w;
            imgRed(i,j) = r;
        end;
    end;
end;


for i=1+pad:1:R-pad;
    for j=1+pad:1:C-pad;
        if (mod(i+j,2)==1)

            %initial estimates of N,E,W,S directions
            rN = imgRed(i-1,j) + (k1*k2).*(imgGrn(i,j) - imgGrn(i-2,j));
            rS = imgRed(i+1,j) + (k1*k2).*(imgGrn(i,j) - imgGrn(i+2,j));
            rE = imgRed(i,j+1) + (k1*k2).*(imgGrn(i,j) - imgGrn(i,j+2));
            rW = imgRed(i,j-1) + (k1*k2).*(imgGrn(i,j) - imgGrn(i,j-2));

            %establish gradients
            gN = abs(imgRed(i-2,j-1) - imgRed(i,j-1)) + ...
                 empOpp.*abs(imgRed(i-1,j) - imgRed(i-2,j-1)) + ...
                 empOpp.*abs(imgRed(i-1,j) - imgRed(i-2,j+1)) + ...
                 abs(imgRed(i-2,j+1) - imgRed(i,j+1)) + ...
                 abs(imgGrn(i-3,j-1) - imgGrn(i-1,j-1)) + ...
                 abs(imgGrn(i-3,j+1) - imgGrn(i-1,j+1)) + ...
                 abs(imgGrn(i-2,j) - imgGrn(i,j)) + ...
                 err;
            gS = abs(imgRed(i,j-1) - imgRed(i+2,j-1)) + ...
                 empOpp.*abs(imgRed(i+1,j) - imgRed(i+2,j-1)) + ...
                 empOpp.*abs(imgRed(i+1,j) - imgRed(i+2,j+1)) + ...
                 abs(imgRed(i,j+1) - imgRed(i+2,j+1)) + ...
                 abs(imgGrn(i+1,j-1) - imgGrn(i+3,j-1)) + ...
                 abs(imgGrn(i+1,j+1) - imgGrn(i+3,j+1)) + ...
                 abs(imgGrn(i,j) - imgGrn(i+2,j)) + ...
                 err;
            gW = abs(imgRed(i-1,j-2) - imgRed(i-1,j)) + ...
                 abs(imgRed(i+1,j-2) - imgRed(i+1,j)) + ...
                 empOpp.*abs(imgRed(i,j-1) - imgRed(i-1,j-2)) + ...
                 empOpp.*abs(imgRed(i,j-1) - imgRed(i+1,j-2)) + ...
                 abs(imgGrn(i-1,j-3) - imgGrn(i-1,j-1)) + ...
                 abs(imgGrn(i+1,j-3) - imgGrn(i+1,j-1)) + ...
                 abs(imgGrn(i,j-2) - imgGrn(i,j))+ ...
                 err;
            gE = empOpp.*abs(imgRed(i,j+1) - imgRed(i-1,j+2)) + ...
                 empOpp.*abs(imgRed(i,j+1) - imgRed(i+1,j+2)) + ...
                 abs(imgRed(i-1,j) - imgRed(i-1,j+2)) + ...
                 abs(imgRed(i+1,j) - imgRed(i+1,j+2)) + ...
                 abs(imgGrn(i-1,j+1) - imgGrn(i-1,j+3)) + ...
                 abs(imgGrn(i+1,j+1) - imgGrn(i+1,j+3)) + ...
                 abs(imgGrn(i,j) - imgGrn(i,j+2)) + ...
                 err;

            % weights
            wN = 1./gN;
            wS = 1./gS;
            wW = 1./gW;
            wE = 1./gE;
            w = wN + wS + wE + wW;

            imgRed(i,j) = (wN*rN + wE*rE + wW*rW + wS*rS)/w;
        end;
    end;
end;

% SECTION III: BLUE CHANNEL INTERPOLATION
% blue content in red pixel points
for i=1+pad:1:R-pad;
    for j=1+pad:1:C-pad;
```

```
        if (mod(i,2)==1 && mod(j,2)==1)
            %initial estimates in the NW, NE, SE, SW directions
            bNW = imgBayer(i-1,j-1) - imgGrn(i-1,j-1);
            bNE = imgBayer(i-1,j+1) - imgGrn(i-1,j+1);
            bSE = imgBayer(i+1,j+1) - imgGrn(i+1,j+1);
            bSW = imgBayer(i+1,j-1) - imgGrn(i+1,j-1);

            %gradient determination
            gNW = k2.*abs(imgBayer(i-1,j-1) - imgBayer(i+1,j+1)) + ... %opp plane
                  empGrn.*abs(imgGrn(i,j) - imgGrn(i-1,j-1)) + ... %green in-line
                  empGrn.*abs(imgGrn(i-1,j-1) - imgGrn(i-2,j-2)) + ...
                  noemp.*abs(imgGrn(i-1,j) - imgGrn(i-2,j-1)) + ...%green outlier
                  noemp.*abs(imgGrn(i,j-1) - imgGrn(i-1,j-2)) + ...
                  err;%desired

            gNE = k2.*abs(imgBayer(i-1,j+1) - imgBayer(i+1,j-1)) + ...
                  empGrn.*abs(imgGrn(i,j) - imgGrn(i-1,j+1)) + ...
                  empGrn.*abs(imgGrn(i-1,j+1) - imgGrn(i-2,j+2)) + ...
                  noemp.*abs(imgGrn(i-1,j) - imgGrn(i-2,j+1))+ ...
                  noemp.*abs(imgGrn(i,j+1) - imgGrn(i-1,j+2)) + ...
                  err;

            gSE = k2.*abs(imgBayer(i+1,j+1) - imgBayer(i-1,j-1)) + ...
                  empGrn.*abs(imgGrn(i,j) - imgGrn(i+1,j+1)) + ...
                  empGrn.*abs(imgGrn(i+1,j+1) - imgGrn(i+2,j+2)) + ...
                  noemp.*abs(imgGrn(i+1,j) - imgGrn(i+2,j+1)) + ...
                  noemp.*abs(imgGrn(i,j+1) - imgGrn(i+1,j+2)) + ...
                  err;

            gSW = k2.*abs(imgBayer(i+1,j-1) - imgBayer(i-1,j+1)) + ...
                  empGrn.*abs(imgGrn(i,j) - imgGrn(i+1,j-1)) + ...
                  empGrn.*abs(imgGrn(i+1,j-1) - imgGrn(i+2,j-2)) + ...
                  noemp.*abs(imgGrn(i+1,j) - imgGrn(i+2,j-1)) + ...
                  noemp.*abs(imgGrn(i,j-1) - imgGrn(i+1,j-2)) + ...
                  err;

            % weights
            wNW = 1./gNW;
            wNE = 1./gNE;
            wSE = 1./gSE;
            wSW = 1./gSW;
            w = wNW + wNE + wSE + wSW;


            b = imgGrn(i,j) + (wNW.*bNW + wNE.*bNE + wSE.*bSE + wSW.*bSW)/w;
            imgBlu(i,j) = b;
        end;
    end;
end;

for i=1+pad:1:R-pad;
    for j=1+pad:1:C-pad;
        if (mod(i+j,2)==1)

            %initial estimates of N,E,W,S directions
            bN = imgBlu(i-1,j) + (k1*k2).*(imgGrn(i,j) - imgGrn(i-2,j));
            bS = imgBlu(i+1,j) + (k1*k2).*(imgGrn(i,j) - imgGrn(i+2,j));
            bE = imgBlu(i,j+1) + (k1*k2).*(imgGrn(i,j) - imgGrn(i,j+2));
            bW = imgBlu(i,j-1) + (k1*k2).*(imgGrn(i,j) - imgGrn(i,j-2));

            %establish gradients
            gN = abs(imgBlu(i-2,j-1) - imgBlu(i,j-1)) + ...
                 empOpp.*abs(imgBlu(i-1,j) - imgBlu(i-2,j-1)) + ...
                 empOpp.*abs(imgBlu(i-1,j) - imgBlu(i-2,j+1)) + ...
```

```
                    abs(imgBlu(i-2,j+1) - imgBlu(i,j+1)) + ...
                    abs(imgGrn(i-3,j-1) - imgGrn(i-1,j-1)) + ...
                    abs(imgGrn(i-3,j+1) - imgGrn(i-1,j+1)) + ...
                    abs(imgGrn(i-2,j) - imgGrn(i,j)) + ...
                    err;
                gS = abs(imgBlu(i,j-1) - imgBlu(i+2,j-1)) + ...
                    empOpp.*abs(imgBlu(i+1,j) - imgBlu(i+2,j-1)) + ...
                    empOpp.*abs(imgBlu(i+1,j) - imgBlu(i+2,j+1)) + ...
                    abs(imgBlu(i,j+1) - imgBlu(i+2,j+1)) + ...
                    abs(imgGrn(i+1,j-1) - imgGrn(i+3,j-1)) + ...
                    abs(imgGrn(i+1,j+1) - imgGrn(i+3,j+1)) + ...
                    abs(imgGrn(i,j) - imgGrn(i+2,j)) + ...
                    err;
                gW = abs(imgBlu(i-1,j-2) - imgBlu(i-1,j)) + ...
                    abs(imgBlu(i+1,j-2) - imgBlu(i+1,j)) + ...
                    empOpp.*abs(imgBlu(i,j-1) - imgBlu(i-1,j-2)) + ...
                    empOpp.*abs(imgBlu(i,j-1) - imgBlu(i+1,j-2)) + ...
                    abs(imgGrn(i-1,j-3) - imgGrn(i-1,j-1)) + ...
                    abs(imgGrn(i+1,j-3) - imgGrn(i+1,j-1)) + ...
                    abs(imgGrn(i,j-2) - imgGrn(i,j))+ ...
                    err;
                gE = empOpp.*abs(imgBlu(i,j+1) - imgBlu(i-1,j+2)) + ...
                    empOpp.*abs(imgBlu(i,j+1) - imgBlu(i+1,j+2)) + ...
                    abs(imgBlu(i-1,j) - imgBlu(i-1,j+2)) + ...
                    abs(imgBlu(i+1,j) - imgBlu(i+1,j+2)) + ...
                    abs(imgGrn(i-1,j+1) - imgGrn(i-1,j+3)) + ...
                    abs(imgGrn(i+1,j+1) - imgGrn(i+1,j+3)) + ...
                    abs(imgGrn(i,j) - imgGrn(i,j+2)) + ...
                    err;

                % weights
                wN = 1./gN;
                wS = 1./gS;
                wW = 1./gW;
                wE = 1./gE;
                w = wN + wS + wE + wW;

                imgBlu(i,j) = (wN*bN + wE*bE + wW*bW + wS*bS)/w;
            end;
        end;
    end;


%==========================================================================
%   Results
%==========================================================================
%----------------------------------------
% display images
%----------------------------------------
% imtool(img);
% imtool(imgBayer);
% imtool(uint8(imgRed));
% imtool(uint8(imgGrn));
% imtool(uint8(imgBlu));


%--------------------------
% performance metrics
%---------------------------
imgPTI(:,:,1) = imgRed;
imgPTI(:,:,2) = imgGrn;
imgPTI(:,:,3) = imgBlu;

% imtool(uint8(imgPTI));
%imwrite(uint8(imgMWILP),'C:\Users\Kinyua Wachira\Desktop\MWILP_dem.tiff');

MSE = fcn_measureMSESinglev2(uint8(imgPTI(:,:,2)),uint8(img(:,:,2)),4);
[FSIM,FSIMc] = FeatureSIM(uint8(img),uint8(imgPTI));
```

```
CPSNR = fcn_measureCPSNRv2(uint8(imgPTI),uint8(img),4);
SSIM = ssim(img,uint8(imgPTI));
```

A.2 Functions comprising the Image Acquisition Algorithm Block

Ground Truth Reference Image to Bayer CFA Equivalent Conversion Function

```
%==========================================================================
% Author:   Kinyua Wachira, Univ. of Nairobi
% Date:     2016 (Development)
% File:     function_ConvertToBayerCFAv2.m
% Desc:     a function to convert an RGB image to its Bayer equivalent

% Notes:    this function assumes the CFA is RGBG from top left clockwise
%==========================================================================

function [imgBayer, imgFullBayer] = function_ConvertToBayerCFA(img)

%setup
[R,C,k] = size(img);
imgBayer = uint8(zeros(R,C));
imgFullBayer = uint8(zeros(R,C,k));

%populate the Bayer CFA matrices
for i=1:1:R;
    for j=1:1:C;
        if (mod(i+j,2)==1)
                imgBayer(i,j) = img(i,j,2); %green component
                imgFullBayer(i,j,2) = img(i,j,2);
        end;
        if (mod(i,2)==0 && mod(j,2)==0)
                imgBayer(i,j) = img(i,j,3); %blue component
                imgFullBayer(i,j,3) = img(i,j,3);
        end;
        if (mod(i,2)==1 && mod(j,2)==1)
                imgBayer(i,j) = img(i,j,1); %red component
                imgFullBayer(i,j,1) = img(i,j,1);
        end;
    end;
end;
```

Ground Truth Reference Image to RGBW CFA Equivalent Conversion Function

```
%==========================================================================
% Author:   Kinyua Wachira, Univ. of Nairobi
% Date:     2016-2017 (Development)
% File:     function_ConvertToRGBWCFAv2.m
% Desc:     a function to convert an RGB image to its Bayer equivalent

% Notes:    this function assumes the CFA is RGBW from top left clockwise
%==========================================================================

function [imgRGBW]= function_ConvertToRGBWCFA(img)

%setup
[R,C,N] = size(img);
imgRGBW = uint8(zeros(R,C,N));

%populate the Bayer CFA matrices
for i=1:1:R;
    for j=1:1:C;
        for k=1:1:N;
            if (mod(i,2)==0 && mod(j,2)==0)
```

```
                imgRGBW(i,j,3) = img(i,j,3); %blue component
            end;
            if (mod(i,2)==1 && mod(j,2)==1)
                imgRGBW(i,j,1) = img(i,j,1); %red component
            end;
            if (mod(i,2)==1 && mod(j,2)==0)
                imgRGBW(i,j,2) = img(i,j,2); %green component
            end;
            if (mod(i,2)==0 && mod(j,2)==1)
                imgRGBW(i,j,k) = img(i,j,k); %white component
            end;
        end;
    end;
end;
```

Ground Truth Reference Image to WRGB CFA Equivalent Conversion Function (a 90° clockwise shift of the preceding function block used in the ACR and EDCR algorithms)

```
%==========================================================================
% Author:   Kinyua Wachira, Univ. of Nairobi
% Date:     21-01-2016 (Completion)
% File:     function_ConvertToRGBWCFA.m
% Desc:     a function to convert an RGB image to its Bayer equivalent

% Notes:    this function assumes the CFA is RGBW from top left clockwise
%==========================================================================

function [imgWRGB]= function_ConvertToWRGBCFA(img)

%setup
[R,C,N] = size(img);
ImgWRGB = uint8(zeros(R,C,N));

%populate the Bayer CFA matrices
for i=1:1:R;
    for j=1:1:C;
        for k=1:1:N;
            if (mod(i,2)==1 && mod(j,2)==1)
                imgWRGB(i,j,k) = img(i,j,k); %white component
            end;
            if (mod(i,2)==1 && mod(j,2)==0)
                imgWRGB(i,j,1) = img(i,j,1); %red component
            end;
            if (mod(i,2)==0 && mod(j,2)==0)
                imgWRGB(i,j,2) = img(i,j,2); %green component
            end;
            if (mod(i,2)==0 && mod(j,2)==1)
                imgWRGB(i,j,3) = img(i,j,3); %blue component
            end;
        end;
    end;
end;
```

Bayerisation Algorithm (Author's Implementation)

```
%==========================================================================
%   Name:       fcn_bayerisation.m
%   Author:     Kinyua Wachira, Univ. of Nairobi
%   Date:       2017 (development)
%   Desc:       a function to simulate the Bayer CFA
%
%   Notes:      the Bayer image is going to be grayscale
%==========================================================================

function [imageBayer] = fcn_bayerisation(image)
%construct the Bayer representation
[R,C,k] = size(image);
imageBayer = uint8(zeros(R,C));

h = 0.125.* [0 0 -1.5 0 0; 0 2 0 2 0; -1.5 0 6 0 -1.5; 0 2 0 2 0;
    0 0 -1.5 0 0];

image = h.*image;
image = function_ConvertToBayerCFA(image);
for i=1:1:R;
    for j=1:1:C;
        if (mod(i,2)==1 && mod(j,2)==1)
            imageBayer(i,j) = image(i,j,1);
        elseif (mod(i+j,2) == 1)
            imageBayer(i,j) = image(i,j,2);
        else
            imageBayer(i,j) = image(i,j,3);
        end;
    end;
end;
```

Image Reading Function Block (Default MATLAB source implementation)

```
function [X, map, alpha] = imread(varargin)
%IMREAD Read image from graphics file.
%   A = IMREAD(FILENAME,FMT) reads a grayscale or color image from the file
%   specified by the string FILENAME. FILENAME must be in the current
%   directory, in a directory on the MATLAB path, or include a full or
%   relative path to a file.
%
%   The text string FMT specifies the format of the file by its standard
%   file extension. For example, specify 'gif' for Graphics Interchange
%   Format files. To see a list of supported formats, with their file
%   extensions, use the IMFORMATS function. If IMREAD cannot find a file
%   named FILENAME, it looks for a file named FILENAME.FMT.
%
%   The return value A is an array containing the image data. If the file
%   contains a grayscale image, A is an M-by-N array. If the file contains
%   a truecolor image, A is an M-by-N-by-3 array. For TIFF files containing
%   color images that use the CMYK color space, A is an M-by-N-by-4 array.
%   See TIFF in the Format-Specific Information section for more
%   information.
%
%   The class of A depends on the bits-per-sample of the image data,
%   rounded to the next byte boundary. For example, IMREAD returns 24-bit
%   color data as an array of uint8 data because the sample size for each
%   color component is 8 bits. See the Remarks section for a discussion of
%   bitdepths, and see the Format-Specific Information section for more
%   detail about supported bitdepths and sample sizes for a particular
%   format.
%
%   [X,MAP] = IMREAD(FILENAME,FMT) reads the indexed image in FILENAME into
%   X and its associated colormap into MAP. Colormap values in the image
%   file are automatically rescaled into the range [0,1].
%
%   [...] = IMREAD(FILENAME) attempts to infer the format of the file
%   from its content.
%
%   [...] = IMREAD(URL,...) reads the image from an Internet URL.
%
%   Remarks
%
%   Bitdepth is the number of bits used to represent each image pixel.
%   Bitdepth is calculated by multiplying the bits-per-sample with the
%   samples-per-pixel. Thus, a format that uses 8-bits for each color
%   component (or sample) and three samples per pixel has a bitdepth of 24.
%   Sometimes the sample size associated with a bitdepth can be ambiguous:
%   does a 48-bit bitdepth represent six 8-bit samples or three 16-bit
%   samples? The following format-specific sections provide sample size
%   information to avoid this ambiguity.
%
%   Format-Specific Information (Listed Alphabetically by Format)
%
%   BMP  --  Windows Bitmap
%
%   Supported  Compression     Output
%   Bitdepths  None    RLE     Class    Notes
%   ---------------------------------------------------------
%    1-bit      x        -     logical
%    4-bit      x        x     uint8
%    8-bit      x        x     uint8
%   16-bit      x        -     uint8    1 sample/pixel
%   24-bit      x        -     uint8    3 samples/pixel
%   32-bit      x        -     uint8    3 samples/pixel (1 byte padding)
%
%   CUR  -- Cursor File
%
%   Supported    Compression      Output
%   Bitdepths   None Compressed   Class
%   --------------------------------------------------
%   1-bit        x      -         logical
%   4-bit        x      -         uint8
%   8-bit        x      -         uint8
%
%   Special syntaxes:
%
%   [...] = IMREAD(...,IDX) reads in one image from a multi-image icon or
%   cursor file. IDX is an integer value that specifies the order that the
%   image appears in the file. For example, if IDX is 3, IMREAD reads the
%   third image in the file. If you omit this argument, IMREAD reads the
%   first image in the file.
%
%   [A,MAP,ALPHA] = IMREAD(...) returns the AND mask for the resource,
%   which can be used to determine transparency information.  For cursor
%   files, this mask may contain the only useful data.
%
%   GIF  --  Graphics Interchange Format
%
%   Supported     Output Class
%   ---------------------------
%   1-bit          logical
%   2-to-8 bit     uint8
%
%   Special syntaxes:
%
%   [...] = IMREAD(...,IDX) reads in one or more frames from a multiframe
%   (i.e., animated) GIF file. IDX must be an integer scalar or vector of
%   integer values.  For example, if IDX is 3, IMREAD reads the third image
%   in the file.  If IDX is 1:5, only the first five frames are returned.
%
%   [...] = IMREAD(...,'Frames',IDX) is the same as the syntax above except
%   that IDX can be 'all'.  In this case, all of the frames are read and
%   returned in the order that they appear in the file.
%
%   Note: Because of the way GIF files are structured, all of the frames
%   must be read when a particular frame is requested. Consequently, it is
%   much faster to specify a vector of frames or 'all' for IDX than to call
%   IMREAD in a loop when reading multiple frames from the same GIF file.
%
%   HDF  --  Hierarchical Data Format
%
%   Supported   Raster image   Raster image     Output
%   Bitdepths   with colormap  without colormap Class    Notes
%   ------------------------------------------------------------
%    8-bit        x                x             uint8
%   24-bit        -                x             uint8   3 samples/pixel
%
%   Special Syntaxes:
%
%   [...] = IMREAD(...,REF) reads in one image from a multi-image HDF file.
%   REF is an integer value that specifies the reference number used to
%   identify the image. For example, if REF is 12, IMREAD reads the image
%   whose reference number is 12. (Note that in an HDF file the reference
%   numbers do not necessarily correspond with the order of the images in
%   the file. You can use IMFINFO to match up image order with reference
%   number.) If you omit this argument, IMREAD reads the first image in
%   the file.
%
%   ICO  -- Icon File
%
%   See CUR.
%
%   JPEG  --  Joint Photographic Experts Group
%
%   Note: IMREAD can read any baseline JPEG image as well as JPEG images
%   with some commonly used extensions.
%
%   Supported    Compression      Output
%   Bitdepths   Lossy Lossless    Class      Notes
%   --------------------------------------------------------
%    8-bit        x      x        uint8     Grayscale or RGB
%   12-bit        x      x        uint16    Grayscale
%   16-bit        -      x        uint16    Grayscale
%   36-bit        x      x        uint16    RGB(Three 12-bit samples/pixel)
%
%   JPEG 2000 - Joint Photographic Experts Group 2000
%
%   Supported      Compression      Output
%   Bitdepths     Lossy Lossless    Class
%   (per sample)
%   ----------------------------------------------------------
%    1-bit          x      x        logical
%    2- to 8-bit    x      x        uint8, int8
%    9- to 16-bit   x      x        uint16, int16
%
%   Note: Indexed JPEG 2000 images are not supported. Only JP2 compatible
%   color spaces are supported for JP2/JPX files.   By default, all image
%   channels are returned in the order they are stored in the file.
%
%   Special Syntaxes
%
%   [...] = IMREAD(..., 'Param1', value1, 'Param2', value2, ...) uses
%   parameter-value pairs to control the read operation.
%
%       Parameter name   Value
%       --------------   -----
%       'ReductionLevel' A non-negative integer specifying the reduction in
%                        the resolution of the image. For a reduction
%                        level 'L', the image resolution is reduced by a
%                        factor of 2^L. The default value is 0 implying
%                        no reduction. The reduction level is limited by
%                        the total number of decomposition levels as
%                        provided by 'WaveletDecompositionLevels' field
%                        in the structure returned from IMFINFO function.
%
%       'PixelRegion'    {ROWS, COLS}.  IMREAD returns the sub-image
%                        specified by the boundaries in ROWS and COLS.
%                        ROWS and COLS must both be two-element vectors
%                        that denote the 1-based indices [START STOP]. If
%                        'ReductionLevel' is greater than 0, then ROWS and
%                        COLS are coordinates in the reduced-sized image.
```

```
%
%      'V79Compatible'  A logical value. If true, the image returned is
%                       transformed to gray-scale or RGB as consistent with
%                       previous versions of IMREAD (MATLAB 7.9 [R2009b]
%                       and earlier).  Use this option to transform YCC
%                       images into RGB.  The default is false.
%
%   PBM  --  Portable Bitmap
%
%   Supported  Raw     ASCII (Plain)  Output
%   Bitdepths  Binary  Encoded        Class
%   ----------------------------------------
%   1-bit        x        x          logical
%
%   PCX  --  Windows Paintbrush
%
%   Supported     Output
%   Bitdepths     Class       Notes
%   ----------------------------------------------
%    1-bit        logical     Grayscale only
%    8-bit        uint8       Grayscale or indexed
%   24-bit        uint8       RGB (8-bit samples)
%
%   PGM  --  Portable Graymap
%
%   Supported        Raw      ASCII (Plain)  Output
%   Bitdepths        Binary   Encoded        Class
%   ------------------------------------------------
%   up to 16-bit      x            -         uint8
%   Arbitrary         -            x
%
%   PNG  --  Portable Network Graphics
%
%   Supported     Output
%   Bitdepths     Class      Notes
%   -------------------------------------------
%    1-bit        logical    Grayscale only
%    2-bit        uint8      Grayscale only
%    4-bit        uint8      Grayscale only
%    8-bit        uint8      Grayscale or Indexed
%   16-bit        uint16     Grayscale or Indexed
%   24-bit        uint8      RGB (Three 8-bit samples/pixel)
%   48-bit        uint16     RGB (Three 16-bit samples/pixel)
%
%   Special Syntaxes:
%
%   [...] = IMREAD(...,'BackgroundColor',BG) composites any transparent
%   pixels in the input image against the color specified in BG.  If BG is
%   'none', then no compositing is performed. Otherwise, if the input image
%   is indexed, BG should be an integer in the range [1,P] where P is the
%   colormap length. If the input image is grayscale, BG should be a value
%   in the range [0,1].  If the input image is RGB, BG should be a
%   three-element vector whose values are in the range [0,1]. The string
%   'BackgroundColor' may be abbreviated.
%
%   If the ALPHA output argument is used (see below), then BG defaults to
%   'none' if not specified by the user. Otherwise, if the PNG file
%   ontains a background color chunk, that color is used as the default
%   value for BG. If ALPHA is not used and the file does not contain a
%   background color chunk, then the default value for BG is 1 for indexed
%   images; 0 for grayscale images; and [0 0 0] for RGB images.
%
%   [A,MAP,ALPHA] = IMREAD(...) returns the alpha channel if one is
%   present; otherwise ALPHA is []. If 'BackgroundColor' is specified by
%   the user then ALPHA is []. Note that MAP may be empty if the file
%   contains a grayscale or truecolor image.
%
%   PPM  --  Portable Pixmap
```

```
%
%   Supported        Raw      ASCII (Plain)  Output
%   Bitdepths        Binary   Encoded        Class
%   ------------------------------------------------
%   up to 16-bit      x            -         uint8
%   Arbitrary         -            x
%
%   RAS  --  Sun Raster
%
%   Supported    Output
%   Bitdepths    Class     Notes
%   ----------------------------------------------------
%    1-bit       logical   Bitmap
%    8-bit       uint8     Indexed
%   24-bit       uint8     RGB (8-bit samples)
%   32-bit       uint8     RGB with Alpha (8-bit samples)
%
%   TIFF  --  Tagged Image File Format
%
%   NOTE:  Images with a YCbCr photometric interpretation are converted to
%   the RGB colorspace.
%
%   Special Syntaxes:
%
%   A = IMREAD(...) returns color data that uses the RGB, CIELAB, ICCLAB,
%   or CMYK color spaces.  If the color image uses the CMYK color space, A
%   is an M-by-N-by-4 array.
%
%   [...] = IMREAD(..., 'Param1', value1, 'Param2', value2, ...) uses
%   parameter-value pairs to control the read operation.  There are three
%   different parameters you can use:
%
%       Parameter name   Value
%       --------------   -----
%       'Index'          A positive integer specifying which image to read in
%                        a multi-image TIFF file.  For example, if 'Index' is
%                        3, IMREAD reads the third image in the file.
%
%       'Info'           A structure array; the output of IMFINFO.  When
%                        reading images from a multi-image TIFF file, passing
%                        the output of IMFINFO as the 'Info' parameter helps
%                        IMREAD locate the images in the file more quickly.
%
%       'PixelRegion'    {ROWS, COLS}.  IMREAD returns the sub-image
%                        specified by the boundaries in ROWS and COLS.  ROWS
%                        and COLS must be either two- or three-element
%                        vectors.  If two elements are provided, they denote
%                        the 1-based indices [START STOP].  If three elements
%                        are provided, the indices [START INCREMENT STOP]
%                        allow image downsampling.
%
%   XWD  --  X Window Dump
%
%   Supported                                 Output
%   Bitdepths  ZPixmaps  XYBitmaps  XYPixmaps  Class
%   ------------------------------------------------
%   1-bit        x          -         x        logical
%   8-bit        x          -         -        uint8
%
%   Please read the file libtiffcopyright.txt for more information.
%
%   Example:
%
%       imdata = imread('ngc6543a.jpg');
```

```
%
%   See also IMFINFO, IMWRITE, IMFORMATS, FREAD, IMAGE, DOUBLE, UINT8.

%   Copyright 1984-2015 The MathWorks, Inc.

[filename, fmt_s, extraArgs] = parse_inputs(varargin{:});

% Download remote file.
if (strfind(filename, '://'))

    url = true;

    if (~usejava('jvm'))
        error(message('MATLAB:imagesci:imread:noJava'))
    end

    try
        filename = urlwrite(filename, tempname);
    catch %#ok<*CTCH>
        error(message('MATLAB:imagesci:imread:readURL', filename));
    end

else

    url = false;

end

if (isempty(fmt_s))
    % The format was not specified explicitly.

    % Verify that the file exists.
    fid = fopen(filename, 'r');
    if (fid == -1)

        if ~isempty(dir(filename))
            error(message('MATLAB:imagesci:imread:fileReadPermission',
filename));
        else
            error(message('MATLAB:imagesci:imread:fileDoesNotExist',
filename));
        end

    else
        % File exists.  Get full filename.
        filename = fopen(fid);
        fclose(fid);
    end

    % Try to determine the file type.
    [format, fmt_s] = imftype(filename);

    if (isempty(format))
        error(message('MATLAB:imagesci:imread:fileFormat'));
    end

else
    % The format was specified explicitly.

    % Verify that the file exists.
    fid = fopen(filename, 'r');
    if (fid == -1)
        % Couldn't open using the given filename; search for a
        % file with an appropriate extension.
        for p = 1:length(fmt_s.ext)
            fid = fopen([filename '.' fmt_s.ext{p}]);

            if (fid ~= -1)
```

```
                    % The file was found.  Don't continue searching.
                    break
                end
            end
        end

        if (fid == -1)
            if ~isempty(dir(filename))
                error(message('MATLAB:imagesci:imread:fileReadPermission',
filename));
            else
                error(message('MATLAB:imagesci:imread:fileDoesNotExist',
filename));
            end
        else
            filename = fopen(fid);
            fclose(fid);
        end

end

if isempty(fmt_s)
    % Get format details.
    fmt_s = imformats(format);
end

% Verify that a read function exists
if (isempty(fmt_s.read))
    error(message('MATLAB:imagesci:imread:readFunctionRegistration',
fmt_s.ext{ 1 }));
end

if ((fmt_s.alpha) && (nargout == 3))

    % Use the alpha channel.
    [X, map, alpha] = feval(fmt_s.read, filename, extraArgs{:});

else

    % Alpha channel is not requested or is not applicable.
    alpha = [];
    [X, map] = feval(fmt_s.read, filename, extraArgs{:});

end

% Delete temporary file from Internet download.
if (url)
    delete_download(filename);
end



%--------------------------------------------------------------------------
function delete_download(filename)

try
    delete(filename);
catch
    warning(message('MATLAB:imagesci:imread:tempFileDelete', filename))
end




%--------------------------------------------------------------------------
function [filename, fmt_s, extraArgs] = parse_inputs(varargin)

filename = '';
```

```
fmt_s = struct([]);
extraArgs = {};

% Parse arguments based on their number.
switch(nargin)
case 0

    % Not allowed.
    error(message('MATLAB:imagesci:imread:inputParsing'));

case 1

    % Filename only.
    filename = varargin{1};
    if ~ischar(filename)
        error(message('MATLAB:imagesci:imread:badImageSourceDatatype'));
    end

otherwise

    % Filename and format or other arguments.
    filename = varargin{1};

    % Check whether second argument is a format.
    if (ischar(varargin{2}))
        fmt_s = imformats(varargin{2});
    end

    if (~isempty(fmt_s))
        % The argument matches a format.
        extraArgs = varargin(3:end);
    else
        % The argument begins the format-specific parameters.
        extraArgs = varargin(2:end);
    end

end
```

A.3 Functions comprising the Comparison Algorithm Block

MSE Evaluation Function (Author's Implementation)

```
%measure MSE in a single channel
function [MSE] = fcn_measureMSESinglev2(obsImg,actImg,b)

if( nargin < 3 )
 b = 0;
End

if( b > 0 )
 actImg = actImg(b:size(actImg,1)-b, b:size(actImg,2)-b);
 obsImg = obsImg(b:size(obsImg,1)-b, b:size(obsImg,2)-b);
End

diff = (actImg - obsImg);
diff = diff .* diff;

MSE = sum( diff(:) ) / numel(diff) + 1e-32; %small positive factor to avoid
0 MSE

end
```

CPSNR Evaluation Function (Author's Implementation)

```
%measure CPSNR
%obsImg = observedImage
%actImg = actualImage
%b = paddding border width

function [CPSNR] = fcn_measureCPSNRv2(obsImg,actImg,b)
peak = 255;

if( nargin < 3 )
 b = 0;
end

if( b > 0 )
 actImg = actImg(b:size(actImg,1)-b, b:size(actImg,2)-b,:);
 obsImg = obsImg(b:size(obsImg,1)-b, b:size(obsImg,2)-b,:);
end

dif = (actImg - obsImg);
dif = dif .* dif;

MSE = sum( dif(:) ) / numel(dif) + 1e-32; %small positive factor to avoid 0
MSE

CPSNR = 10 * log10( peak*peak / MSE );

end
```

SSIM Evaluation Function (source [131])

```
function [mssim, ssim_map] = ssim(img1, img2, K, window, L)

% ========================================================================
% SSIM Index with automatic downsampling, Version 1.0
% Copyright(c) 2009 Zhou Wang
% All Rights Reserved.
%
% ----------------------------------------------------------------------
% Permission to use, copy, or modify this software and its documentation
% for educational and research purposes only and without fee is hereby
% granted, provided that this copyright notice and the original authors'
% names appear on all copies and supporting documentation. This program
% shall not be used, rewritten, or adapted as the basis of a commercial
% software or hardware product without first obtaining permission of the
% authors. The authors make no representations about the suitability of
% this software for any purpose. It is provided "as is" without express
% or implied warranty.
%----------------------------------------------------------------------
%
% This is an implementation of the algorithm for calculating the
% Structural SIMilarity (SSIM) index between two images
%
% Please refer to the following paper and the website with suggested usage
%
% Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image
% quality assessment: From error visibility to structural similarity,"
% IEEE Transactios on Image Processing, vol. 13, no. 4, pp. 600-612,
% Apr. 2004.
%
% http://www.ece.uwaterloo.ca/~z70wang/research/ssim/
%
% Note: This program is different from ssim_index.m, where no automatic
% downsampling is performed. (downsampling was done in the above paper
% and was described as suggested usage in the above website.)
```

```
%
% Kindly report any suggestions or corrections to zhouwang@ieee.org
%
%----------------------------------------------------------------------
%
%Input : (1) img1: the first image being compared
%        (2) img2: the second image being compared
%        (3) K: constants in the SSIM index formula (see the above
%            reference). defualt value: K = [0.01 0.03]
%        (4) window: local window for statistics (see the above
%            reference). default widnow is Gaussian given by
%            window = fspecial('gaussian', 11, 1.5);
%        (5) L: dynamic range of the images. default: L = 255
%
%Output: (1) mssim: the mean SSIM index value between 2 images.
%            If one of the images being compared is regarded as
%            perfect quality, then mssim can be considered as the
%            quality measure of the other image.
%            If img1 = img2, then mssim = 1.
%        (2) ssim_map: the SSIM index map of the test image. The map
%            has a smaller size than the input images. The actual size
%            depends on the window size and the downsampling factor.
%
%Basic Usage:
%   Given 2 test images img1 and img2, whose dynamic range is 0-255
%
%   [mssim, ssim_map] = ssim(img1, img2);
%
%Advanced Usage:
%   User defined parameters. For example
%
%   K = [0.05 0.05];
%   window = ones(8);
%   L = 100;
%   [mssim, ssim_map] = ssim(img1, img2, K, window, L);
%
%Visualize the results:
%
%   mssim                        %Gives the mssim value
%   imshow(max(0, ssim_map).^4)  %Shows the SSIM index map
%========================================================================


if (nargin < 2 || nargin > 5)
   mssim = -Inf;
   ssim_map = -Inf;
   return;
end

if (size(img1) ~= size(img2))
   mssim = -Inf;
   ssim_map = -Inf;
   return;
end

[M N] = size(img1);

if (nargin == 2)
   if ((M < 11) || (N < 11))
       mssim = -Inf;
       ssim_map = -Inf;
      return
   end
   window = fspecial('gaussian', 11, 1.5);  %
   K(1) = 0.01;                            % default settings
   K(2) = 0.03;                            %
   L = 255;                                                %
end
```

```
if (nargin == 3)
   if ((M < 11) || (N < 11))
       mssim = -Inf;
       ssim_map = -Inf;
      return
   end
   window = fspecial('gaussian', 11, 1.5);
   L = 255;
   if (length(K) == 2)
      if (K(1) < 0 || K(2) < 0)
           mssim = -Inf;
        ssim_map = -Inf;
        return;
      end
   else
       mssim = -Inf;
    ssim_map = -Inf;
       return;
   end
end

if (nargin == 4)
   [H W] = size(window);
   if ((H*W) < 4 || (H > M) || (W > N))
       mssim = -Inf;
       ssim_map = -Inf;
      return
   end
   L = 255;
   if (length(K) == 2)
      if (K(1) < 0 || K(2) < 0)
           mssim = -Inf;
        ssim_map = -Inf;
        return;
      end
   else
       mssim = -Inf;
    ssim_map = -Inf;
       return;
   end
end

if (nargin == 5)
   [H W] = size(window);
   if ((H*W) < 4 || (H > M) || (W > N))
       mssim = -Inf;
       ssim_map = -Inf;
      return
   end
   if (length(K) == 2)
      if (K(1) < 0 || K(2) < 0)
           mssim = -Inf;
        ssim_map = -Inf;
        return;
      end
   else
       mssim = -Inf;
    ssim_map = -Inf;
       return;
   end
end


img1 = double(img1);
img2 = double(img2);

% automatic downsampling
```

```
f = max(1,round(min(M,N)/256));
%downsampling by f
%use a simple low-pass filter
if(f>1)
    lpf = ones(f,f);
    lpf = lpf/sum(lpf(:));
    img1 = imfilter(img1,lpf,'symmetric','same');
    img2 = imfilter(img2,lpf,'symmetric','same');

    img1 = img1(1:f:end,1:f:end);
    img2 = img2(1:f:end,1:f:end);
end

C1 = (K(1)*L)^2;
C2 = (K(2)*L)^2;
window = window/sum(sum(window));

mu1   = filter2(window, img1, 'valid');
mu2   = filter2(window, img2, 'valid');
mu1_sq = mu1.*mu1;
mu2_sq = mu2.*mu2;
mu1_mu2 = mu1.*mu2;
sigma1_sq = filter2(window, img1.*img1, 'valid') - mu1_sq;
sigma2_sq = filter2(window, img2.*img2, 'valid') - mu2_sq;
sigma12 = filter2(window, img1.*img2, 'valid') - mu1_mu2;

if (C1 > 0 && C2 > 0)
   ssim_map = ((2*mu1_mu2 + C1).*(2*sigma12 + C2))./((mu1_sq + mu2_sq +
C1).*(sigma1_sq + sigma2_sq + C2));
else
   numerator1 = 2*mu1_mu2 + C1;
   numerator2 = 2*sigma12 + C2;
    denominator1 = mu1_sq + mu2_sq + C1;
   denominator2 = sigma1_sq + sigma2_sq + C2;
   ssim_map = ones(size(mu1));
   index = (denominator1.*denominator2 > 0);
   ssim_map(index) =
(numerator1(index).*numerator2(index))./(denominator1(index).*denominator2(i
ndex));
   index = (denominator1 ~= 0) & (denominator2 == 0);
   ssim_map(index) = numerator1(index)./denominator1(index);
end

mssim = mean2(ssim_map);

return
```

FSIM/FSIM$_C$ Evaluation Function (source [131])

```
function [FSIM, FSIMc] = FeatureSIM(imageRef, imageDis)
% ========================================================================
% FSIM Index with automatic downsampling, Version 1.0
% Copyright(c) 2010 Lin ZHANG, Lei Zhang, Xuanqin Mou and David Zhang
% All Rights Reserved.
%
% ----------------------------------------------------------------------
% Permission to use, copy, or modify this software and its documentation
% for educational and research purposes only and without fee is here
% granted, provided that this copyright notice and the original authors'
% names appear on all copies and supporting documentation. This program
% shall not be used, rewritten, or adapted as the basis of a commercial
% software or hardware product without first obtaining permission of the
% authors. The authors make no representations about the suitability of
% this software for any purpose. It is provided "as is" without express
% or implied warranty.
%----------------------------------------------------------------------
```

```
%
% This is an implementation of the algorithm for calculating the
% Feature SIMilarity (FSIM) index between two images.
%
% Please refer to the following paper
%
% Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang,"FSIM: a feature similarity
% index for image qualtiy assessment", IEEE Transactions on Image Processing, vol. 20, no. 8, pp. 2378-2386, 2011.
%
%----------------------------------------------------------------------
%
%Input : (1) imageRef: the first image being compared
%        (2) imageDis: the second image being compared
%
%Output: (1) FSIM: is the similarty score calculated using FSIM algorithm. FSIM
%        only considers the luminance component of images. For colorful images,
%            they will be converted to the grayscale at first.
%        (2) FSIMc: is the similarity score calculated using FSIMc algorithm. FSIMc
%            considers both the grayscale and the color information.
%Note: For grayscale images, the returned FSIM and FSIMc are the same.
%
%-----------------------------------------------------------------------
%
%Usage:
%Given 2 test images img1 and img2. For gray-scale images, their dynamic range should be 0-255.
%For colorful images, the dynamic range of each color channel should be 0-255.
%
%[FSIM, FSIMc] = FeatureSIM(img1, img2);
%-----------------------------------------------------------------------

[rows, cols] = size(imageRef(:,:,1));
I1 = ones(rows, cols);
I2 = ones(rows, cols);
Q1 = ones(rows, cols);
Q2 = ones(rows, cols);

if ndims(imageRef) == 3 %images are colorful
    Y1 = 0.299 * double(imageRef(:,:,1)) + 0.587 * double(imageRef(:,:,2)) + 0.114 * double(imageRef(:,:,3));
    Y2 = 0.299 * double(imageDis(:,:,1)) + 0.587 * double(imageDis(:,:,2)) + 0.114 * double(imageDis(:,:,3));
    I1 = 0.596 * double(imageRef(:,:,1)) - 0.274 * double(imageRef(:,:,2)) - 0.322 * double(imageRef(:,:,3));
    I2 = 0.596 * double(imageDis(:,:,1)) - 0.274 * double(imageDis(:,:,2)) - 0.322 * double(imageDis(:,:,3));
    Q1 = 0.211 * double(imageRef(:,:,1)) - 0.523 * double(imageRef(:,:,2)) + 0.312 * double(imageRef(:,:,3));
    Q2 = 0.211 * double(imageDis(:,:,1)) - 0.523 * double(imageDis(:,:,2)) + 0.312 * double(imageDis(:,:,3));
else %images are grayscale
    Y1 = imageRef;
    Y2 = imageDis;
end

Y1 = double(Y1);
Y2 = double(Y2);
%%%%%%%%%%%%%%%%%%%%%%%%%
% Downsample the image
%%%%%%%%%%%%%%%%%%%%%%%%%
minDimension = min(rows,cols);
F = max(1,round(minDimension / 256));
```

```
aveKernel = fspecial('average',F);

aveI1 = conv2(I1, aveKernel,'same');
aveI2 = conv2(I2, aveKernel,'same');
I1 = aveI1(1:F:rows,1:F:cols);
I2 = aveI2(1:F:rows,1:F:cols);

aveQ1 = conv2(Q1, aveKernel,'same');
aveQ2 = conv2(Q2, aveKernel,'same');
Q1 = aveQ1(1:F:rows,1:F:cols);
Q2 = aveQ2(1:F:rows,1:F:cols);

aveY1 = conv2(Y1, aveKernel,'same');
aveY2 = conv2(Y2, aveKernel,'same');
Y1 = aveY1(1:F:rows,1:F:cols);
Y2 = aveY2(1:F:rows,1:F:cols);

%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate the phase congruency maps
%%%%%%%%%%%%%%%%%%%%%%%%%
PC1 = phasecong2(Y1);
PC2 = phasecong2(Y2);

%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate the gradient map
%%%%%%%%%%%%%%%%%%%%%%%%%
dx = [3 0 -3; 10 0 -10;  3  0 -3]/16;
dy = [3 10 3; 0  0   0; -3 -10 -3]/16;
IxY1 = conv2(Y1, dx, 'same');
IyY1 = conv2(Y1, dy, 'same');
gradientMap1 = sqrt(IxY1.^2 + IyY1.^2);

IxY2 = conv2(Y2, dx, 'same');
IyY2 = conv2(Y2, dy, 'same');
gradientMap2 = sqrt(IxY2.^2 + IyY2.^2);

%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate the FSIM
%%%%%%%%%%%%%%%%%%%%%%%%%
T1 = 0.85;  %fixed
T2 = 160; %fixed
PCSimMatrix = (2 * PC1 .* PC2 + T1) ./ (PC1.^2 + PC2.^2 + T1);
gradientSimMatrix = (2*gradientMap1.*gradientMap2 + T2) ./(gradientMap1.^2 +
gradientMap2.^2 + T2);
PCm = max(PC1, PC2);
SimMatrix = gradientSimMatrix .* PCSimMatrix .* PCm;
FSIM = sum(sum(SimMatrix)) / sum(sum(PCm));

%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate the FSIMc
%%%%%%%%%%%%%%%%%%%%%%%%%
T3 = 200;
T4 = 200;
ISimMatrix = (2 * I1 .* I2 + T3) ./ (I1.^2 + I2.^2 + T3);
QSimMatrix = (2 * Q1 .* Q2 + T4) ./ (Q1.^2 + Q2.^2 + T4);

lambda = 0.03;

SimMatrixC = gradientSimMatrix .* PCSimMatrix .* real((ISimMatrix .*
QSimMatrix) .^ lambda) .* PCm;
FSIMc = sum(sum(SimMatrixC)) / sum(sum(PCm));

return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

function [ResultPC]=phasecong2(im)
```

```
% ========================================================================
% Copyright (c) 1996-2009 Peter Kovesi
% School of Computer Science & Software Engineering
% The University of Western Australia
% http://www.csse.uwa.edu.au/
%
% Permission is hereby  granted, free of charge, to any  person obtaining a copy
% of this software and associated  documentation files (the "Software"), to deal
% in the Software without restriction, subject to the following conditions:
%
% The above copyright notice and this permission notice shall be included in all
% copies or substantial portions of the Software.
%
% The software is provided "as is", without warranty of any kind.
% References:
%
%     Peter Kovesi, "Image Features From Phase Congruency". Videre: A
%     Journal of Computer Vision Research. MIT Press. Volume 1, Number 3,
%     Summer 1999 http://mitpress.mit.edu/e-journals/Videre/001/v13.html

nscale          = 4;     % Number of wavelet scales.
norient         = 4;     % Number of filter orientations.
minWaveLength   = 6;     % Wavelength of smallest scale filter.
mult            = 2;   % Scaling factor between successive filters.
sigmaOnf        = 0.55;  % Ratio of the standard deviation of the
                             % Gaussian describing the log Gabor filter's
                             % transfer function in the frequency domain
                             % to the filter center frequency.
dThetaOnSigma   = 1.2;   % Ratio of angular interval between filter orientations
                             % and the standard deviation of the angular Gaussian
                             % function used to construct filters in the
                             % freq. plane.
k               = 2.0;   % No of standard deviations of the noise
                             % energy beyond the mean at which we set the
                             % noise threshold point.
                             % below which phase congruency values get
                             % penalized.
epsilon         = .0001;                % Used to prevent division by zero.

thetaSigma = pi/norient/dThetaOnSigma;  % Calculate the standard deviation of the
                                        % angular Gaussian function used to
                                        % construct filters in the freq. plane.

[rows,cols] = size(im);
imagefft = fft2(im);              % Fourier transform of image

zero = zeros(rows,cols);
EO = cell(nscale, norient);       % Array of convolution results.

estMeanE2n = [];
ifftFilterArray = cell(1,nscale); % Array of inverse FFTs of filters

% Pre-compute some stuff to speed up filter construction

% Set up X and Y matrices with ranges normalised to +/- 0.5
% The following code adjusts things appropriately for odd and even values
% of rows and columns.
if mod(cols,2)
    xrange = [-(cols-1)/2:(cols-1)/2]/(cols-1);
else
    xrange = [-cols/2:(cols/2-1)]/cols;
```

```matlab
end

if mod(rows,2)
    yrange = [-(rows-1)/2:(rows-1)/2]/(rows-1);
else
    yrange = [-rows/2:(rows/2-1)]/rows;
end

[x,y] = meshgrid(xrange, yrange);

radius = sqrt(x.^2 + y.^2);       % Matrix values contain *normalised* radius from centre.
theta = atan2(-y,x);              % Matrix values contain polar angle.
                                  % (note -ve y is used to give +ve
                                  % anti-clockwise angles)

radius = ifftshift(radius);       % Quadrant shift radius and theta so that filters
theta  = ifftshift(theta);        % are constructed with 0 frequency at the corners.
radius(1,1) = 1;                  % Get rid of the 0 radius value at the 0
                                  % frequency point (now at top-left corner)
                                  % so that taking the log of the radius will
                                  % not cause trouble.

sintheta = sin(theta);
costheta = cos(theta);
clear x; clear y; clear theta;    % save a little memory

% Filters are constructed in terms of two components.
% 1) The radial component, which controls the frequency band that the filter
%    responds to
% 2) The angular component, which controls the orientation that the filter
%    responds to.
% The two components are multiplied together to construct the overall filter.

% Construct the radial filter components...

% First construct a low-pass filter that is as large as possible, yet falls
% away to zero at the boundaries.  All log Gabor filters are multiplied by
% this to ensure no extra frequencies at the 'corners' of the FFT are
% incorporated as this seems to upset the normalisation process when
% calculating phase congrunecy.
lp = lowpassfilter([rows,cols],.45,15);   % Radius .45, 'sharpness' 15

logGabor = cell(1,nscale);

for s = 1:nscale
    wavelength = minWaveLength*mult^(s-1);
    fo = 1.0/wavelength;                  % Centre frequency of filter.
    logGabor{s} = exp((-(log(radius/fo)).^2) / (2 * log(sigmaOnf)^2));
    logGabor{s} = logGabor{s}.*lp;        % Apply low-pass filter
    logGabor{s}(1,1) = 0;                 % Set the value at the 0 frequency point of the filter
                                          % back to zero (undo the radius fudge).
end

% Then construct the angular filter components...

spread = cell(1,norient);

for o = 1:norient
  angl = (o-1)*pi/norient;           % Filter angle.

  % For each point in the filter matrix calculate the angular distance from
  % the specified filter orientation.  To overcome the angular wrap-around
  % problem sine difference and cosine difference values are first computed
  % and then the atan2 function is used to determine angular distance.

  ds = sintheta * cos(angl) - costheta * sin(angl);    % Difference in sine.
  dc = costheta * cos(angl) + sintheta * sin(angl);    % Difference in
cosine.
  dtheta = abs(atan2(ds,dc));                          % Absolute angular
distance.
  spread{o} = exp((-dtheta.^2) / (2 * thetaSigma^2));  % Calculate the
                                                       % angular filter
component.
end

% The main loop...
EnergyAll(rows,cols) = 0;
AnAll(rows,cols) = 0;

for o = 1:norient                    % For each orientation.
  sumE_ThisOrient   = zero;          % Initialize accumulator matrices.
  sumO_ThisOrient   = zero;
  sumAn_ThisOrient  = zero;
  Energy            = zero;
  for s = 1:nscale,                  % For each scale.
    filter = logGabor{s} .* spread{o};   % Multiply radial and angular
                                         % components to get the filter.
    ifftFilt = real(ifft2(filter))*sqrt(rows*cols);  % Note rescaling to
match power
    ifftFilterArray{s} = ifftFilt;                   % record ifft2 of
filter
    % Convolve image with even and odd filters returning the result in EO
    EO{s,o} = ifft2(imagefft .* filter);

    An = abs(EO{s,o});                         % Amplitude of even & odd
filter response.
    sumAn_ThisOrient = sumAn_ThisOrient + An;  % Sum of amplitude responses.
    sumE_ThisOrient = sumE_ThisOrient + real(EO{s,o}); % Sum of even filter
convolution results.
    sumO_ThisOrient = sumO_ThisOrient + imag(EO{s,o}); % Sum of odd filter
convolution results.
    if s==1                                 % Record mean squared filter
value at smallest
      EM_n = sum(sum(filter.^2));           % scale. This is used for noise
estimation.
      maxAn = An;                           % Record the maximum An over all
scales.
    else
      maxAn = max(maxAn, An);
    end
  end                                       % ... and process the next scale

  % Get weighted mean filter response vector, this gives the weighted mean
  % phase angle.

  XEnergy = sqrt(sumE_ThisOrient.^2 + sumO_ThisOrient.^2) + epsilon;
  MeanE = sumE_ThisOrient ./ XEnergy;
  MeanO = sumO_ThisOrient ./ XEnergy;

  % Now calculate An(cos(phase_deviation) - | sin(phase_deviation)) | by
  % using dot and cross products between the weighted mean filter response
  % vector and the individual filter response vectors at each scale.  This
  % quantity is phase congruency multiplied by An, which we call energy.

  for s = 1:nscale,
      E = real(EO{s,o}); O = imag(EO{s,o});    % Extract even and odd
                                               % convolution results.
      Energy = Energy + E.*MeanE + O.*MeanO - abs(E.*MeanO - O.*MeanE);
  end

  % Compensate for noise
  % We estimate the noise power from the energy squared response at the
  % smallest scale.  If the noise is Gaussian the energy squared will have a
  % Chi-squared 2DOF pdf.  We calculate the median energy squared response
  % as this is a robust statistic.  From this we estimate the mean.
  % The estimate of noise power is obtained by dividing the mean squared
  % energy value by the mean squared filter value

  medianE2n = median(reshape(abs(EO{1,o}).^2,1,rows*cols));
  meanE2n = -medianE2n/log(0.5);
  estMeanE2n(o) = meanE2n;

  noisePower = meanE2n/EM_n;                       % Estimate of noise power.

  % Now estimate the total energy^2 due to noise
  % Estimate for sum(An^2) + sum(Ai.*Aj.*(cphi.*cphj + sphi.*sphj))

  EstSumAn2 = zero;
  for s = 1:nscale
    EstSumAn2 = EstSumAn2 + ifftFilterArray{s}.^2;
  end

  EstSumAiAj = zero;
  for si = 1:(nscale-1)
    for sj = (si+1):nscale
      EstSumAiAj = EstSumAiAj + ifftFilterArray{si}.*ifftFilterArray{sj};
    end
  end
  sumEstSumAn2 = sum(sum(EstSumAn2));
  sumEstSumAiAj = sum(sum(EstSumAiAj));

  EstNoiseEnergy2 = 2*noisePower*sumEstSumAn2 + 4*noisePower*sumEstSumAiAj;

  tau = sqrt(EstNoiseEnergy2/2);                     % Rayleigh parameter
  EstNoiseEnergy = tau*sqrt(pi/2);                   % Expected value of noise energy
  EstNoiseEnergySigma = sqrt( (2-pi/2)*tau^2 );

  T =  EstNoiseEnergy + k*EstNoiseEnergySigma;       % Noise threshold

  % The estimated noise effect calculated above is only valid for the PC_1 measure.
  % The PC_2 measure does not lend itself readily to the same analysis. However
  % empirically it seems that the noise effect is overestimated roughly by a factor
  % of 1.7 for the filter parameters used here.

  T = T/1.7;        % Empirical rescaling of the estimated noise effect to
                    % suit the PC_2 phase congruency measure
  Energy = max(Energy - T, zero);          % Apply noise threshold

  EnergyAll = EnergyAll + Energy;
  AnAll = AnAll + sumAn_ThisOrient;
end  % For each orientation
ResultPC = EnergyAll ./ AnAll;
return;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LOWPASSFILTER - Constructs a low-pass butterworth filter.
%
% usage: f = lowpassfilter(sze, cutoff, n)
%
% where: sze    is a two element vector specifying the size of filter
%               to construct [rows cols].
%        cutoff is the cutoff frequency of the filter 0 - 0.5
%        n      is the order of the filter, the higher n is the sharper
%               the transition is. (n must be an integer >= 1).
%               Note that n is doubled so that it is always an even integer.
%
%                      1
%      f =    --------------------
%                              2n
%              1.0 + (w/cutoff)
%
% The frequency origin of the returned filter is at the corners.
%
% See also: HIGHPASSFILTER, HIGHBOOSTFILTER, BANDPASSFILTER
%

% Copyright (c) 1999 Peter Kovesi
% School of Computer Science & Software Engineering
% The University of Western Australia
% http://www.csse.uwa.edu.au/
%
% Permission is hereby granted, free of charge, to any person obtaining a
copy
% of this software and associated documentation files (the "Software"), to
deal
% in the Software without restriction, subject to the following conditions:
%
% The above copyright notice and this permission notice shall be included in
% all copies or substantial portions of the Software.
%
% The Software is provided "as is", without warranty of any kind.

% October 1999
% August  2005 - Fixed up frequency ranges for odd and even sized filters
%                (previous code was a bit approximate)

function f = lowpassfilter(sze, cutoff, n)

    if cutoff < 0 || cutoff > 0.5
    error('cutoff frequency must be between 0 and 0.5');
    end

    if rem(n,1) ~= 0 || n < 1
    error('n must be an integer >= 1');
    end

    if length(sze) == 1
    rows = sze; cols = sze;
    else
    rows = sze(1); cols = sze(2);
    end

    % Set up X and Y matrices with ranges normalised to +/- 0.5
    % The following code adjusts things appropriately for odd and even
values
    % of rows and columns.
    if mod(cols,2)
    xrange = [-(cols-1)/2:(cols-1)/2]/(cols-1);
    else
    xrange = [-cols/2:(cols/2-1)]/cols;
    end

    if mod(rows,2)
    yrange = [-(rows-1)/2:(rows-1)/2]/(rows-1);
    else
    yrange = [-rows/2:(rows/2-1)]/rows;
    end

    [x,y] = meshgrid(xrange, yrange);
    radius = sqrt(x.^2 + y.^2);        % A matrix with every pixel = radius
relative to centre.
    f = ifftshift( 1 ./ (1.0 + (radius ./ cutoff).^(2*n)) );    % The filter
    return;
```

A.4 Test Bed Demosaicking Algorithm Blocks

Constant Difference Based Interpolation Algorithm (Author's Implementation)

```
%==========================================================================
%script:    constantDiffBasedInterpolation.m
%author:    Kinyua Wachira
%date:      --/07/2014
%desc:      a script that takes a Bayer filtered image and performs
%           an extension of bilinear interpolation called constant
%           difference based (CDB) interpolation on it
%==========================================================================

%utility functions
clc; clear all; close all;

%load image
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\USC-
SIPI\sipi_im16.tiff');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Kodak\kodim24.png');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\McM\mcm18.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Condat\codim04.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\ARRI\arri_im12.tif');
img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Custom1\cusim15.jpg');

%establish the Bayer CFA representation
imgBayer = img2Bayer(img);

%extract the colour components
imgRed = double(imgBayer(:,:,1));
imgGrn = double(imgBayer(:,:,2));
imgBlu = double(imgBayer(:,:,3));

pad = 0; %padding region width for interpolation error

%interpolate the green component
fG = [0 1 0; 1 4 1;0 1 0]/4;
imgGrnInt = conv2(fG,imgGrn,'full');

%interpolate the red component, by first establishing it's difference, then
%doing interpolation -- the steps follow
[R,C] = size(imgRed);
%set up a matrix to hold the image differences
imgRedDiff = double(zeros(R,C));
%populate the difference matrix
for i=1:1:R;
    for j=1:1:C;
        if (mod(i,2)==1 && mod(j,2)==1)
            imgRedDiff(i,j) = imgRed(i,j) - imgGrnInt(i+1,j+1);
        end;
    end;
end;
%interpolate the red component difference
fRB = [1 2 1; 2 4 2; 1 2 1]/4;
imgRedDiffInt = conv2(fRB,imgRedDiff,'full');
%establish the final red interpolation matrix
```

```
imgRedInt = imgRedDiffInt + imgGrnInt;

%interpolate the blue component, in a similar way to the red
[R,C] = size(imgBlu);
%set up a matrix to hold the image differences
imgBluDiff = double(zeros(R,C));
%populate the difference matrix
for i=1:1:R;
    for j=1:1:C;
        if (mod(i,2)==0 && mod(j,2)==0)
            imgBluDiff(i,j) = imgBlu(i,j) - imgGrnInt(i+1,j+1);
        end;
    end;
end;
%interpolate the blue component difference
fRB = [1 2 1; 2 4 2; 1 2 1]/4;
imgBluDiffInt = conv2(fRB,imgBluDiff,'full');
%establish the final blue interpolation matrix
imgBluInt = imgBluDiffInt + imgGrnInt;

%finally reconstruct the image from its constitutent colour components
[R,C] = size(imgGrnInt);
imgCDBInt = uint8(zeros(R,C,3));
imgCDBInt(:,:,1) = uint8(imgRedInt);
imgCDBInt(:,:,2) = uint8(imgGrnInt);
imgCDBInt(:,:,3) = uint8(imgBluInt);

%results
%imtool(img);
%imtool(imgBayer);
%imtool(imgCDBInt);

%preconditioning img before PSNR
[R,C,dim] = size(img);
imgCDB = double(zeros(R,C,dim));
for i=1:1:R;
    for j=1:1:C;
        for k=1:1:dim;
            imgCDB(i,j,k) = imgCDBInt(i+1,j+1,k);
        end;
    end;
end;

%pre-conditioning image to measure CIEDE2000 (if desired)
imgCrop = uint8(zeros(R-(2*pad),C-(2*pad),k));
imgCDBCrop = double(zeros(R-(2*pad),C-(2*pad),k));

for i=1:1:R-(2*pad);
    for j=1:1:C-(2*pad);
        for k=1:1:3;
            imgCrop(i,j,k) = img(i+pad,j+pad,k);
            imgCDBCrop(i,j,k) = imgCDB(i+pad,j+pad,k);
        end;
    end;
end;

%comparison measures
[PSNR_R, PSNR_G, PSNR_B] = measurePSNR(imgCDB,img,pad);
[CPSNR] = fcn_measureCPSNRv2(imgCDB,img,pad);
MSE = fcn_measureMSESinglev2(uint8(imgCDB(:,:,2)),uint8(img(:,:,2)),4);
[FSIM,FSIMc] = FeatureSIM(uint8(img),uint8(imgCDB));
SSIM = ssim(img,uint8(imgCDB));
PSNR_R;
PSNR_G;
PSNR_B;
CPSNR;
```

Edge Directed Interpolation Algorithm (Author’s Implementation)

```
%==========================================================================
%script:    edgeDirectedInterpolation.m
%author:    Kinyua Wachira
%date:      --/07/2014
%desc:      a script that takes a Bayer filtered image and performs
%           simple edge directed interpolation as shown in fig [3] of
%           gunturk article in 2005 Jan Issue of SPM
%==========================================================================

%utility functions
clc; clear all; close all;

%load image
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\USC-
SIPI\sipi_im16.tiff');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Kodak\kodim24.png');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\McM\mcm18.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Condat\codim30.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\ARRI\arri_im12.tif');
img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Custom1\cusim15.jpg');

pad = 3; %padding region width for interpolation error

%obtain the 3D (colour) bayer CFA representation
imgBayer3D = img2Bayer(img);
[R,C,k] = size(img);

%----------------------------
%work on the green component
%----------------------------
imgGrn = double(imgBayer3D(:,:,2));
[R,C] = size(imgGrn);
%populate the image edges first so that all inner missing values with hade
%sufficient neighbours for edge directed interpolation
imgGrnInt = imgGrn;
if (mod(R,2) == 0 && mod(C,2) == 0)
    imgGrnInt(1,1) = (imgGrn(2,1)+imgGrn(1,2))/2;
    imgGrn(R,C) = (imgGrn(R-1,C)+imgGrn(R,C-1))/2;
    for i=3:2:R-1; %left edge of image
        imgGrnInt(i,1) = (imgGrn(i-1,1)+imgGrn(i+1,1))/2;
    end;
    for i=2:2:R-2; %right edge of image
        imgGrnInt(i,C) = (imgGrn(i-1,C)+imgGrn(i+1,C))/2;
    end;
    for j=3:2:C-1; %top edge of image
        imgGrnInt(1,j) = (imgGrn(1,j-1)+imgGrn(1,j+1))/2;
    end;
    for j=2:2:C-2; %bottom edge of image
        imgGrnInt(R,j) = (imgGrn(R,j-1)+imgGrn(R,j+1))/2;
    end;
end;

%populate the green plane using the simple edge directed interpolation
%algorithm
for i=1:1:R;
    for j = 1:1:C;
        if (mod(i+j,2)==0 && (i>1 && i<R) && (j>1 && j<C))
            HD = abs(imgGrn(i,j-1) - imgGrn(i,j+1));
            VD = abs(imgGrn(i-1,j) - imgGrn(i+1,j));
            if (HD>VD)
                imgGrnInt(i,j) = (imgGrn(i-1,j) + imgGrn(i+1,j))/2;
            elseif (HD<VD)
                imgGrnInt(i,j) = (imgGrn(i,j-1) + imgGrn(i,j+1))/2;
            else
                imgGrnInt(i,j) = ( (imgGrn(i-1,j) + imgGrn(i+1,j)) + ...
                                (imgGrn(i,j-1) + imgGrn(i,j+1)) )/4;
            end;
        end;
    end;
end;

%----------------------------
%work on the red component
%----------------------------
imgRed = double(imgBayer3D(:,:,1));
imgRedInt = imgRed;
for i=1:2:R-1;%Red Horizontals
    for j=2:2:C-2;
        imgRedInt(i,j) = (imgRed(i,j-1) - imgGrnInt(i,j-1))/2 + ...
                         (imgRed(i,j+1) - imgGrnInt(i,j+1))/2 + ...
                         imgGrnInt(i,j);
    end;
end;

for i=2:2:R-2;%Red Verticals
    for j=1:2:C-1;
        imgRedInt(i,j) = (imgRed(i-1,j) - imgGrnInt(i-1,j))/2 + ...
                         (imgRed(i+1,j) - imgGrnInt(i+1,j))/2 + ...
                         imgGrnInt(i,j);
    end;
end;

for i=2:2:R-2; %Red in Blue pixel locations
    for j=2:2:C-2;
        imgRedInt(i,j) = (imgRed(i-1,j-1) - imgGrnInt(i-1,j-1))/4 + ...
                         (imgRed(i-1,j+1) - imgGrnInt(i-1,j+1))/4 + ...
                         (imgRed(i+1,j-1) - imgGrnInt(i+1,j-1))/4 + ...
                         (imgRed(i+1,j+1) - imgGrnInt(i+1,j+1))/4 + ...
                         imgGrnInt(i,j);
    end;
end;

%----------------------------
%work on the blue component
%----------------------------
imgBlu = double(imgBayer3D(:,:,3));
imgBluInt = imgBlu;
for i=2:2:R;%Blue Horizontals
    for j=3:2:C-1;
        imgBluInt(i,j) = (imgBlu(i,j-1) - imgGrnInt(i,j-1))/2 + ...
                         (imgBlu(i,j+1) - imgGrnInt(i,j+1))/2 + ...
                         imgGrnInt(i,j);
    end;
end;

for i=3:2:R-1;%Blue Verticals
    for j=2:2:C;
        imgBluInt(i,j) = (imgBlu(i-1,j) - imgGrnInt(i-1,j))/2 + ...
                         (imgBlu(i+1,j) - imgGrnInt(i+1,j))/2 + ...
                         imgGrnInt(i,j);
    end;
end;

for i=3:2:R-1; %Blue in Red pixel locations
    for j=3:2:C-1;
        imgBluInt(i,j) = (imgBlu(i-1,j-1) - imgGrnInt(i-1,j-1))/4 + ...
                         (imgBlu(i-1,j+1) - imgGrnInt(i-1,j+1))/4 + ...
                         (imgBlu(i+1,j-1) - imgGrnInt(i+1,j-1))/4 + ...
                         (imgBlu(i+1,j+1) - imgGrnInt(i+1,j+1))/4 + ...
```

```
                      imgGrnInt(i,j);
    end;
end;

%finally reconstruct the entire image from its interpolated planes
imgEdgeInt(:,:,1) = uint8(imgRedInt);
imgEdgeInt(:,:,2) = uint8(imgGrnInt);
imgEdgeInt(:,:,3) = uint8(imgBluInt);

%results
%imtool(img);
%imtool(imgBayer3D);
%imtool(uint8(imgGrnInt));
%imtool(uint8(imgRedInt));
%imtool(uint8(imgBluInt));
%imtool(imgEdgeInt);

%pre-conditioning image to measure CIEDE2000
imgCrop = uint8(zeros(R-(2*pad),C-(2*pad),k));
imgEdgeIntCrop = double(zeros(R-(2*pad),C-(2*pad),k));

for i=1:1:R-(2*pad);
    for j=1:1:C-(2*pad);
        for k=1:1:3;
            imgCrop(i,j,k) = img(i+pad,j+pad,k);
            imgEdgeIntCrop(i,j,k) = imgEdgeInt(i+pad,j+pad,k);
        end;
    end;
end;

% [PSNR_R, PSNR_G, PSNR_B] = measurePSNR(imgEdgeInt,img,pad);
% [CPSNR] = measureCPSNR(imgEdgeInt,img,pad);
% PSNR_R;
% PSNR_G;
% PSNR_B;
% CPSNR;

MSE = fcn_measureMSESinglev2(uint8(imgEdgeInt(:,:,2)),uint8(img(:,:,2)),4);
[FSIM,FSIMc] = FeatureSIM(uint8(img),uint8(imgEdgeInt));
CPSNR = fcn_measureCPSNRv2(uint8(imgEdgeInt),uint8(img),4);
SSIM = ssim(img,uint8(imgEdgeInt));
%notes:
%1. this algorithm is too dependent on image dimensions especially when
%   establishing the green component. I had to establish corner points before
%   I could even start interpolating. Can this be abstracted/ generalised?
%2. I did not bother with interpolating some edge areas in the blue and red
%   images as there is insufficient information to interpolate - this led to
%   some color artefacts at the edges of the image, however these errors are
%   1 pixel wide
%3. from subjective observation, the results here mirrored the images seen
%   in gurturk's paper 2005 Jan SPM issue
%4. the algorithm can be found in its entirety in the Laroche Prescott Patent
%5. there also seems to be a washout effect this algorithm is performing -
%   colours are less vibrant than in the original


%NOTE WELL
%there was a time PSNR Blue was <<< other PSNR, this was because somewhere
%I forgot to define blue matrix as type double/ it was approximating
%everything to uint8... be very careful about datatype management... use
%double always when calculating
```

The Malvar-He-Cutler Algorithm (Author's Implementation)

```
%===========================================================================
% script:   malvar_HE_CutlerMethod
% date:     --/08/2014
% author:   Kinyua Wachira
% desc:     a script to perform the Malvar-He-Cutler demosaicking technique
%===========================================================================

%utility fcns
%clc; clear all; close all;

%load image
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\USC-SIPI\sipi_im16.tiff');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\Kodak\kodim24.png');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\McM\mcm18.tif');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\Condat\codim30.tif');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\ARRI\arri_im12.tif');
img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\Custom1\cusim15.jpg');


pad = 2; %padding region width for interpolation error

%obtain 3D (color) Bayer image
imgBayer3D = img2Bayer(img);

%obtain the grayscale equivalent
[R,C,k] = size(imgBayer3D);
imgBayer = double(zeros(R,C));

for i=1:1:R;
    for j=1:1:C;
        if(mod(i,2)==1 && mod(j,2)==1)
            imgBayer(i,j) = imgBayer3D(i,j,1);
        elseif (mod(i+j,2)==1)
            imgBayer(i,j) = imgBayer3D(i,j,2);
        else
            imgBayer(i,j) = imgBayer3D(i,j,3);
        end;
    end;
end;

%================
%green component
%================
imgGrn=double(zeros(R,C));
%green in green pixel locations
for i=1:1:R;
    for j=1:1:C;
        if(mod(i+j,2)==1)
            imgGrn(i,j) = imgBayer(i,j);
        end;
    end;
end;
%green in red/blue pixel locations
for i=1+2:1:R-2;
    for j=1+2:1:C-2;
        if (mod(i+j,2)~=1)
            imgGrn(i,j) = 0.125.* (-1*imgBayer(i-2,j) + 2*imgGrn(i-1,j) + ...
                -1*imgBayer(i,j-2) + 2*imgGrn(i,j-1) + 4*imgBayer(i,j) + ...
                2*imgGrn(i,j+1) - 1*imgBayer(i,j+2) + ...
                2*imgGrn(i+1,j) -1*imgBayer(i+2,j) );
        end;
    end;
```

```
end;

%=============
%red component
%=============
imgRed=double(zeros(R,C));
%red in red pixel locations
for i=1:1:R;
    for j=1:1:C;
        if(mod(i,2)==1 && mod(j,2)==1)
            imgRed(i,j) = imgBayer(i,j);
        end;
    end;
end;
%red in green and blue pixel locations
for i=1+2:1:R-2;
    for j=1+2:1:C-2;
        if (mod(i+j,2)==1 && mod(i,2)==1)
            imgRed(i,j) = 0.125.* (0.5*imgGrn(i-2,j) ...
                - 1*imgGrn(i-1,j-1) - 1*imgGrn(i-1,j+1) ...
                - 1*imgGrn(i,j-2) + 4*imgBayer(i,j-1) + 5*imgGrn(i,j) ...
                + 4*imgBayer(i,j+1) - 1*imgGrn(i,j+2) ...
                - 1*imgGrn(i+1,j-1) - 1*imgGrn(i+1,j+1) ...
                + 0.5*imgGrn(i+2,j) );
        elseif (mod(i+j,2)==1 && mod(i,2)==0)
            imgRed(i,j) = 0.125.* (-1*imgGrn(i-2,j) ...
                - 1*imgGrn(i-1,j-1) + 4*imgBayer(i-1,j) - 1*imgGrn(i-1,j+1) ...
                + 0.5*imgGrn(i,j-2) + 5*imgGrn(i,j) + 0.5*imgGrn(i,j+2) + ...
                - 1*imgGrn(i+1,j-1) + 4*imgBayer(i+1,j) - 1*imgGrn(i+1,j+1) ...
                - 1*imgGrn(i+2,j) );
        elseif (mod(i,2)==0 && mod(j,2)==0)
            imgRed(i,j) = 0.125.* (-1.5*imgBayer(i-2,j) ...
                + 2*imgBayer(i-1,j-1) + 2*imgBayer(i-1,j+1) ...
                - 1.5*imgBayer(i,j-2) + 6*imgBayer(i,j) - 1.5*imgBayer(i,j+2) + ...
                + 2*imgBayer(i+1,j-1) + 2*imgBayer(i+1,j+1) ...
                - 1.5*imgBayer(i+2,j) );
        end;
    end;
end;

%=============
%blue component
%=============
imgBlu=double(zeros(R,C));
%blue in blue pixel locations
for i=1:1:R;
    for j=1:1:C;
        if(mod(i,2)==0 && mod(j,2)==0)
            imgBlu(i,j) = imgBayer(i,j);
        end;
    end;
end;
%blue in green and red pixel locations
for i=1+2:1:R-2;
    for j=1+2:1:C-2;
        if (mod(i+j,2)==1 && mod(i,2)==0)
            imgBlu(i,j) = 0.125.* (0.5*imgGrn(i-2,j) ...
                - 1*imgGrn(i-1,j-1) - 1*imgGrn(i-1,j+1) ...
                - 1*imgGrn(i,j-2) + 4*imgBayer(i,j-1) + 5*imgGrn(i,j) ...
                + 4*imgBayer(i,j+1) - 1*imgGrn(i,j+2) ...
                - 1*imgGrn(i+1,j-1) - 1*imgGrn(i+1,j+1) ...
                + 0.5*imgGrn(i+2,j) );
        elseif (mod(i+j,2)==1 && mod(i,2)==1)
            imgBlu(i,j) = 0.125.* (-1*imgGrn(i-2,j) ...
                - 1*imgGrn(i-1,j-1) + 4*imgBayer(i-1,j) - 1*imgGrn(i-1,j+1) ...
                + 0.5*imgGrn(i,j-2) + 5*imgGrn(i,j) + 0.5*imgGrn(i,j+2) + ...
                - 1*imgGrn(i+1,j-1) + 4*imgBayer(i+1,j) - 1*imgGrn(i+1,j+1) ...
                - 1*imgGrn(i+2,j) );
        elseif (mod(i,2)==1 && mod(j,2)==1)
            imgBlu(i,j) = 0.125.* (-1.5*imgBayer(i-2,j) ...
                + 2*imgBayer(i-1,j-1) + 2*imgBayer(i-1,j+1) ...
                - 1.5*imgBayer(i,j-2) + 6*imgBayer(i,j) - 1.5*imgBayer(i,j+2) + ...
                + 2*imgBayer(i+1,j-1) + 2*imgBayer(i+1,j+1) ...
                - 1.5*imgBayer(i+2,j) );
        end;
    end;
end;

%=====================
%final reconstruction
%=====================
imgMCH = zeros(R,C,k);

imgMCH(:,:,1) = imgRed;
imgMCH(:,:,2) = imgGrn;
imgMCH(:,:,3) = imgBlu;


%results
%imtool(img);
%imtool(imgBayer3D);
%imtool(uint8(imgBayer));
%imtool(uint8(imgGrn));
%imtool(uint8(imgRed));
%imtool(uint8(imgBlu));
%imtool(uint8(imgMCH));

%pre-conditioning image to measure CIEDE2000 (if desired)
imgCrop = uint8(zeros(R-(2*pad),C-(2*pad),k));
imgMCHCrop = double(zeros(R-(2*pad),C-(2*pad),k));

for i=1:1:R-(2*pad);
    for j=1:1:C-(2*pad);
        for k=1:1:3;
            imgCrop(i,j,k) = img(i+pad,j+pad,k);
            imgMCHCrop(i,j,k) = imgMCH(i+pad,j+pad,k);
        end;
    end;
end;

%[PSNR_R, PSNR_G, PSNR_B] = measurePSNR(imgMCH,img,pad);
%[CPSNR] = measureCPSNR(imgMCH,img,pad);
%SSIM = ssim(img,uint8(imgMCH));

MSE = fcn_measureMSESinglev2(uint8(imgMCH(:,:,2)),uint8(img(:,:,2)),4);
[FSIM,FSIMc] = FeatureSIM(uint8(img),uint8(imgMCH));
CPSNR = fcn_measureCPSNRv2(uint8(imgMCH),uint8(img),4);
SSIM = ssim(img,uint8(imgMCH));

%PSNR_R;
%PSNR_G;
%PSNR_B;
%CPSNR;
```

The Edge Strength Filter Based Interpolation Algorithm (Author’s Implementation)

```
%==========================================================================
%   Name:       ESFBI_v1.m
%   Author:     Kinyua Wachira
%   Date:       24/09/2014
%   Desc:       an implementation of the Edge Strength Filter Based
%               Interpolation technique from seen in IEEE TIP v21 n1 2012
%               (05770218)
%
%   Notes:     did not convert dmap to d'map because I did not understand
%   how the neighbours are generated dmap is quincunx in nature
%
%   25/9 Almost done just need to find out why the R,B plane is a bit low
%   in the image quality
%   Finished the ESFBI algorithm, used W =0.5 (non-adaptive) and only one
%   updation loop. And a border size pad of 4 pixels
%==========================================================================

%utility fcns
clc; clear all; close all hidden;

%load image and establish Bayer CFA pattern
%load image
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\USC-
SIPI\sipi_im16.tiff');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Kodak\kodim24.png');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\McM\mcm18.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Condat\codim30.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\ARRI\arri_im12.tif');
img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Custom1\cusim15.jpg');

imgBayer = fcn_bayer(img);

%==========================================================================
% Preamble
%==========================================================================
imgBayer = double(imgBayer);
[R,C] = size(imgBayer);
[imgRed,imgGrn,imgBlu] = deal(double(zeros(R,C)));
imgESFBI = deal(double(zeros(R,C,3)));
padBorder = 4;
%populate Red, Green and Blue planes with information from Bayer (what is
%known)
for i = 1:1:R;
    for j=1:1:C;
        if (mod(i,2)==1 && mod(j,2)==1) %red
            imgRed(i,j) = imgBayer(i,j);
        elseif (mod(i+j,2)==1) %green
            imgGrn(i,j) = imgBayer(i,j);
        else %blue
            imgBlu(i,j) = imgBayer(i,j);
        end;
    end;
end;

%==========================================================================
% Generate the S map - that is the edge strength map %border size of 1
%==========================================================================
Smap = double(zeros(R,C));

for i=1+1:1:R-1;
    for j=1+1:1:C-1;
        Smap(i,j) = abs(imgBayer(i-1,j-1) - imgBayer(i+1,j+1))/2 ...
                  + abs(imgBayer(i-1,j+1) - imgBayer(i+1,j-1))/2 ...
                  + abs(imgBayer(i-1,j) - imgBayer(i+1,j)) ...
                  + abs(imgBayer(i,j-1) - imgBayer(i,j+1));
    end;
end;


%=============================================================================
% Green Plane Interpolation
%=============================================================================
%generate the Hmap and Vmap in the R,B regions
[Hmap,Vmap,dmap] = deal(double(zeros(R,C)));

for i=1+2:1:R-2;
    for j=1+2:1:C-2;
        if ~(mod(i+j,2)==1)
            Hmap(i,j) = sum(sum(Smap(i-2:i+2,j-2:j+1)-Smap(i-2:i+2,j-2+1:j+1+1)));
            Vmap(i,j) = sum(sum(Smap(i-2:i+1,j-2:j+2)-Smap(i-2+1:i+1+1,j-2:j+2)));
        end;
    end;
end;

%generate the dmap
%here 0 means invalid location, 128 means H, 255 means V
for i=1+2:1:R-2;
    for j=1+2:1:C-2;
        if ~(mod(i+j,2)==1)
            if (Hmap(i,j)<Vmap(i,j))
                dmap(i,j) = 128;
            else
                dmap(i,j) = 255;
            end;
        end;
    end;
end;

%perform the interpolation
for i=1+3:1:R-3;
    for j=1+3:1:C-3;
        if ~(mod(i+j,2)==1)
            %directional estimations
            GH = (imgGrn(i,j-1) + imgGrn(i,j+1))/2 + ...
                 (2*imgBayer(i,j) - imgBayer(i,j-2) - imgBayer(i,j+2))/4;

            GV = (imgGrn(i-1,j) + imgGrn(i+1,j))/2 + ...
                 (2*imgBayer(i,j) - imgBayer(i-2,j) - imgBayer(i+2,j))/4;

            BHneg = (imgBayer(i,j-2) + imgBayer(i,j))/2 + ...
                 (2*imgGrn(i,j-1) - imgGrn(i,j-3) - imgGrn(i,j+1))/4;

            BHpos = (imgBayer(i,j) + imgBayer(i,j+2))/2 + ...
                 (2*imgGrn(i,j+1) - imgGrn(i,j-1) - imgGrn(i,j+3))/4;

            BVneg = (imgBayer(i-2,j) + imgBayer(i,j))/2 + ...
                 (2*imgGrn(i-1,j) - imgGrn(i-3,j) - imgGrn(i+1,j))/4;

            BVpos = (imgBayer(i,j) + imgBayer(i+2,j))/2 + ...
                 (2*imgGrn(i+1,j) - imgGrn(i-1,j) - imgGrn(i+3,j))/4;

            if (dmap(i,j)==128)
                imgGrn(i,j) = imgBayer(i,j) + (GH - imgBayer(i,j))/2 ...
                    + (imgGrn(i,j-1) - BHneg)/4 + (imgGrn(i,j+1) - BHpos)/4;
            elseif (dmap(i,j)==255)
                imgGrn(i,j) = imgBayer(i,j) + (GV - imgBayer(i,j))/2 ...
                    + (imgGrn(i-1,j) - BVneg)/4 + (imgGrn(i+1,j) - BVpos)/4;
            end;
        end;
    end;
```

```
end;


%doing the green channel update
c = 0.1;
W = 0.5;

for i=1+3:1:R-3;
    for j=1+3:1:C-3;
        if ~(mod(i+j,2)==1)
            %set up the update parameters
            D1 = abs(Smap(i,j) - Smap(i-1,j)) + abs(Smap(i-1,j) - Smap(i-2,j)) +...
                 abs(Smap(i-2,j) - Smap(i-3,j)) +c;
            D2 = abs(Smap(i,j) - Smap(i,j-1)) + abs(Smap(i,j-1) - Smap(i,j-2)) +...
                 abs(Smap(i,j-2) - Smap(i,j-3)) +c;
            D3 = abs(Smap(i,j) - Smap(i,j+1)) + abs(Smap(i,j+1) - Smap(i,j+2)) +...
                 abs(Smap(i,j+2) - Smap(i,j+3)) +c;
            D4 = abs(Smap(i,j) - Smap(i+1,j)) + abs(Smap(i+1,j) - Smap(i+2,j)) +...
                 abs(Smap(i+2,j) - Smap(i+3,j)) +c;

            M1 = D2.*D3.*D4;
            M2 = D1.*D3.*D4;
            M3 = D1.*D2.*D4;
            M4 = D1.*D2.*D2;
            MT = M1+M2+M3+M4;

            gUP = imgBayer(i,j) + ...
                W.*(imgGrn(i,j) - imgBayer(i,j)) + ...
                (1-W).*( (M1/MT).*(imgGrn(i-2,j) - imgBayer(i-2,j)) ...
                       + (M2/MT).*(imgGrn(i,j-2) - imgBayer(i,j-2)) ...
                       + (M3/MT).*(imgGrn(i,j+2) - imgBayer(i,j+2)) ...
                       + (M4/MT).*(imgGrn(i+2,j) - imgBayer(i+2,j)) );


            imgGrn(i,j) = gUP;
        end;
    end;
end;

%=========================================================================
% Red and Blue Plane Interpolation
%=========================================================================
%in the opposing planes
e=0; %needed to add this to prevent the regions of homogeneity from giving
%a NaN error

for i=1+2:1:R-2;
    for j=1+2:1:C-2;
        if (mod(i+j,2)==0) %red in blue pixels and blue in red pixels
            M1 = abs(imgGrn(i-2,j-2) - imgGrn(i,j)) + ...
                 abs(imgGrn(i-1,j-1) - imgGrn(i+1,j+1)) + ...
                 abs(imgGrn(i,j) - imgGrn(i+2,j+2))+e;
            M2 = abs(imgGrn(i-2,j+2) - imgGrn(i,j)) + ...
                 abs(imgGrn(i-1,j+1) - imgGrn(i+1,j-1)) + ...
                 abs(imgGrn(i,j) - imgGrn(i+2,j-2))+e;

            if (mod(i,2)==1 && mod(j,2)==1) %red pixel location
                imgBlu(i,j) = imgGrn(i,j) ...
                - ((M2./(2.*(M1+M2))).*( (imgGrn(i-1,j-1)-imgBlu(i-1,j-1)) + (imgGrn(i+1,j+1)-imgBlu(i+1,j+1)) )) ...
                - ((M1./(2.*(M1+M2))).*( (imgGrn(i-1,j+1)-imgBlu(i-1,j+1)) + (imgGrn(i+1,j-1)-imgBlu(i+1,j-1)) ));
            end;
            if (mod(i,2)==0 && mod(j,2)==0) % blue pixel location
```

```
                    imgRed(i,j) = imgGrn(i,j) ...
                    - ((M2./(2.*(M1+M2))).*( (imgGrn(i-1,j-1)-imgRed(i-1,j-1)) + 
(imgGrn(i+1,j+1)-imgRed(i+1,j+1)) )) ...
                    - ((M1./(2.*(M1+M2))).*( (imgGrn(i-1,j+1)-imgRed(i-1,j+1)) + 
(imgGrn(i+1,j-1)-imgRed(i+1,j-1)) ));
                end;
            end;
        end;
    end;

    %in the green plane
    for i=1+2:1:R-2;
        for j=1+2:1:C-2;
            if (mod(i+j,2)==1) %green pixel locations
                if (mod(i,2)==1) %red rows/blue cols
                    imgRed(i,j) = imgGrn(i,j) - ...
                        ((imgGrn(i,j-1)-imgRed(i,j-1))+(imgGrn(i,j+1)-
imgRed(i,j+1)))/2;
                    imgBlu(i,j) = imgGrn(i,j) - ...
                        ((imgGrn(i-1,j)-imgBlu(i-1,j))+(imgGrn(i+1,j)-
imgBlu(i+1,j)))/2;
                end;
                if (mod(i,2)==0) %blue rows/red cols
                    imgRed(i,j) = imgGrn(i,j) - ...
                        ((imgGrn(i-1,j)-imgRed(i-1,j))+(imgGrn(i+1,j)-
imgRed(i+1,j)))/2;
                    imgBlu(i,j) = imgGrn(i,j) - ...
                        ((imgGrn(i,j-1)-imgBlu(i,j-1))+(imgGrn(i,j+1)-
imgBlu(i,j+1)))/2;
                end;
            end;
        end;
    end;


    %=========================================================================
    % Results
    %=========================================================================
    imgESFBI(:,:,1) = imgRed;
    imgESFBI(:,:,2) = imgGrn;
    imgESFBI(:,:,3) = imgBlu;

    % imtool(img);
    % imtool(uint8(imgBayer));
    % imtool(uint8(imgRed));
    % imtool(uint8(imgGrn));
    % imtool(uint8(imgBlu));
    % imtool(uint8(imgESFBI));
    %imtool(uint8(dmap));

    % [PSNR(1),PSNR(2),PSNR(3)] = 
    fcn_measurePSNRv2(uint8(imgESFBI),img,padBorder);
    % PSNR = PSNR';


    MSE = fcn_measureMSESinglev2(uint8(imgESFBI(:,:,2)),uint8(img(:,:,2)),4);
    [FSIM,FSIMc] = FeatureSIM(uint8(img),uint8(imgESFBI));
    CPSNR = fcn_measureCPSNRv2(uint8(imgESFBI),uint8(img),4);
    SSIM = ssim(img,uint8(imgESFBI));
```

The Wang Algorithm (Author's Implementation)

```
    %=========================================================================
    % script:   wangMethod
    % date:     --/--/2015
    % author:   Kinyua Wachira
```

```
% desc:     a script to perform the Wang demosaicking technique from SPIE
%           v8420 of 2012
%=============================================================================

%utility functions
clc; clear all; close all;

%load image and establish Bayer CFA pattern
%load image
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\USC-
SIPI\sipi_im16.tiff');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Kodak\kodim24.png');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\McM\mcm18.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Condat\codim30.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\ARRI\arri_im12.tif');
img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Custom1\cusim15.jpg');


pad = 2; %padding region width for interpolation error

%obtain the 3D (colour) Bayer CFA representation
imgBayer3D = img2Bayer(img);

%convert 3D Bayer to 2D (grayscale) equivalent
[R,C,k] = size(imgBayer3D);
imgBayer = double(zeros(R,C));
for i=1:1:R;
    for j=1:1:C;
        if (mod(i+j,2)==1)
            imgBayer(i,j) = imgBayer3D(i,j,2);
        elseif (mod(i,2)==1 && mod(j,2)==1)
            imgBayer(i,j) = imgBayer3D(i,j,1);
        else
            imgBayer(i,j) = imgBayer3D(i,j,3);
        end;
    end;
end;

%=========================
%green plane interpolation
%=========================
imgGrnEst = double(zeros(R,C));
imgGrn = double(zeros(R,C));

for i=1:1:R;
    for j=1:1:C;
        if (mod(i+j,2)==1)
            imgGrnEst(i,j) = imgBayer(i,j);
            imgGrn(i,j) = imgBayer(i,j);
        end;
    end;
end;

%1st interpolation
for i=1+2:1:R-2;
    for j=1+2:1:C-2;
        if (mod(i+j,2)~=1)
            Hgrad = abs(imgBayer(i,j-2)-imgBayer(i,j)) +
abs(imgBayer(i,j+2)-imgBayer(i,j))...
                + abs(imgGrn(i,j-1)-imgGrn(i,j+1));
            Vgrad = abs(imgBayer(i-2,j)-imgBayer(i,j)) +
abs(imgBayer(i+2,j)-imgBayer(i,j))...
                + abs(imgGrn(i-1,j)-imgGrn(i+1,j));
            if (Hgrad < Vgrad)
```

```
                imgGrnEst(i,j) = (imgGrn(i,j-1)+imgGrn(i,j+1))/2 + ...
                  (2.*imgBayer(i,j)-imgBayer(i,j-2)-imgBayer(i,j+2))/4;
            else
                imgGrnEst(i,j) = (imgGrn(i-1,j)+imgGrn(i+1,j))/2 + ...
                  (2.*imgBayer(i,j)-imgBayer(i-2,j)-imgBayer(i+2,j))/4;
            end;
        end;

    end;
end;

%2nd interpolation
for i=1+2:1:R-2;
    for j=1+2:1:C-2;
        if (mod(i+j,2)~=1)
            Hgrad = abs(imgBayer(i,j-2)-imgBayer(i,j)) +
abs(imgBayer(i,j+2)-imgBayer(i,j))...
                + abs(imgGrnEst(i,j-1)-imgGrnEst(i,j+1));
            Vgrad = abs(imgBayer(i-2,j)-imgBayer(i,j)) +
abs(imgBayer(i+2,j)-imgBayer(i,j))...
                + abs(imgGrnEst(i-1,j)-imgGrnEst(i+1,j));
            Ngrad = abs(imgBayer(i-2,j-2)-imgBayer(i,j)) +
abs(imgBayer(i+2,j+2)-imgBayer(i,j))...
                + abs(imgGrnEst(i-1,j-1)-imgGrnEst(i+1,j+1));
            Pgrad = abs(imgBayer(i+2,j-2)-imgBayer(i,j)) + abs(imgBayer(i-
2,j+2)-imgBayer(i,j))...
                + abs(imgGrnEst(i-1,j+1)-imgGrnEst(i+1,j-1));

            grad = [Hgrad,Vgrad,Ngrad,Pgrad];
            Thres = min(grad);

            if (Thres==Hgrad)
                imgGrn(i,j) = (imgGrnEst(i,j-1) + imgGrnEst(i,j+1))/2 ...
                    + (2*imgBayer(i,j)-imgBayer(i,j-2)-imgBayer(i,j+2))/4;
            elseif (Thres==Vgrad)
                imgGrn(i,j) = (imgGrnEst(i-1,j) + imgGrnEst(i+1,j))/2 ...
                    + (2*imgBayer(i,j)-imgBayer(i-2,j)-imgBayer(i+2,j))/4;
            elseif (Thres==Ngrad)
                imgGrn(i,j) = (imgGrnEst(i-1,j-1) + imgGrnEst(i+1,j+1))/2
...
                    + (2*imgBayer(i,j)-imgBayer(i-2,j-2)-
imgBayer(i+2,j+2))/4;
            else
                imgGrn(i,j) = (imgGrnEst(i-1,j+1) + imgGrnEst(i+1,j-1))/2
...
                    + (2*imgBayer(i,j)-imgBayer(i-2,j+2)-imgBayer(i+2,j-
2))/4;
            end;
        end;
    end;
end;

%========================
%red plane interpolation
%========================
imgRed = double(zeros(R,C));

%red in red pixel locations
for i=1:1:R;
    for j=1:1:C;
        if (mod(i,2)==1 && mod(j,2)==1)
            imgRed(i,j) = imgBayer(i,j);
        end;
    end;
end;

%red in green pixel locations
for i=1+1:1:R-1;
```

```
    for j=1+1:1:C-1;
        if (mod(i,2)==1 && mod(i+j,2)==1)
            imgRed(i,j) = imgGrn(i,j) + ...
                (imgRed(i,j-1)-imgGrn(i,j-1) + imgRed(i,j+1)-imgGrn(i,j+1))/2;
        end;
        if (mod(i,2)==0 && mod(i+j,2)==1)
            imgRed(i,j) = imgGrn(i,j) + ...
                (imgRed(i-1,j)-imgGrn(i-1,j) + imgRed(i+1,j)-imgGrn(i+1,j))/2;
        end;
    end;
end;

%red in blue pixel locations
for i=1+1:1:R-1;
    for j=1+1:1:C-1;
        if(mod(i,2)==0 && mod(j,2)==0)
            imgRed(i,j) = imgGrn(i,j) + ...
               ( imgRed(i-1,j-1)-imgGrn(i-1,j-1) + imgRed(i-1,j+1)-imgGrn(i-1,j+1) + ...
                 imgRed(i+1,j-1)-imgGrn(i+1,j-1) + imgRed(i+1,j+1)-imgGrn(i+1,j+1) )/4;
        end;
    end;
end;
%=========================
%blue plane interpolation
%=========================
imgBlu = double(zeros(R,C));

%blue in blue pixel locations
for i=1:1:R;
    for j=1:1:C;
        if (mod(i,2)==0 && mod(j,2)==0)
            imgBlu(i,j) = imgBayer(i,j);
        end;
    end;
end;

%blue in green pixel locations
for i=1+1:1:R-1;
    for j=1+1:1:C-1;
        if (mod(i,2)==0 && mod(i+j,2)==1)
            imgBlu(i,j) = imgGrn(i,j) + ...
                (imgBlu(i,j-1)-imgGrn(i,j-1) + imgBlu(i,j+1)-imgGrn(i,j+1))/2;
        end;
        if (mod(i,2)==1 && mod(i+j,2)==1)
            imgBlu(i,j) = imgGrn(i,j) + ...
                (imgBlu(i-1,j)-imgGrn(i-1,j) + imgBlu(i+1,j)-imgGrn(i+1,j))/2;
        end;
    end;
end;

%blue in red pixel locations
for i=1+1:1:R-1;
    for j=1+1:1:C-1;
        if(mod(i,2)==1 && mod(j,2)==1)
            imgBlu(i,j) = imgGrn(i,j) + ...
               ( imgBlu(i-1,j-1)-imgGrn(i-1,j-1) + imgBlu(i-1,j+1)-imgGrn(i-1,j+1) + ...
                 imgBlu(i+1,j-1)-imgGrn(i+1,j-1) + imgBlu(i+1,j+1)-imgGrn(i+1,j+1) )/4;
        end;
    end;
end;
```

```

%====================
%final reconstruction
%====================
imgWang = zeros(R,C,k);
imgWang(:,:,1) = imgRed;
imgWang(:,:,2) = imgGrn;
imgWang(:,:,3) = imgBlu;

%results
%imtool(img);
%imtool(imgBayer3D);
%imtool(uint8(imgBayer));
%imtool(uint8(imgGrnEst));
%imtool(uint8(imgGrn));
%imtool(uint8(imgRed));
%imtool(uint8(imgBlu));
%imtool(uint8(imgWang));

%pre-conditioning image to measure CIEDE2000 (if desired)
 imgCrop = uint8(zeros(R-(2*pad),C-(2*pad),k));
 imgWangCrop = double(zeros(R-(2*pad),C-(2*pad),k));

 for i=1:1:R-(2*pad);
     for j=1:1:C-(2*pad);
         for k=1:1:3;
             imgCrop(i,j,k) = img(i+pad,j+pad,k);
             imgWangCrop(i,j,k) = imgWang(i+pad,j+pad,k);
         end;
     end;
 end;

% [PSNR_R, PSNR_G, PSNR_B] = measurePSNR(imgWang,img,pad);
% [CPSNR] = measureCPSNR(imgWang,img,pad);
% PSNR_R;
% PSNR_G;
% PSNR_B;
% CPSNR;
%
% [observedLAB,actualLAB,diffLAB] = measureLAB(uint8(imgWang),img,pad);
% [diffdelta00] = measureDELTA00(uint8(imgWangCrop),imgCrop);

% padBorder = 4;
%
% [PSNR(1),PSNR(2),PSNR(3)] = fcn_measurePSNRv2(uint8(imgWang),img,padBorder);
% PSNR = PSNR';
% CPSNR = fcn_measureCPSNRv2(uint8(imgWang),img,padBorder);
% SSIM = ssim(img,uint8(imgWang));
% [FSIM,FSIMc] = FeatureSIM(img,uint8(imgWang));

MSE = fcn_measureMSESinglev2(uint8(imgWang(:,:,2)),uint8(img(:,:,2)),4);
[FSIM,FSIMc] = FeatureSIM(uint8(img),uint8(imgWang));
CPSNR = fcn_measureCPSNRv2(uint8(imgWang),uint8(img),4);
SSIM = ssim(img,uint8(imgWang));
```

Multi-scale Gradient Based Interpolation Algorithm (Author’s Implementation)

```
%==========================================================================
%   Name:       MGBI_v1.m
%   Author:     Kinyua Wachira
%   Date:       --/--/2015
%   Desc:       an implementation of Multi-gradient Based Interpolation
%   seen in IEEE TIP v22 n1 2013 (06253257)
```

```
%
%   Notes:      will try to make note of any assumptions here
%           1. i think at this stage diffH, diffV are correct
%           2. created DHmap and DVmap
%           3. managed to create the green plane - has a border of 4
%              also not implemented the green update section using
%              w,ws,wn,we,ww yet
%
%
%     18/9  4. implemented the green updation stage, since w not given
%     picked w=0.5 - seen improvment i.e kodim19 -> PSNR_G from 41.6174 to
%     43.2833
%
%           5. only performed one updation iteration for this algorithm
%           implementation
%
%     24/9 My implementation of this algorithm fails at white-only regions
%     -this has resulted in problems for kodim 06 08 10 15 20 23 24 that
%     all have large white regions in the image --> need to determine
%     whether this is due to bad coding or is it inherent from the
%     algorithm itself
%===========================================================================

%utility fcns
clc; clear all; close all hidden;

%load image and generate the Bayer representation
%load image
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\USC-
SIPI\sipi_im16.tiff');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Kodak\kodim24.png');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\McM\mcm18.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Condat\codim30.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\ARRI\arri_im12.tif');
img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Custom1\cusim15.jpg');


imgBayer = fcn_bayer(img);

%generate maps
[R,C] = size(imgBayer);
imgBayer = double(imgBayer);
[rGHmap, RHmap, rGVmap, RVmap] = deal(double(zeros(R,C))); %used deal to do
multiple variable initialisation
[bGHmap, BHmap, bGVmap, BVmap] = deal(double(zeros(R,C)));

%error term to prevent
err=0;
padBorder =4;

%rGHmap, RHmap, bGHmap, BHmap - all horizontal maps
%NOTE: here G_row (RED) = i is odd, j is even
%           G_row (BLUE) = i is even, j is odd
for i=1:1:R;
    for j=1+2:1:C-2;
        if (mod(i,2) == 1 && mod(j,2) == 1) %rGHmap
            rGHmap(i,j) = (0.5.*(imgBayer(i,j-1) + imgBayer(i,j+1))) ...
            + (0.25*(2*imgBayer(i,j) - imgBayer(i,j-2) - imgBayer(i,j+2)));
        end;
        if (mod(i,2) ==1 && mod(i+j,2) == 1) %RHmap
            RHmap(i,j) = (0.5.*(imgBayer(i,j-1) + imgBayer(i,j+1))) ...
            + (0.25*(2*imgBayer(i,j) - imgBayer(i,j-2) - imgBayer(i,j+2)));
        end;
        if (mod(i,2) == 0 && mod(j,2) == 0) %bGHmap
```

```
                bGHmap(i,j) = (0.5.*(imgBayer(i,j-1) + imgBayer(i,j+1))) ...
                + (0.25*(2*imgBayer(i,j) - imgBayer(i,j-2) - imgBayer(i,j+2)));
            end;
            if (mod(i,2) ==0 && mod(i+j,2) == 1) %BHmap
                BHmap(i,j) = (0.5.*(imgBayer(i,j-1) + imgBayer(i,j+1))) ...
                + (0.25*(2*imgBayer(i,j) - imgBayer(i,j-2) - imgBayer(i,j+2)));
            end;
        end;
end;

%rGVmap, RVmap, bGVmap, BVmap - all vertical maps
%NOTE: here G_col (RED) = i is even, j is odd
%            G_col (BLUE) = i is odd, j is even
for i=1+2:1:R-2;
    for j=1:1:C;
        if (mod(i,2)==1 && mod(j,2)==1) %rGVmap
            rGVmap(i,j) = (0.5*(imgBayer(i-1,j) + imgBayer(i+1,j))) ...
            + (0.25*(2*imgBayer(i,j) - imgBayer(i-2,j) - imgBayer(i+2,j)));
        end;
        if (mod(i,2)==0 && mod(i+j,2)==1) %RVmap
            RVmap(i,j) = (0.5*(imgBayer(i-1,j) + imgBayer(i+1,j))) ...
            + (0.25*(2*imgBayer(i,j) - imgBayer(i-2,j) - imgBayer(i+2,j)));
        end;
        if (mod(i,2)==0 && mod(j,2)==0) %bGVmap
            bGVmap(i,j) = (0.5*(imgBayer(i-1,j) + imgBayer(i+1,j))) ...
            + (0.25*(2*imgBayer(i,j) - imgBayer(i-2,j) - imgBayer(i+2,j)));
        end;
        if (mod(i,2)==1 && mod(i+j,2) ==1) %BVmap
            BVmap(i,j) = (0.5*(imgBayer(i-1,j) + imgBayer(i+1,j))) ...
            + (0.25*(2*imgBayer(i,j) - imgBayer(i-2,j) - imgBayer(i+2,j)));
        end;
    end;
end;

%establish the diffH and diffV maps
[diffH, diffV] = deal(double(zeros(R,C)));

for i=1:1:R;
    for j=1+2:1:C-2;
        if (mod(i,2)==1 && mod(j,2)==1)
            diffH(i,j) = rGHmap(i,j) - imgBayer(i,j);
        end;
        if (mod(i,2)==1 && mod(i+j,2)==1)
            diffH(i,j) = imgBayer(i,j) - RHmap(i,j);
        end;
        if (mod(i,2)==0 && mod(j,2)==0)
            diffH(i,j) = bGHmap(i,j) - imgBayer(i,j);
        end;
        if (mod(i,2)==0 && mod(i+j,2)==1)
            diffH(i,j) = imgBayer(i,j) - BHmap(i,j);
        end;
    end;
end;

for i=1+2:1:R-2;
    for j=1:1:C;
        if (mod(i,2)==1 && mod(j,2)==1)
            diffV(i,j) = rGVmap(i,j) - imgBayer(i,j);
        end;
        if (mod(i,2)==0 && mod(i+j,2)==1)
            diffV(i,j) = imgBayer(i,j) - RVmap(i,j);
        end;
        if (mod(i,2)==0 && mod(j,2)==0)
            diffV(i,j) = bGVmap(i,j) - imgBayer(i,j);
        end;
        if (mod(i,2)==1 && mod(i+j,2)==1)
            diffV(i,j) = imgBayer(i,j) - BVmap(i,j);
        end;
    end;
end;

%establish the absolute difference maps (DHmap,DVmap)
[DHmap, DVmap] = deal(double(zeros(R,C)));

for i=1:1:R;
    for j=1+3:1:C-3;
        DHmap(i,j) = abs(diffH(i,j-1) - diffH(i,j+1));
    end;
end;
for i=1+3:1:R-3;
    for j=1:1:C;
        DVmap(i,j) = abs(diffV(i-1,j) - diffV(i+1,j));
    end;
end;

clear i j

%=========================================================================
[imgRed,imgGrn,imgBlu] = deal(double(zeros(R,C)));
%for red in red pixels
for i=1:1:R;
    for j=1:1:C;
        if (mod(i,2)==1 && mod(j,2)==1)
            imgRed(i,j) = imgBayer(i,j);
        end;
    end;
end;
%for green in green pixels
for i=1:1:R;
    for j=1:1:C;
        if (mod(i+j,2) == 1)
            imgGrn(i,j) = imgBayer(i,j);
        end;
    end;
end;
%for blue in blue pixels
for i=1:1:R;
    for j=1:1:C;
        if (mod(i,2)==0 && mod(j,2)==0)
            imgBlu(i,j) = imgBayer(i,j);
        end;
    end;
end;
%=========================================================================

%=========================================================================
% Green Plane Interpolation
%=========================================================================
f = [0.25, 0.5, 0.25];
%note we are looking for the missing components in the green plane

%for green in red pixels
%gmap --> can be grmap or gbmap depending on place
gmap = double(zeros(R,C));
for i=1+4:1:R-4;
    for j=1+4:1:C-4;
        if ~(mod(i+j,2)==1)
            %establish the weights
            wV = 1/(sum(sum(DVmap(i-2:i+2,j-2:j+2),2)));
            wH = 1/(sum(sum(DHmap(i-2:i+2,j-2:j+2),2)));
            %construct the gmap
            gmap(i,j) = (wH.*(diffH(i,j-1:j+1)*f') + wV.*(f*diffV(i-1:i+1,j)))...
                         /(wV+wH);
        end;
    end;
```

```
end;

%green channel update
ggmap = double(zeros(R,C));
for i=1+4:1:R-4;
    for j=1+4:1:C-4;
        if ~(mod(i+j,2)==1)
            %establish weights
            w = 0.5;
            wN = 1/((sum(sum(DVmap(i-4:i,j-1:j+1),2)))+err);
            wS = 1/((sum(sum(DHmap(i:i+4,j-1:j+1),2)))+err);
            wW = 1/((sum(sum(DVmap(i-1:i+1,j-4:j),2)))+err);
            wE = 1/((sum(sum(DHmap(i-1:i+1,j:j+4),2)))+err);
            wT = wN + wS + wW + wE;
            %construct the ggmap

            ggmap(i,j) = (gmap(i,j).*(1-w)) + ...
             ( w.*(wN.*gmap(i-2,j) + wS.*gmap(i+2,j) + ...
                 wW.*gmap(i,j-2) + wE.*gmap(i,j+2))/wT );

            if (mod(i,2)==1 && mod(j,2)==1)
                %get the green contents in red pixels
                imgGrn(i,j) = imgRed(i,j) + ggmap(i,j);
            end;
            if (mod(i,2)==0 && mod(j,2)==0)
                %get the green contents in red pixels
                imgGrn(i,j) = imgBlu(i,j) + ggmap(i,j);
            end;
        end;
    end;
end;

%========================================================================
% Red and Blue Plane Interpolation
%========================================================================
prb = 1/32.*[ 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];

[Rmap,Bmap] = deal(double(zeros(R,C)));

%for the red in blue pixels, and blue in red pixels data
for i=1+4:1:R-4;
    for j=1+4:1:C-4;
        if (mod(i,2)==0  && mod(j,2)==0)
            Rmap(i,j) = imgGrn(i,j) - sum(sum(ggmap(i-3:i+3,j-3:j+3).*prb));
            imgRed(i,j) = Rmap(i,j);
        end;
        if (mod(i,2)==1  && mod(j,2)==1)
            Bmap(i,j) = imgGrn(i,j) - sum(sum(ggmap(i-3:i+3,j-3:j+3).*prb));
            imgBlu(i,j) = Bmap(i,j);
        end;
    end;
end;

%for the red or blue in green pixel locations
for i=1+4:1:R-4;
    for j=1+4:1:C-4;
        if (mod(i+j,2)==1)
            %establish the weights
            wV = 1/(sum(sum(DVmap(i-2:i+2,j-2:j+2),2)));
            wH = 1/(sum(sum(DHmap(i-2:i+2,j-2:j+2),2)));
            %populate the red and blue planes
            imgRed(i,j) = imgGrn(i,j) - ...
```

```
               (  wV.*((imgGrn(i-1,j)-imgRed(i-1,j)) + (imgGrn(i+1,j)-imgRed(i+1,j))) ...
                + wH.*((imgGrn(i,j-1)-imgRed(i,j-1)) + (imgGrn(i,j+1)-imgRed(i,j+1))) )...
               /(2.*(wV+wH));
           imgBlu(i,j) = imgGrn(i,j) - ...
               (  wV.*((imgGrn(i-1,j)-imgBlu(i-1,j)) + (imgGrn(i+1,j)-imgBlu(i+1,j))) ...
                + wH.*((imgGrn(i,j-1)-imgBlu(i,j-1)) + (imgGrn(i,j+1)-imgBlu(i,j+1))) )...
               /(2.*(wV+wH));
        end;
    end;
end;

%========================================================================
% Final Reconstruction
%========================================================================
imgMGBI = double(zeros(R,C,3));
imgMGBI(:,:,1) = imgRed;
imgMGBI(:,:,2) = imgGrn;
imgMGBI(:,:,3) = imgBlu;

%========================================================================
% Results
%========================================================================

%imtool(img);
%imtool(uint8(imgBayer));
%imtool(uint8(rGVmap));
%imtool(uint8(RVmap));
%imtool(uint8(diffV));
%imtool(uint8(DVmap));
%imtool(uint8(imgMGBI));
%
% [PSNR(1),PSNR(2),PSNR(3)] = fcn_measurePSNRv2(uint8(imgMGBI),img,padBorder);
% PSNR = PSNR';
% CPSNR = fcn_measureCPSNRv2(uint8(imgMGBI),img,padBorder);
% SSIM = ssim(img,uint8(imgMGBI));
% [FSIM,FSIMc] = FeatureSIM(img,uint8(imgMGBI));

MSE = fcn_measureMSESinglev2(uint8(imgMGBI(:,:,2)),uint8(img(:,:,2)),4);
[FSIM,FSIMc] = FeatureSIM(uint8(img),uint8(imgMGBI));
CPSNR = fcn_measureCPSNRv2(uint8(imgMGBI),uint8(img),4);
SSIM = ssim(img,uint8(imgMGBI));
```

The Average-based Colour Reconstruction Algorithm (Author's Implementation)

```
%=========================================================================
% Author: Kinyua Wachira
% Date: 25-10-2016
% Name: algorithm_AWCR.m
%
% Desc: Average-based Color Reconstruction is an algorithm proposed in
% 2007 by Honda et. al in "A novel Bayer-like WRGB color filter array for \
% CMOS image sensors".
% This is my implementation of it.
%
% Sections
% [1] Utility Functions
% [2] Load Image
% [3] Call function_Convert2WRGBCFA.m
% [4] Pre-Algorithm Setup
% [5a] Algorithm - White Pixel Centre Reconstruction
% [5b] Algorithm - Red Pixel Centre Reconstruction
% [5c] Algorithm - Green Pixel Centre Reconstruction
% [5d] Algorithm - Blue Pixel Centre Reconstruction
% [6] Display Result
%===========================================================================

% [1] Utility Functions
clc;    %clear command window
clear; %clear any prior variables in workspace
close all hidden; % close all figures

% [2] Load Image
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\USC-
SIPI\sipi_im16.tiff');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Kodak\kodim24.png');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\McM\mcm18.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Condat\codim30.tif');
%img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\ARRI\arri_im12.tif');
img = imread('C:\Users\Kinyua
Wachira\Desktop\IMAGESETS\Custom1\cusim15.jpg');

% [3] Call function_Convert2WRGBCFA.m
imgWRGB = function_Convert2WRGBCFA(img);

% [4] Pre-Algorithm Setup
imgRes = double(imgWRGB);
[R,C,N] = size(imgRes);
pBord = 2; % Pixel Border width

% [5a] Algorithm - White Pixel Centre Reconstruction
for i=(1+pBord):1:(R-pBord);
    for j=(1+pBord):1:(C-pBord);
        if (mod(i,2)==1 && mod(j,2)==1) % white pixel
            W =(imgRes(i,j,1) + imgRes(i,j,2) + imgRes(i,j,3)); % define W
            Rav = 0.5.*(imgRes(i,j-1,1) + imgRes(i,j+1,1)); % define Rav
            Gav = 0.25.*(imgRes(i-1,j-1,2) + imgRes(i-1,j+1,2) + ...
                         imgRes(i+1,j-1,2) + imgRes(i+1,j+1,2)); % define
Gav
            Bav = 0.5.*(imgRes(i-1,j,3) + imgRes(i+1,j,3)); % define Bav
            imgRes(i,j,1) = Rav./(Rav+Gav+Bav).*W; % compute Rw
            imgRes(i,j,2) = Gav./(Rav+Gav+Bav).*W; % compute Gw
            imgRes(i,j,3) = Bav./(Rav+Gav+Bav).*W; % compute Bw
        end;
    end;
end;

% [5b] Algorithm - Red Pixel Centre Construction
for i=(1+pBord):1:(R-pBord)
    for j=(1+pBord):1:(C-pBord)
        if (mod(i,2)==1 && mod(j,2)==0) % red pixel
            imgRes(i,j,2) = (imgRes(i-1,j,2) + imgRes(i+1,j,2) + ...
                            imgRes(i,j-1,2) + imgRes(i,j+1,2))./4; % compute
Gr
            imgRes(i,j,3) = (imgRes(i-1,j-1,3) + imgRes(i-1,j+1,3) + ...
                             imgRes(i+1,j-1,3) + imgRes(i+1,j+1,3) + ...
                             imgRes(i,j-1,3) + imgRes(i,j+1,3))./6; %compute
Br
        end;
    end;
end;

% [5c] Algorithm - Green Pixel Centre Construction
for i=(1+pBord):1:(R-pBord)
    for j=(1+pBord):1:(C-pBord)
        if (mod(i,2)==0 && mod(j,2)==0) % green pixel
```

```
            imgRes(i,j,1) = (imgRes(i-1,j-1,1) + imgRes(i-1,j,1) + imgRes(i-1,j+1,1) + ...
                             imgRes(i+1,j-1,1) + imgRes(i+1,j,1) + imgRes(i+1,j+1,1))./6; % compute Rg
            imgRes(i,j,3) = (imgRes(i-1,j-1,3) + imgRes(i,j-1,3) + imgRes(i+1,j-1,3) + ...
                             imgRes(i-1,j+1,3) + imgRes(i,j+1,3) + imgRes(i+1,j+1,3))./6; % compute Bg
        end;
    end;
end;

% [5d] Algorithm - Blue Pixel Centre Construction
for i=(1+pBord):1:(R-pBord)
    for j=(1+pBord):1:(C-pBord)
        if (mod(i,2)==0 && mod(j,2)==1) % blue pixel
            imgRes(i,j,1) = (imgRes(i-1,j-1,1) + imgRes(i-1,j+1,1) + ...
                             imgRes(i+1,j-1,1) + imgRes(i+1,j+1,1) + ...
                             imgRes(i-1,j,1) + imgRes(i+1,j,1))./6; % compute Rb
            imgRes(i,j,2) = (imgRes(i-1,j,2) + imgRes(i+1,j,2) + ...
                             imgRes(i,j-1,2) + imgRes(i,j+1,2))./4; % compute Gb
        end;
    end;
end;

% [6] Display Result
 imgACR = uint8(imgRes);
% clear imgRes;
% imtool(img);
% imtool(imgACR);

MSE = fcn_measureMSESinglev2(uint8(imgACR(:,:,2)),uint8(img(:,:,2)),4);
CPSNR = fcn_measureCPSNRv2(imgACR,img,4); %border length of 4 is used
SSIM = ssim(imgACR,img);
[FSIM,FSIMc] = FeatureSIM(img,imgACR);
% end of M-file
%=======================================================================
```

The Edge Detection-based Colour Reconstruction Algorithm (Author's Implementation)

```
%==========================================================================
% Author: Kinyua Wachira
% Date: 28-10-2016
% Name: algorithm_EDCR.m
%
% Desc: Edge Detection-based Color Reconstruction is
% an algorithm proposed in 2007 by Honda et. al in "High Sensitivity Color
% CMOS Image Sensor with WRGB Color Filter Array and Color Separation
% Process Using Edge Detection"
% This is my implementation of it.
%
% Sections
% [1] Utility Functions
% [2] Load Image
% [3] Call function_Convert2WRGBCFA.m
% [4] Pre-Algorithm Setup
% [5] Edge Detection Process
% [6a] Algorithm - White Pixel Centre, No Edge Detected, Reconstruction
% [6b] Algorithm - White Pixel Centre, Edge Detected, Reconstruction
% [6c] Algorithm - Green Pixel Centre Reconstruction
% [6d] Algorithm - Red Pixel Centre Reconstruction
% [6e] Algorithm - Blue Pixel Centre Reconstruction
% [7] Display Results
%============================================================================

% [1] Utility Functions
clc;    %clear command window
clear; %clear any prior variables in workspace
close all hidden; % close all figures

% [2] Load Image
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\USC-SIPI\sipi_im16.tiff');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\Kodak\kodim24.png');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\McM\mcm18.tif');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\Condat\codim30.tif');
%img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\ARRI\arri_im12.tif');
img = imread('C:\Users\Kinyua Wachira\Desktop\IMAGESETS\Custom1\cusim15.jpg');


% [3] Call function_Convert2WRGBCFA.m
imgWRGB = function_Convert2WRGBCFA(img);

% [4] Pre-Algorithm Setup
imgRes = double(imgWRGB);
[R,C,N] = size(imgRes);
pBord = 2; % Pixel Border width
thres = 60; % Threshold Value

% [5] Edge Detection Process
imgEdge = uint8(zeros(R,C)); %setup an edge map to contain edge detection information

for i=(1+pBord):1:(R-pBord)
    for j=(1+pBord):1:(C-pBord)
        if (mod(i,2)==1 && mod(j,2)==1) % white pixel
            % horizontal edge |(G11+B11+G12) - (G21+B21+G22)|
            if abs((imgRes(i-1,j-1,2)+imgRes(i-1,j,3)+imgRes(i-1,j+1,2)) - (imgRes(i+1,j-1,2)+imgRes(i+1,j,3)+imgRes(i+1,j+1,2)))>thres
                imgEdge(i,j) = 255;
            % vertical edge |(G11+R11+G21) - (G12+R12+G22)|
            elseif abs((imgRes(i-1,j-1,2)+imgRes(i,j-1,1)+imgRes(i+1,j-1,2)) - (imgRes(i-1,j+1,2)+imgRes(i,j+1,1)+imgRes(i+1,j+1,2)))>thres
                imgEdge(i,j) = 255;
            % left-handed diagonal edge |(R11+B21) - (B11+R12)|
            elseif abs((imgRes(i,j-1,1)+imgRes(i+1,j,3))-(imgRes(i-1,j,3)+imgRes(i,j+1,1)))>thres
                imgEdge(i,j) = 255;
            % right-handed diagonal edge |(B11+R11) - (R12+B21)|
            elseif abs((imgRes(i-1,j,3)+imgRes(i,j-1,1))-(imgRes(i,j+1,1)+imgRes(i+1,j,3)))>thres
                imgEdge(i,j) = 255;
            else
                imgEdge(i,j) = 0;
            end;
        end;
    end;
end;

% [6a] Algorithm - White Pixel, No Edge Detected, Reconstruction
for i=(1+pBord):1:(R-pBord)
    for j=(1+pBord):1:(C-pBord)
        if (mod(i,2)==1 && mod(j,2)==1 && imgEdge(i,j)==0) % white pixel
            W =(imgRes(i,j,1) + imgRes(i,j,2) + imgRes(i,j,3)); % define W
            Rav = 0.5.*(imgRes(i,j-1,1) + imgRes(i,j+1,1)); % define Rav
            Gav = 0.25.*(imgRes(i-1,j-1,2) + imgRes(i-1,j+1,2) + ...
                         imgRes(i+1,j-1,2) + imgRes(i+1,j+1,2)); % define Gav
            Bav = 0.5.*(imgRes(i-1,j,3) + imgRes(i+1,j,3)); % define Bav
            imgRes(i,j,1) = Rav./(Rav+Gav+Bav).*W; % compute Rw
            imgRes(i,j,2) = Gav./(Rav+Gav+Bav).*W; % compute Gw
            imgRes(i,j,3) = Bav./(Rav+Gav+Bav).*W; % compute Bw
        end;
    end;
end;

% [6b] Algorithm - White Pixel, Edge Detected, Reconstruction
for i=(1+pBord):1:(R-pBord)
    for j=(1+pBord):1:(C-pBord)
        if (mod(i,2)==1 && mod(j,2)==1 && imgEdge(i,j)==255) % white pixel
            W =(imgRes(i,j,1) + imgRes(i,j,2) + imgRes(i,j,3)); % define W
            Gav = 0.25.*(imgRes(i-1,j-1,2) + imgRes(i-1,j+1,2) + ...
                         imgRes(i+1,j-1,2) + imgRes(i+1,j+1,2)); % define Gav
            imgRes(i,j,1) = (imgRes(i,j-1,1)+imgRes(i,j+1,1))./2; % Rw = 0
            imgRes(i,j,2) = Gav./(Rav+Gav+Bav).*W; % compute Gw
            imgRes(i,j,3) = (imgRes(i-1,j,3)+imgRes(i+1,j,3))./2; % Bw = 0
        end;
    end;
end;

% [6c] Algorithm - Green Pixel Reconstruction
for i=(1+pBord):1:(R-pBord)
    for j=(1+pBord):1:(C-pBord)
        if (mod(i,2)==0 && mod(j,2)==0) % green pixel
            imgRes(i,j,1) = (imgRes(i-1,j-1,1) + imgRes(i-1,j,1) + imgRes(i-1,j+1,1) + ...
                             imgRes(i+1,j-1,1) + imgRes(i+1,j,1) + imgRes(i+1,j+1,1))./6; % compute Rg
            imgRes(i,j,3) = (imgRes(i-1,j-1,3) + imgRes(i,j-1,3) + imgRes(i+1,j-1,3) + ...
                             imgRes(i-1,j+1,3) + imgRes(i,j+1,3) + imgRes(i+1,j+1,3))./6; % compute Bg
        end;
    end;
end;

% [6d] Algorithm - Red Pixel Reconstruction
for i=(1+pBord):1:(R-pBord)
    for j=(1+pBord):1:(C-pBord)
        if (mod(i,2)==1 && mod(j,2)==0) % red pixel
            imgRes(i,j,2) = (imgRes(i-1,j,2) + imgRes(i+1,j,2) + ...
                            imgRes(i,j-1,2) + imgRes(i,j+1,2))./4; % compute Gr
            imgRes(i,j,3) = (imgRes(i-1,j-1,3) + imgRes(i-1,j+1,3) + ...
                             imgRes(i+1,j-1,3) + imgRes(i+1,j+1,3) + ...
                             imgRes(i,j-1,3) + imgRes(i,j+1,3))./6; %compute Br
        end;
    end;
end;

% [6e] Algorithm - Blue Pixel Reconstruction
for i=(1+pBord):1:(R-pBord)
    for j=(1+pBord):1:(C-pBord)
        if (mod(i,2)==0 && mod(j,2)==1) % blue pixel
            imgRes(i,j,1) = (imgRes(i-1,j-1,1) + imgRes(i-1,j+1,1) + ...
                             imgRes(i+1,j-1,1) + imgRes(i+1,j+1,1) + ...
                             imgRes(i-1,j,1) + imgRes(i+1,j,1))./6; % compute Rb
            imgRes(i,j,2) = (imgRes(i-1,j,2) + imgRes(i+1,j,2) + ...
                             imgRes(i,j-1,2) + imgRes(i,j+1,2))./4; % compute Gb
        end;
    end;
end;

% [7] Display Result
 imgEDCR = uint8(imgRes);
% clear imgRes;
% imtool(img);
% imtool(uint8(imgEdge));
% imtool(imgEDCR);

MSE = fcn_measureMSESinglev2(uint8(imgEDCR(:,:,2)),uint8(img(:,:,2)),4);
CPSNR = fcn_measureCPSNRv2(imgEDCR,img,10); %border length of 10 i used
SSIM = ssim(imgEDCR,img);
[FSIM,FSIMc] = FeatureSIM(img,imgEDCR);

% end of M-file
%======================================================================
```

A.5 Supplementary Functions

Analysis Function Used to Determine the Value of the Corrective Term ε (Author's Implementation)

```
%==========================================================================
%   Name:       fcn_epsilonAnalysis.m
%   Author:     Kinyua Wachira
%   Date:       16/10/2014
%   Desc:       a function to write out PSNR values for an image by
%               changing the value of epsilon
%
%   Notes:      modified on 4/11/2014 to cater for ssim
%               modified on 5/11/2014 to analyses effect of different
%               directional combinations on PSNR
%==========================================================================

%==========================================================================
%   Preamble
%==========================================================================

function [PSNRArray,SSIMArray,FSIMArray,GMSDArray] =
fcn_epsilonAnalysis(imgLoc,epsilonArray)

img = imread(imgLoc);
imgBayer = fcn_bayer(img);

[R,C] = size(imgBayer);
imgGRN = double(zeros(R,C));
for i=1:1:R;
    for j=1:1:C;
        if (mod(i+j,2)==1)
            imgGRN(i,j) = imgBayer(i,j);
        end;
    end;
end;

%==========================================================================
%   Algorithm and Analysis in the Green Channel Using a Generic Weighting
%   System
%==========================================================================
% Green channel interpolation in the cardinal directions

max = length(epsilonArray);
[PSNRArray, SSIMArray, FSIMArray, GMSDArray] = deal(zeros(max,1));

for count = 1:1:max;
```

```
    imgGrn = imgGRN;
    e = epsilonArray(count);

    for i=1+2:1:R-2;
        for j=1+2:1:C-2;
            if ~(mod(i+j,2)==1)
                %initial estimates
                GN = imgGrn(i-1,j) + 0.5.*(imgBayer(i,j) - imgBayer(i-2,j));
                GS = imgGrn(i+1,j) + 0.5.*(imgBayer(i,j) - imgBayer(i+2,j));
                GW = imgGrn(i,j-1) + 0.5.*(imgBayer(i,j) - imgBayer(i,j-2));
                GE = imgGrn(i,j+1) + 0.5.*(imgBayer(i,j) - imgBayer(i,j+2));

                %establish the gradients
                dN = abs(imgGrn(i-1,j) - imgGrn(i-2,j-1))...
                    +abs(imgGrn(i-1,j) - imgGrn(i-2,j+1))+e;
                dS = abs(imgGrn(i+1,j) - imgGrn(i+2,j-1))...
                    +abs(imgGrn(i+1,j) - imgGrn(i+2,j+1))+e;
                dW = abs(imgGrn(i,j-1) - imgGrn(i-1,j-2))...
                    +abs(imgGrn(i,j-1) - imgGrn(i+1,j-2))+e;
                dE = abs(imgGrn(i,j+1) - imgGrn(i-1,j+2))...
                    +abs(imgGrn(i,j+1) - imgGrn(i+1,j+2))+e;

                wN = 1./dN;
                wS = 1./dS;
                wW = 1./dW;
                wE = 1./dE;

                %using 2 directions (6 possible combinations - 4C2)
%                 imgGrn(i,j) = (wN.*GN + wS.*GS)./(wN+wS);
%                 imgGrn(i,j) = (wN.*GN + wW.*GW)./(wN+wW);
%                 imgGrn(i,j) = (wN.*GN + wE.*GE)./(wN+wE);
%                 imgGrn(i,j) = (wS.*GS + wW.*GW)./(wS+wW);
%                 imgGrn(i,j) = (wS.*GS + wE.*GE)./(wS+wE);
%                 imgGrn(i,j) = (wW.*GW + wE.*GE)./(wW+wE);

               %using 3 directions (4 possible combinations -4C3)
%                 imgGrn(i,j) = (wN.*GN + wS.*GS + wW.*GW)./(wN+wS+wW);
%                 imgGrn(i,j) = (wN.*GN + wS.*GS + wE.*GE)./(wN+wS+wE);
%                 imgGrn(i,j) = (wS.*GS + wW.*GW + wE.*GE)./(wS+wW+wE);
%                 imgGrn(i,j) = (wN.*GN + wW.*GW + wE.*GE)./(wN+wW+wE);

%                 %using 4 direction NSEW (1 combination - 4C4)
                imgGrn(i,j) = (wN.*GN+wS.*GS+wW.*GW+wE.*GE)./(wN+wS+wW+wE);
            end;
        end;
    end;

    PSNRArray(count) = fcn_measurePSNRSinglev2(uint8(imgGrn),img(:,:,2),4);
    SSIMArray(count) = ssim(img(:,:,2),uint8(imgGrn));
    FSIMArray(count) = FeatureSIM(img(:,:,2),uint8(imgGrn));
%     GMSDArray(count) = fcn_GMSD(double(img(:,:,2)),imgGrn);

end;
```

Appendix B: Detailed Simulation Result Data

B.1 Mean Square Error (MSE) Data

Table B.1 MSE evaluation of test bed demosaicking algorithms over the USC-SIPI Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
sipi_im1	5.81	5.98	5.10	5.55	6.94	6.73	8.83	4.60	5.68	
sipi_im2	5.54	4.39	3.70	3.73	5.17	4.48	37.22	6.52	4.59	
sipi_im3	3.27	2.10	2.28	1.82	3.52	2.39	4.43	13.10	3.15	
sipi_im4	4.55	3.33	2.96	2.70	3.73	3.70	7.34	15.98	3.91	
sipi_im5	6.33	4.95	4.32	4.73	7.55	6.02	10.54	5.45	5.85	
sipi_im6	16.84	15.60	11.12	13.05	14.82	12.74	24.78	10.17	13.46	
sipi_im7	2.43	2.01	2.31	2.46	6.38	5.17	4.20	15.32	2.61	
sipi_im8	3.72	2.69	3.46	3.30	8.32	6.81	6.16	22.65	4.17	
sipi_im9	4.03	4.10	2.88	3.90	4.36	4.19	7.11	4.93	3.19	
sipi_im10	8.75	7.79	8.34	8.06	9.97	9.86	12.42	16.10	8.92	
sipi_im11	29.55	28.07	25.34	26.17	26.47	24.98	41.47	26.71	27.59	
sipi_im12	7.29	7.22	5.34	6.32	7.58	6.72	11.31	9.00	6.37	
sipi_im13	5.54	4.56	2.76	3.74	4.39	3.88	8.70	8.09	4.55	
sipi_im14	18.31	19.47	17.26	18.83	17.80	17.38	24.90	40.05	18.46	
sipi_im15	9.69	9.37	9.23	9.05	10.32	10.43	14.89	25.85	9.76	
sipi_im16	9.49	8.27	7.21	7.75	8.89	8.61	13.17	44.45	11.02	
average	6.99	6.07	5.39	5.72	7.76	6.96	11.72	13.21	6.58	4

Table B.2 MSE evaluation of test bed demosaicking algorithms over the Kodak Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
kodim01	20.00	15.92	8.54	11.06	7.00	3.63	26.81	24.26	15.03	
kodim02	5.65	6.06	2.80	4.05	3.08	2.70	8.18	4.82	4.82	
kodim03	4.57	3.62	1.70	2.68	1.81	1.58	5.98	8.10	2.82	
kodim04	5.17	5.30	1.72	3.61	1.72	1.70	8.26	7.31	3.33	
kodim05	16.17	13.74	6.30	9.13	5.20	3.55	21.89	27.66	10.24	
kodim06	14.11	11.02	6.19	7.13	4.74	2.55	18.46	14.52	9.62	
kodim07	4.87	3.72	2.00	2.42	1.83	1.74	6.58	5.46	4.71	
kodim08	22.10	13.07	12.31	10.47	10.27	5.44	31.63	57.76	17.29	
kodim09	5.16	3.51	1.91	2.25	1.73	1.26	7.69	5.61	3.77	
kodim10	4.73	3.44	1.62	2.39	1.48	1.34	6.95	6.85	3.14	
kodim11	9.94	8.62	4.61	5.74	4.11	2.66	13.32	18.63	6.71	
kodim12	5.08	3.93	2.05	2.51	1.96	1.62	6.72	9.12	3.17	
kodim13	24.09	25.73	12.53	19.65	9.57	6.56	31.14	32.35	17.00	
kodim14	11.54	11.31	4.94	7.30	4.37	3.48	15.18	7.43	7.53	
kodim15	5.98	5.86	3.05	4.59	3.38	3.98	8.30	18.75	4.34	
kodim16	8.29	6.98	3.18	4.15	2.80	1.41	11.00	5.98	5.08	
kodim17	5.10	4.65	2.00	3.03	1.46	1.27	7.25	9.13	3.38	
kodim18	11.27	11.63	4.75	7.98	4.13	3.32	15.42	12.23	7.56	
kodim19	11.15	7.23	4.97	4.95	4.04	1.77	15.66	10.31	8.35	
kodim20	5.95	5.21	2.66	3.64	2.52	15.25	7.82	6.78	3.78	
kodim21	11.16	10.34	5.17	7.24	4.49	2.74	14.89	8.59	7.87	
kodim22	8.83	8.96	3.98	6.40	0.98	3.13	11.71	9.11	6.44	
kodim23	3.15	2.76	1.39	1.90	1.89	1.89	4.44	1.93	2.37	
kodim24	12.24	10.61	6.00	8.23	5.86	8.22	16.74	14.17	8.89	
average	8.42	7.19	3.62	4.95	3.13	2.74	11.57	10.40	5.91	5

Table B.3 MSE evaluation of test bed demosaicking algorithms over the McMaster-IMAX Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
mcm01	16.42	14.49	15.16	13.48	17.58	17.16	24.40	25.81	17.26	
mcm02	7.56	6.34	5.33	5.33	7.00	6.33	11.38	12.88	6.56	
mcm03	15.53	12.53	9.98	11.19	10.47	8.77	21.02	63.36	13.50	
mcm04	8.83	4.94	7.95	6.05	7.59	6.96	11.93	31.13	8.40	
mcm05	6.97	5.43	6.26	5.75	8.28	7.75	11.27	14.12	7.54	
mcm06	3.44	3.16	3.78	3.79	7.63	7.22	6.49	5.24	4.86	
mcm07	11.50	12.09	4.80	8.66	3.96	2.83	14.68	7.49	7.25	
mcm08	6.22	4.96	3.25	3.74	2.94	2.41	8.67	23.56	4.49	
mcm09	5.34	4.04	4.08	3.77	5.77	5.57	7.95	15.97	5.14	
mcm10	3.95	3.65	2.78	2.94	4.89	4.63	7.14	12.93	3.37	
mcm11	4.01	3.88	2.74	3.26	4.82	4.84	5.87	7.61	3.33	
mcm12	5.63	3.92	3.00	2.76	5.02	4.30	8.76	9.89	3.75	
mcm13	2.08	1.17	1.79	1.56	3.49	2.93	3.46	8.40	2.79	
mcm14	3.02	2.71	2.37	2.47	3.52	3.28	4.56	9.64	2.82	
mcm15	2.88	2.85	2.39	2.71	3.75	3.71	4.41	7.24	2.77	
mcm16	14.54	12.81	9.26	10.52	13.72	13.10	19.19	8.71	11.09	
mcm17	7.24	6.99	8.33	8.05	13.04	13.00	11.83	5.49	8.81	
mcm18	11.92	9.39	7.67	8.25	8.97	8.25	16.44	20.26	9.51	
average	6.39	5.27	4.71	4.88	6.44	5.87	9.61	12.72	5.86	4

Table B.4 MSE evaluation of test bed demosaicking algorithms over the Condat Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
codim01	13.18	10.43	7.99	8.05	8.48	7.12	18.67	16.66	10.27	
codim02	7.60	6.31	5.22	5.05	6.46	5.73	11.26	14.19	5.95	
codim03	6.83	5.48	4.11	4.43	5.46	4.92	9.62	8.19	6.22	
codim04	9.46	6.51	6.29	5.32	6.25	5.04	13.30	24.87	7.18	
codim05	7.26	6.71	4.89	5.70	6.41	5.73	10.51	4.37	6.26	
codim06	5.87	4.21	4.44	3.86	5.54	5.06	8.90	4.30	5.64	
codim07	11.44	9.12	6.80	7.51	7.60	10.32	15.15	19.54	8.47	
codim08	10.47	7.91	7.25	6.90	7.74	6.10	15.16	3.13	8.22	
codim09	22.09	20.86	16.53	17.63	16.74	16.71	31.55	7.17	17.84	
codim10	6.59	3.15	3.49	2.62	4.10	3.05	9.82	27.35	8.33	
codim11	19.79	20.72	18.60	21.05	23.13	22.29	30.43	9.17	21.39	
codim12	17.42	11.57	11.22	9.76	9.60	7.17	24.74	5.44	14.17	
codim13	16.16	13.10	11.03	11.16	10.81	9.48	22.83	27.49	13.35	
codim14	3.73	2.97	3.42	3.18	5.37	4.76	6.38	12.59	4.75	
codim15	11.92	9.10	8.80	7.76	9.00	7.29	16.75	12.57	9.30	
codim16	4.29	3.44	3.67	3.38	5.16	4.81	6.61	12.50	4.75	
codim17	7.79	6.14	7.23	6.03	9.27	33.60	13.10	4.25	8.16	
codim18	7.12	5.00	5.23	4.37	6.40	5.13	11.50	4.25	7.13	
codim19	6.45	5.14	3.14	3.82	3.75	2.48	8.97	4.20	4.08	
codim20	6.88	4.93	5.59	4.32	7.08	6.75	10.31	3.19	6.37	
codim21	3.82	3.13	2.40	2.29	2.61	1.97	5.66	4.83	3.22	
codim22	11.65	11.77	8.65	10.46	9.14	10.20	16.47	6.78	10.04	
codim23	2.05	1.31	0.67	0.81	0.86	1.04	3.24	1.32	1.30	
codim24	17.00	14.29	9.09	11.48	9.08	7.49	23.00	29.28	13.65	
codim25	2.24	1.40	2.23	1.82	4.40	4.41	4.48	1.84	2.45	
codim26	11.20	8.75	7.32	6.74	7.45	4.85	15.74	5.18	7.23	
codim27	14.61	13.23	8.46	9.56	8.39	6.99	20.57	26.55	11.59	
codim28	2.36	1.88	2.22	2.08	3.88	3.68	4.07	2.00	2.28	
codim29	3.33	3.25	2.80	2.94	4.44	4.50	5.26	2.39	2.95	
codim30	12.10	8.00	8.37	6.70	8.64	6.97	18.02	12.38	10.32	
average	7.84	6.12	5.42	5.27	6.45	5.96	11.72	7.43	6.82	6

Table B.5 MSE evaluation of test bed demosaicking algorithms over the ARRI Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
arri_im1	3.46	3.38	2.55	2.53	4.38	4.40	2.50	31.62	4.16	
arri_im2	2.43	2.23	1.35	1.25	2.14	2.18	2.27	9.00	3.44	
arri_im3	0.65	0.50	0.48	0.52	0.67	2.08	0.81	4.99	1.26	
arri_im4	1.48	1.27	1.64	1.45	2.09	2.11	1.90	1.52	1.95	
arri_im5	0.82	0.76	0.56	0.58	0.91	0.89	0.90	0.82	0.82	
arri_im6	1.67	1.44	1.27	1.19	2.01	2.00	1.61	8.95	2.27	
arri_im7	5.45	4.17	3.42	2.72	4.38	4.28	6.10	17.08	6.75	
arri_im8	1.34	1.10	1.47	1.30	2.33	2.40	1.69	1.11	1.85	
arri_im9	5.47	0.69	2.26	0.40	2.59	3.87	9.36	23.60	5.80	
arri_im10	1.22	1.08	1.48	1.43	2.39	2.34	1.88	1.59	1.82	
arri_im11	2.51	1.76	2.11	1.39	3.18	3.04	3.03	1.34	4.76	
arri_im12	6.95	5.11	5.17	4.83	7.66	7.71	10.55	2.84	7.13	
average	2.14	1.53	1.63	1.29	2.40	2.72	2.54	4.32	2.85	8

Table B.6 MSE evaluation of test bed demosaicking algorithms over the Custom Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
cusim01	5.19	4.43	1.92	2.69	1.50	0.91	7.37	7.09	3.75	
cusim02	3.33	3.24	1.22	2.03	1.23	0.71	4.62	3.18	2.05	
cusim03	5.63	4.27	2.30	2.78	1.41	1.21	8.70	10.53	4.18	
cusim04	7.80	8.21	2.12	4.62	1.36	0.95	9.82	5.95	4.30	
cusim05	5.16	2.91	1.82	1.79	1.57	0.58	7.42	1.87	4.32	
cusim06	6.32	4.72	2.16	2.81	1.85	1.42	8.75	9.79	4.96	
cusim07	5.53	4.20	1.69	2.31	1.37	0.63	7.84	5.18	4.73	
cusim08	5.01	4.41	1.61	2.56	1.35	5.83	6.96	11.53	3.46	
cusim09	8.17	7.50	2.54	4.32	1.88	21.08	10.96	16.68	5.50	
cusim10	7.57	7.63	2.64	4.46	1.66	4.51	10.10	4.50	4.78	
cusim11	3.48	4.09	1.34	2.75	1.12	0.79	4.68	7.57	1.84	
cusim12	2.59	3.08	0.77	1.71	0.85	0.58	3.40	1.98	1.55	
cusim13	5.80	4.67	1.93	2.50	1.36	0.69	8.07	6.11	4.04	
cusim14	0.97	1.03	0.27	0.61	0.42	0.63	1.16	1.37	0.63	
cusim15	4.89	4.51	1.66	2.64	1.15	0.93	6.86	8.19	2.95	
average	4.66	4.17	1.55	2.47	1.27	1.28	6.38	5.48	3.13	5

B.2 Colour Peak Signal-to-Noise Ratio (CPSNR) Data

Table B.7 CPSNR evaluation of test bed demosaicking algorithms over the USC-SIPI Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
sipi_im1	33.49	37.66	37.56	37.58	37.35	37.10	37.22	37.66	39.79	
sipi_im2	34.57	39.47	39.16	39.54	38.92	38.62	37.61	37.80	39.23	
sipi_im3	36.80	42.95	42.34	43.79	42.37	42.24	41.72	39.21	43.44	
sipi_im4	36.01	40.70	40.10	41.16	39.87	39.46	38.83	37.41	42.80	
sipi_im5	33.65	38.06	37.90	38.33	37.77	37.49	36.48	37.00	40.50	
sipi_im6	29.55	34.79	35.15	35.31	35.11	35.16	34.05	34.72	35.51	
sipi_im7	37.33	42.23	41.74	41.87	39.38	39.19	41.91	38.83	42.35	
sipi_im8	35.81	40.92	40.26	40.69	38.21	38.05	39.94	37.01	41.05	
sipi_im9	37.38	41.05	41.42	41.11	40.45	39.96	40.83	41.46	42.26	
sipi_im10	33.19	37.43	37.68	37.42	36.92	36.61	38.18	37.57	41.86	
sipi_im11	24.72	32.13	32.20	32.27	32.27	32.39	31.78	32.24	35.27	
sipi_im12	33.91	37.80	37.88	38.01	37.57	37.44	37.31	37.51	38.05	
sipi_im13	35.57	39.60	40.44	40.39	39.79	39.73	38.62	38.46	40.12	
sipi_im14	28.95	34.16	34.32	34.30	34.24	34.25	34.31	33.44	34.63	
sipi_im15	31.79	35.96	36.00	36.06	35.94	35.82	36.22	35.28	38.62	
sipi_im16	32.47	36.81	37.24	37.04	36.68	36.72	36.22	33.78	38.66	
average	33.27	38.12	38.11	38.31	37.60	37.44	37.48	36.76	39.54	1

Table B.8 CPSNR evaluation of test bed demosaicking algorithms over the Kodak Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
kodim01	29.15	35.15	36.08	36.67	38.21	40.73	33.25	33.19	39.21	
kodim02	36.06	40.57	40.56	41.83	42.06	42.10	37.95	38.41	42.06	
kodim03	36.95	41.60	42.19	42.71	44.38	44.49	39.54	38.98	45.25	
kodim04	36.41	40.55	42.85	41.54	43.37	43.08	38.17	38.13	42.18	
kodim05	29.58	36.33	37.61	37.37	39.70	40.85	34.39	33.79	37.86	
kodim06	30.55	36.76	37.45	38.35	39.56	41.13	34.68	34.83	38.93	
kodim07	36.39	41.93	41.87	43.24	44.14	43.98	39.32	39.13	43.83	
kodim08	26.69	35.71	34.83	36.92	36.53	38.66	32.67	31.64	37.29	
kodim09	35.31	41.64	42.30	43.48	43.13	45.21	38.51	38.59	42.28	
kodim10	35.21	41.72	41.94	43.23	42.77	44.47	38.84	38.52	43.84	
kodim11	32.00	38.24	38.93	39.68	40.59	42.36	36.07	35.49	42.26	
kodim12	36.13	41.04	42.02	42.95	43.58	44.65	38.94	38.45	43.95	
kodim13	26.47	33.68	34.91	34.56	37.08	38.38	32.76	32.55	37.09	
kodim14	32.05	37.55	38.97	38.74	40.14	40.82	35.53	35.99	41.70	
kodim15	35.06	39.99	41.23	40.82	41.78	40.48	38.78	37.17	41.19	
kodim16	33.99	39.05	39.97	40.98	41.93	44.51	36.68	37.09	40.21	
kodim17	34.81	41.13	42.56	42.78	45.08	45.37	38.82	38.23	43.75	
kodim18	30.68	37.09	38.97	38.12	40.51	41.18	35.63	35.59	39.79	
kodim19	31.14	38.69	38.66	40.24	40.52	43.38	35.46	35.74	40.79	
kodim20	34.29	40.11	41.09	41.15	41.74	34.20	38.59	38.59	42.56	
kodim21	31.33	37.39	38.42	38.56	40.34	41.65	35.82	36.21	41.92	
kodim22	33.21	38.03	39.56	39.05	40.38	41.14	36.77	36.90	41.33	
kodim23	38.24	43.01	44.23	44.05	44.11	43.17	41.11	41.68	45.65	
kodim24	29.33	36.94	37.95	37.83	39.25	37.04	35.46	35.44	39.93	
average	32.80	38.84	39.72	40.12	41.23	41.70	36.75	36.61	41.39	2

Table B.9 CPSNR evaluation of test bed demosaicking algorithms over the McMaster-IMAX Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
mcm01	27.73	34.76	34.43	34.73	33.97	33.84	34.03	33.84	34.19	
mcm02	32.73	37.78	37.74	37.90	37.31	37.28	36.68	36.34	37.49	
mcm03	29.17	35.82	36.28	36.24	36.74	37.10	34.37	32.04	36.54	
mcm04	31.94	38.89	38.13	38.71	38.36	38.64	35.96	34.12	38.97	
mcm05	32.72	38.07	37.76	37.99	37.09	37.01	37.08	36.68	37.55	
mcm06	36.04	40.00	39.36	39.37	37.50	37.50	39.39	39.35	39.02	
mcm07	32.69	36.81	38.66	37.89	40.71	41.61	35.72	36.08	38.84	
mcm08	33.83	39.64	40.51	40.32	41.45	41.72	38.32	36.31	40.62	
mcm09	34.80	39.73	39.56	39.61	38.61	38.49	38.14	36.93	39.47	
mcm10	36.55	40.49	40.76	40.54	39.33	39.22	38.77	37.77	40.36	
mcm11	37.42	41.29	41.81	41.37	40.57	40.24	40.26	39.69	41.65	
mcm12	34.56	40.61	40.54	40.87	40.10	39.89	38.58	38.32	41.21	
mcm13	38.74	43.08	42.26	42.96	41.43	40.99	41.29	39.82	42.10	
mcm14	37.50	41.44	41.13	41.42	40.57	40.16	40.32	39.06	41.07	
mcm15	37.71	41.63	41.37	41.46	40.52	40.13	40.79	39.91	41.40	
mcm16	31.37	36.69	37.49	36.92	36.33	36.41	36.07	36.65	37.03	
mcm17	31.78	36.93	36.75	36.27	35.14	35.10	36.74	37.28	36.73	
mcm18	31.83	37.49	37.46	37.83	37.22	37.56	36.03	35.59	37.79	
average	33.71	38.89	38.94	38.96	38.44	38.43	37.64	36.92	38.94	2

Table B.10 CPSNR evaluation of test bed demosaicking algorithms over the Condat Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
codim01	30.13	36.92	36.83	37.43	37.27	37.17	34.82	34.75	37.01	
codim02	32.84	38.53	38.32	38.93	38.06	37.92	36.74	36.37	38.72	
codim03	33.35	39.22	38.90	39.63	38.94	38.77	37.16	37.01	39.07	
codim04	31.53	38.30	38.09	38.95	38.54	38.89	36.37	35.15	38.70	
codim05	33.37	38.37	39.10	38.74	38.47	38.42	37.59	38.20	40.75	
codim06	34.31	39.97	39.08	39.99	38.71	38.81	38.34	38.78	39.75	
codim07	32.56	37.09	37.74	37.83	37.49	35.63	36.08	35.51	37.60	
codim08	30.27	37.61	37.39	38.15	37.65	38.14	35.73	36.65	37.79	
codim09	26.19	34.05	34.30	34.26	34.50	34.22	33.16	34.14	38.48	
codim10	34.57	40.78	39.73	40.30	40.22	40.79	37.36	35.49	39.05	
codim11	26.90	33.52	33.65	33.26	32.88	32.95	32.88	33.61	33.48	
codim12	28.18	35.93	35.79	36.86	36.80	37.73	33.84	34.74	36.24	
codim13	28.35	35.92	35.91	36.43	36.35	36.54	34.38	33.85	38.10	
codim14	36.25	40.35	40.19	40.30	39.06	39.07	38.97	37.82	40.03	
codim15	29.70	37.19	36.91	37.64	37.23	37.70	35.54	35.63	37.67	
codim16	35.57	40.34	39.94	40.22	38.97	38.93	38.57	37.42	39.80	
codim17	31.56	38.11	37.19	37.81	35.54	31.19	36.22	37.06	36.96	
codim18	32.69	38.57	38.22	38.87	37.93	38.35	36.66	37.33	39.04	
codim19	35.29	40.29	40.71	40.97	41.13	42.18	38.01	38.59	41.24	
codim20	32.91	38.09	37.99	37.80	37.28	37.38	37.02	37.70	38.64	
codim21	35.55	42.07	42.40	43.01	42.60	43.20	39.66	39.62	42.56	
codim22	30.40	36.06	36.34	36.19	36.43	35.64	35.17	35.51	36.91	
codim23	41.11	45.90	44.75	47.06	47.06	45.67	42.25	42.53	45.95	
codim24	28.52	35.54	36.38	36.31	36.77	37.14	34.10	33.51	38.13	
codim25	37.42	41.59	41.08	36.34	39.61	39.31	40.92	41.81	41.67	
codim26	30.50	37.57	37.27	38.22	37.88	39.23	35.54	36.38	38.32	
codim27	29.85	36.16	36.57	36.75	36.97	37.27	34.58	33.92	36.41	
codim28	38.15	42.71	41.80	39.06	40.33	40.16	41.43	42.04	42.69	
codim29	37.05	41.59	41.50	40.32	40.08	39.94	40.54	41.08	41.39	
codim30	30.55	37.31	36.77	37.89	36.92	37.49	35.09	35.35	36.90	
average	32.34	38.43	38.29	38.44	38.18	38.09	36.75	36.84	38.90	1

Table B.11 CPSNR evaluation of test bed demosaicking algorithms over the ARRI Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
arri_im1	37.02	41.61	41.28	40.67	39.52	39.63	40.37	36.00	39.90	
arri_im2	38.68	42.84	42.91	41.76	40.62	40.88	41.16	39.07	42.02	
arri_im3	43.53	47.34	47.34	44.47	46.24	42.28	46.78	43.61	45.74	
arri_im4	41.35	45.52	44.98	44.91	43.66	42.95	43.94	44.13	43.82	
arri_im5	44.58	48.54	48.27	45.66	46.49	46.46	48.73	48.59	46.90	
arri_im6	40.41	45.10	44.33	44.17	42.93	42.91	44.75	41.28	43.04	
arri_im7	33.02	39.62	39.24	39.36	38.60	38.60	37.60	35.81	39.20	
arri_im8	39.94	44.84	44.05	43.75	42.18	41.91	44.73	45.14	42.79	
arri_im9	34.70	43.64	41.03	41.52	40.44	34.37	38.04	35.87	41.77	
arri_im10	39.87	44.31	43.66	43.09	40.71	41.68	45.12	45.31	46.30	
arri_im11	38.04	43.32	42.14	42.62	40.81	40.69	41.44	41.35	41.19	
arri_im12	32.31	38.59	38.64	38.63	37.62	37.64	38.33	39.47	39.47	
average	38.44	43.68	43.06	42.50	41.57	40.73	42.44	41.11	42.61	3

Table B.12 CPSNR evaluation of test bed demosaicking algorithms over the Custom Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
cusim01	36.50	40.98	42.56	42.79	42.74	45.48	38.43	38.33	43.46	
cusim02	38.80	42.48	43.76	44.24	45.62	46.87	40.33	40.52	45.85	
cusim03	37.19	41.76	43.38	43.37	44.12	44.82	37.99	37.36	42.41	
cusim04	35.62	38.85	42.28	40.77	45.16	46.28	37.35	37.54	41.57	
cusim05	35.86	42.68	42.66	44.91	45.08	48.28	38.54	39.31	42.49	
cusim06	35.46	40.87	42.14	42.81	44.17	44.58	37.80	37.40	41.47	
cusim07	36.47	41.24	42.97	43.78	44.46	47.13	38.07	38.14	44.21	
cusim08	36.99	41.36	43.55	43.39	44.63	38.60	38.69	37.76	43.08	
cusim09	34.93	39.22	41.72	41.23	39.83	33.47	36.85	35.99	41.13	
cusim10	34.90	39.15	41.69	40.89	44.23	39.28	37.32	37.62	41.30	
cusim11	39.05	41.44	44.57	43.02	46.65	47.70	40.37	39.64	44.84	
cusim12	40.45	43.08	46.76	44.97	47.53	48.16	41.39	41.65	46.29	
cusim13	36.15	41.13	42.76	43.37	45.69	47.86	38.16	38.12	42.43	
cusim14	44.61	47.48	46.87	49.20	49.96	48.04	46.00	45.62	50.97	
cusim15	37.46	41.17	42.52	42.73	45.80	43.48	39.16	38.53	42.78	
average	37.29	41.48	43.32	43.39	44.99	44.45	39.04	38.84	43.55	3

B.3 Structure Similarity Index (SSIM) Data

Note: the SSIM values for image *sipi_im1* to *sipi_im8* of USC-SIPI were indeterminate in the ACR & EDCR algorithms due to the implementation of the SSIM algorithm. To ensure uniform analysis over all image sets, the geometric average was established using remaining images sipi_im9 to sipi_im16.

Table B.13 SSIM evaluation of test bed demosaicking algorithms over the USC-SIPI Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
sipi_im1	0.982	0.977	0.943	0.962	0.941	0.935	-	-	0.937	
sipi_im2	0.984	0.981	0.952	0.968	0.950	0.945	-	-	0.945	
sipi_im3	0.966	0.953	0.879	0.910	0.862	0.871	-	-	0.866	
sipi_im4	0.980	0.970	0.937	0.956	0.933	0.920	-	-	0.932	
sipi_im5	0.980	0.967	0.913	0.938	0.892	0.890	-	-	0.892	
sipi_im6	0.970	0.965	0.934	0.947	0.922	0.920	-	-	0.922	
sipi_im7	0.977	0.956	0.884	0.907	0.860	0.874	-	-	0.878	
sipi_im8	0.981	0.969	0.908	0.931	0.886	0.898	-	-	0.902	
sipi_im9	0.975	0.968	0.973	0.968	0.963	0.954	0.969	0.953	0.987	
sipi_im10	0.953	0.940	0.942	0.937	0.929	0.911	0.953	0.951	0.976	
sipi_im11	0.923	0.908	0.911	0.905	0.908	0.912	0.885	0.856	0.973	
sipi_im12	0.968	0.959	0.952	0.957	0.952	0.946	0.954	0.949	0.955	
sipi_im13	0.981	0.973	0.970	0.972	0.966	0.957	0.969	0.966	0.966	
sipi_im14	0.934	0.926	0.933	0.930	0.934	0.930	0.937	0.930	0.932	
sipi_im15	0.939	0.933	0.934	0.932	0.935	0.927	0.940	0.936	0.936	
sipi_im16	0.981	0.976	0.977	0.974	0.970	0.959	0.972	0.957	0.965	
average	0.956	0.948	0.949	0.947	0.944	0.937	0.947	0.937	0.961	1

Table B.14 SSIM evaluation of test bed demosaicking algorithms over the Kodak Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
kodim01	0.961	0.961	0.973	0.974	0.983	0.986	0.917	0.905	0.979	
kodim02	0.976	0.972	0.975	0.978	0.975	0.971	0.943	0.921	0.978	
kodim03	0.983	0.979	0.982	0.980	0.984	0.983	0.968	0.964	0.984	
kodim04	0.990	0.989	0.989	0.984	0.987	0.979	0.979	0.976	0.982	
kodim05	0.981	0.978	0.982	0.981	0.985	0.984	0.954	0.942	0.992	
kodim06	0.964	0.960	0.971	0.974	0.980	0.956	0.926	0.913	0.975	
kodim07	0.992	0.988	0.987	0.988	0.986	0.983	0.980	0.967	0.984	
kodim08	0.967	0.969	0.972	0.978	0.980	0.980	0.923	0.907	0.977	
kodim09	0.991	0.991	0.990	0.986	0.989	0.984	0.983	0.978	0.987	
kodim10	0.991	0.993	0.992	0.988	0.990	0.981	0.984	0.978	0.988	
kodim11	0.973	0.969	0.975	0.978	0.980	0.981	0.941	0.934	0.977	
kodim12	0.978	0.977	0.982	0.984	0.985	0.985	0.958	0.953	0.984	
kodim13	0.950	0.939	0.964	0.956	0.977	0.975	0.899	0.883	0.969	
kodim14	0.977	0.972	0.978	0.978	0.981	0.978	0.948	0.915	0.976	
kodim15	0.982	0.979	0.985	0.983	0.986	0.970	0.968	0.957	0.985	
kodim16	0.969	0.966	0.971	0.977	0.978	0.979	0.936	0.908	0.975	
kodim17	0.994	0.991	0.984	0.987	0.984	0.978	0.980	0.976	0.978	
kodim18	0.987	0.983	0.984	0.981	0.984	0.981	0.972	0.966	0.981	
kodim19	0.979	0.980	0.969	0.974	0.971	0.971	0.955	0.939	0.968	
kodim20	0.980	0.971	0.972	0.972	0.954	0.851	0.959	0.952	0.971	
kodim21	0.979	0.972	0.976	0.975	0.979	0.977	0.952	0.941	0.975	
kodim22	0.976	0.970	0.976	0.974	0.977	0.975	0.949	0.935	0.974	
kodim23	0.991	0.987	0.985	0.985	0.984	0.973	0.980	0.965	0.983	
kodim24	0.976	0.972	0.978	0.977	0.982	0.973	0.948	0.937	0.979	
average	0.979	0.975	0.979	0.979	0.981	0.972	0.954	0.942	0.979	2

Table B.15 SSIM evaluation of test bed demosaicking algorithms over the McMaster-IMAX Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
mcm01	0.959	0.952	0.948	0.947	0.927	0.913	0.949	0.936	0.954	
mcm02	0.978	0.974	0.971	0.973	0.967	0.964	0.968	0.960	0.965	
mcm03	0.980	0.974	0.974	0.973	0.971	0.966	0.958	0.934	0.971	
mcm04	0.991	0.975	0.970	0.972	0.968	0.960	0.963	0.957	0.965	
mcm05	0.975	0.968	0.959	0.963	0.949	0.943	0.965	0.955	0.960	
mcm06	0.982	0.976	0.972	0.969	0.949	0.947	0.978	0.974	0.966	
mcm07	0.977	0.970	0.976	0.975	0.981	0.977	0.954	0.921	0.976	
mcm08	0.988	0.984	0.982	0.984	0.980	0.976	0.976	0.966	0.978	
mcm09	0.985	0.981	0.974	0.977	0.968	0.963	0.977	0.971	0.970	
mcm10	0.986	0.985	0.983	0.980	0.976	0.972	0.980	0.976	0.989	
mcm11	0.981	0.977	0.977	0.974	0.966	0.962	0.981	0.972	0.970	
mcm12	0.980	0.970	0.966	0.967	0.961	0.956	0.961	0.958	0.962	
mcm13	0.982	0.972	0.963	0.966	0.958	0.950	0.967	0.965	0.978	
mcm14	0.985	0.981	0.976	0.979	0.972	0.967	0.978	0.975	0.982	
mcm15	0.983	0.978	0.971	0.971	0.964	0.959	0.974	0.971	0.976	
mcm16	0.959	0.953	0.957	0.942	0.927	0.926	0.972	0.950	0.974	
mcm17	0.975	0.972	0.972	0.962	0.946	0.943	0.977	0.951	0.980	
mcm18	0.975	0.969	0.972	0.968	0.965	0.964	0.965	0.951	0.976	
average	0.979	0.973	0.970	0.969	0.961	0.956	0.969	0.958	0.972	3

Table B.16 SSIM evaluation of test bed demosaicking algorithms over the Condat Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
codim01	0.976	0.970	0.972	0.973	0.974	0.953	0.950	0.939	0.975	
codim02	0.986	0.977	0.970	0.972	0.967	0.949	0.970	0.967	0.968	
codim03	0.986	0.982	0.979	0.981	0.974	0.959	0.970	0.960	0.973	
codim04	0.980	0.977	0.975	0.977	0.973	0.967	0.961	0.953	0.974	
codim05	0.980	0.975	0.975	0.971	0.970	0.963	0.966	0.928	0.971	
codim06	0.972	0.968	0.960	0.963	0.948	0.948	0.968	0.933	0.965	
codim07	0.983	0.972	0.963	0.962	0.953	0.905	0.963	0.956	0.969	
codim08	0.979	0.975	0.976	0.977	0.975	0.974	0.958	0.885	0.975	
codim09	0.965	0.960	0.972	0.965	0.968	0.947	0.942	0.897	0.970	
codim10	0.992	0.988	0.980	0.983	0.978	0.971	0.978	0.970	0.974	
codim11	0.990	0.987	0.987	0.982	0.976	0.974	0.984	0.944	0.979	
codim12	0.981	0.977	0.973	0.975	0.973	0.973	0.963	0.895	0.974	
codim13	0.984	0.982	0.982	0.981	0.980	0.965	0.971	0.960	0.980	
codim14	0.994	0.989	0.983	0.984	0.979	0.975	0.987	0.982	0.989	
codim15	0.988	0.980	0.971	0.971	0.969	0.966	0.971	0.962	0.970	
codim16	0.991	0.987	0.982	0.984	0.978	0.974	0.981	0.973	0.979	
codim17	0.990	0.987	0.981	0.982	0.904	0.771	0.980	0.958	0.970	
codim18	0.974	0.971	0.967	0.972	0.964	0.963	0.949	0.888	0.972	
codim19	0.988	0.979	0.972	0.973	0.971	0.970	0.971	0.938	0.972	
codim20	0.976	0.972	0.970	0.967	0.961	0.959	0.962	0.903	0.965	
codim21	0.994	0.990	0.986	0.987	0.985	0.982	0.984	0.976	0.983	
codim22	0.974	0.970	0.973	0.970	0.973	0.936	0.959	0.937	0.984	
codim23	0.997	0.993	0.989	0.990	0.989	0.985	0.991	0.955	0.987	
codim24	0.977	0.972	0.974	0.971	0.971	0.969	0.952	0.943	0.992	
codim25	0.990	0.988	0.985	0.985	0.976	0.972	0.989	0.982	0.980	
codim26	0.970	0.961	0.967	0.962	0.967	0.968	0.951	0.892	0.979	
codim27	0.987	0.984	0.985	0.986	0.985	0.979	0.970	0.959	0.982	
codim28	0.982	0.976	0.971	0.971	0.958	0.955	0.977	0.936	0.980	
codim29	0.993	0.991	0.990	0.988	0.982	0.980	0.992	0.956	0.984	
codim30	0.973	0.969	0.967	0.970	0.962	0.955	0.948	0.928	0.961	
average	0.983	0.978	0.976	0.976	0.969	0.956	0.969	0.941	0.976	3

Table B.17 SSIM evaluation of test bed demosaicking algorithms over the ARRI Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
arri_im1	0.999	0.989	0.999	0.998	0.996	0.993	0.998	0.987	0.996	
arri_im2	0.999	0.999	0.999	0.999	0.999	0.970	0.999	0.994	0.999	
arri_im3	1.000	0.999	0.999	0.999	0.996	0.972	0.999	0.998	0.999	
arri_im4	0.998	0.999	0.997	0.998	0.997	0.991	0.995	0.982	0.998	
arri_im5	1.000	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	
arri_im6	0.999	0.999	0.999	0.999	0.998	0.998	0.999	0.997	0.998	
arri_im7	0.999	0.986	0.999	0.998	0.998	0.998	0.998	0.995	0.998	
arri_im8	0.998	0.998	0.998	0.997	0.995	0.991	0.997	0.993	0.995	
arri_im9	0.998	0.999	0.998	0.999	0.997	0.641	0.996	0.993	0.997	
arri_im10	0.998	0.997	0.997	0.997	0.995	0.995	0.995	0.994	0.995	
arri_im11	0.999	0.999	0.999	0.999	0.998	0.990	0.999	0.965	0.998	
arri_im12	0.997	0.994	0.995	0.994	0.991	0.987	0.995	0.974	0.992	
average	0.998	0.996	0.998	0.998	0.997	0.954	0.998	0.989	0.997	5

Table B.18 SSIM evaluation of test bed demosaicking algorithms over the Custom Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
cusim01	0.995	0.993	0.993	0.995	0.994	0.992	0.991	0.988	0.994	
cusim02	0.995	0.991	0.991	0.993	0.992	0.989	0.992	0.985	0.990	
cusim03	1.000	0.999	0.999	0.999	0.998	0.976	0.999	0.995	0.997	
cusim04	0.998	0.997	0.996	0.997	0.996	0.993	0.996	0.989	0.999	
cusim05	0.999	0.992	0.987	0.992	0.984	0.978	0.992	0.953	0.992	
cusim06	0.997	0.995	0.994	0.996	0.994	0.987	0.993	0.988	0.997	
cusim07	0.997	0.995	0.995	0.997	0.995	0.992	0.994	0.984	0.993	
cusim08	0.997	0.992	0.992	0.995	0.984	0.950	0.991	0.988	0.993	
cusim09	0.997	0.992	0.990	0.991	0.922	0.860	0.991	0.985	0.993	
cusim10	0.996	0.992	0.993	0.994	0.988	0.947	0.991	0.974	0.992	
cusim11	0.998	0.997	0.996	0.997	0.996	0.994	0.996	0.995	0.996	
cusim12	0.996	0.993	0.994	0.994	0.994	0.988	0.993	0.967	0.996	
cusim13	0.997	0.994	0.994	0.996	0.994	0.992	0.993	0.986	0.997	
cusim14	0.999	0.998	0.995	0.997	0.995	0.984	0.996	0.995	0.996	
cusim15	0.999	0.999	0.998	0.998	0.998	0.951	0.998	0.997	0.998	
average	0.997	0.995	0.994	0.995	0.988	0.971	0.994	0.985	0.995	2

B.4 Feature Similarity Index with chrominance included (FSIM$_C$) Data

Note: the FSIM$_C$ values for image *sipi_im1* to *sipi_im8* of USC-SIPI were indeterminate in the ACR & EDCR algorithms due to the implementation of the FSIM algorithm. To ensure uniform analysis over all image sets, the geometric average was established using remaining images sipi_im9 to sipi_im16.

Table B.19 FSIM$_C$ evaluation of test bed demosaicking algorithms over the USC-SIPI Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
sipi_im1	0.980	0.974	0.939	0.957	0.936	0.928	-	-	0.932	
sipi_im2	0.981	0.977	0.947	0.963	0.944	0.938	-	-	0.939	
sipi_im3	0.962	0.944	0.872	0.901	0.853	0.852	-	-	0.853	
sipi_im4	0.976	0.966	0.932	0.950	0.926	0.911	-	-	0.924	
sipi_im5	0.974	0.961	0.907	0.931	0.884	0.879	-	-	0.884	
sipi_im6	0.963	0.957	0.927	0.939	0.913	0.908	-	-	0.913	
sipi_im7	0.974	0.953	0.882	0.904	0.856	0.867	-	-	0.873	
sipi_im8	0.978	0.966	0.905	0.927	0.883	0.891	-	-	0.898	
sipi_im9	0.997	0.995	0.976	0.988	0.973	0.938	0.981	0.939	0.986	
sipi_im10	0.992	0.987	0.972	0.975	0.961	0.914	0.973	0.971	0.999	
sipi_im11	0.977	0.974	0.962	0.964	0.958	0.924	0.957	0.934	0.986	
sipi_im12	0.995	0.989	0.971	0.978	0.967	0.935	0.973	0.968	0.982	
sipi_im13	0.996	0.992	0.975	0.980	0.964	0.929	0.976	0.974	0.986	
sipi_im14	0.982	0.980	0.971	0.975	0.969	0.945	0.974	0.968	0.988	
sipi_im15	0.990	0.987	0.977	0.981	0.974	0.950	0.978	0.971	0.988	
sipi_im16	0.994	0.991	0.983	0.987	0.975	0.939	0.983	0.963	0.987	
average	0.990	0.987	0.973	0.978	0.968	0.934	0.975	0.961	0.988	2

Table B.20 FSIM$_C$ evaluation of test bed demosaicking algorithms over the Kodak Image Set

	CDBI	**EDI**	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Prop**	**Rank**
kodim01	0.991	0.992	0.985	0.991	0.985	0.955	0.972	0.966	0.994	
kodim02	0.995	0.989	0.977	0.985	0.977	0.929	0.975	0.940	0.990	
kodim03	0.996	0.995	0.984	0.988	0.983	0.956	0.982	0.979	0.983	
kodim04	0.997	0.996	0.988	0.990	0.987	0.964	0.986	0.979	0.988	
kodim05	0.993	0.994	0.986	0.990	0.986	0.966	0.976	0.966	0.987	
kodim06	0.990	0.991	0.983	0.988	0.982	0.933	0.971	0.964	0.984	
kodim07	0.998	0.997	0.988	0.991	0.986	0.962	0.985	0.969	0.997	
kodim08	0.987	0.992	0.986	0.991	0.987	0.971	0.968	0.959	0.981	
kodim09	0.996	0.995	0.985	0.988	0.983	0.955	0.981	0.971	0.986	
kodim10	0.996	0.995	0.983	0.986	0.979	0.947	0.979	0.968	0.990	
kodim11	0.993	0.993	0.984	0.987	0.984	0.954	0.975	0.969	0.990	
kodim12	0.995	0.995	0.990	0.991	0.986	0.964	0.984	0.979	0.989	
kodim13	0.987	0.985	0.984	0.983	0.984	0.960	0.966	0.955	0.988	
kodim14	0.995	0.995	0.980	0.987	0.979	0.959	0.975	0.942	0.988	
kodim15	0.995	0.995	0.991	0.992	0.989	0.956	0.986	0.974	0.990	
kodim16	0.993	0.993	0.970	0.975	0.970	0.936	0.965	0.936	0.983	
kodim17	0.996	0.995	0.980	0.987	0.979	0.962	0.979	0.974	0.987	
kodim18	0.993	0.991	0.983	0.986	0.983	0.960	0.976	0.965	0.984	
kodim19	0.992	0.990	0.974	0.978	0.972	0.941	0.965	0.943	0.986	
kodim20	0.994	0.992	0.982	0.984	0.957	0.864	0.978	0.970	0.988	
kodim21	0.993	0.990	0.976	0.979	0.975	0.938	0.965	0.948	0.989	
kodim22	0.995	0.992	0.975	0.979	0.974	0.937	0.972	0.956	0.999	
kodim23	0.998	0.997	0.985	0.988	0.984	0.952	0.985	0.967	0.999	
kodim24	0.989	0.985	0.976	0.977	0.970	0.928	0.967	0.957	0.999	
average	0.994	0.993	0.982	0.986	0.980	0.948	0.976	0.962	0.989	3

Table B.21 FSIM$_C$ evaluation of test bed demosaicking algorithms over the McMaster-IMAX Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
mcm01	0.982	0.990	0.979	0.984	0.975	0.941	0.976	0.964	0.997	
mcm02	0.990	0.995	0.987	0.991	0.986	0.970	0.984	0.973	1.000	
mcm03	0.990	0.988	0.977	0.979	0.972	0.942	0.972	0.945	1.000	
mcm04	0.973	0.988	0.968	0.976	0.956	0.920	0.967	0.954	0.983	
mcm05	0.990	0.993	0.977	0.983	0.972	0.948	0.979	0.970	0.992	
mcm06	0.988	0.996	0.981	0.988	0.978	0.950	0.983	0.978	0.999	
mcm07	0.985	0.992	0.986	0.988	0.984	0.969	0.981	0.949	0.987	
mcm08	0.986	0.994	0.989	0.991	0.987	0.976	0.986	0.976	0.992	
mcm09	0.986	0.992	0.978	0.983	0.973	0.948	0.979	0.974	0.993	
mcm10	0.992	0.996	0.985	0.989	0.982	0.962	0.986	0.981	0.994	
mcm11	0.998	0.995	0.987	0.991	0.985	0.966	0.988	0.973	0.993	
mcm12	0.996	0.992	0.968	0.972	0.963	0.925	0.969	0.966	0.979	
mcm13	0.995	0.984	0.961	0.964	0.956	0.911	0.964	0.961	0.994	
mcm14	0.998	0.994	0.984	0.988	0.981	0.962	0.986	0.983	0.993	
mcm15	0.997	0.994	0.981	0.986	0.979	0.956	0.983	0.980	0.999	
mcm16	0.997	0.995	0.988	0.990	0.984	0.969	0.986	0.959	0.990	
mcm17	0.996	0.995	0.990	0.992	0.987	0.974	0.989	0.964	0.994	
mcm18	0.993	0.993	0.981	0.986	0.978	0.959	0.970	0.958	0.989	
average	0.991	0.993	0.980	0.984	0.977	0.953	0.979	0.967	0.993	1

Table B.22 FSIM$_C$ evaluation of test bed demosaicking algorithms over the Condat Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
codim01	0.983	0.992	0.978	0.983	0.976	0.934	0.972	0.957	0.987	
codim02	0.985	0.993	0.980	0.985	0.977	0.940	0.979	0.971	0.997	
codim03	0.995	0.993	0.981	0.985	0.978	0.941	0.978	0.967	0.977	
codim04	0.994	0.992	0.980	0.985	0.977	0.952	0.975	0.967	0.987	
codim05	0.996	0.989	0.975	0.978	0.971	0.939	0.973	0.920	0.986	
codim06	0.993	0.994	0.986	0.992	0.985	0.972	0.985	0.934	0.988	
codim07	0.983	0.988	0.971	0.975	0.962	0.903	0.967	0.954	0.985	
codim08	0.988	0.988	0.975	0.979	0.970	0.948	0.971	0.892	0.986	
codim09	0.988	0.985	0.979	0.980	0.974	0.934	0.969	0.914	0.987	
codim10	0.995	0.992	0.971	0.979	0.967	0.928	0.971	0.957	0.985	
codim11	0.989	0.991	0.983	0.986	0.979	0.961	0.979	0.924	0.987	
codim12	0.988	0.988	0.977	0.983	0.976	0.948	0.964	0.867	0.986	
codim13	0.989	0.990	0.982	0.986	0.980	0.947	0.972	0.956	0.997	
codim14	0.990	0.993	0.981	0.986	0.978	0.954	0.982	0.972	0.987	
codim15	0.989	0.990	0.974	0.981	0.972	0.946	0.968	0.950	0.998	
codim16	0.998	0.996	0.986	0.990	0.984	0.968	0.986	0.977	0.988	
codim17	0.990	0.985	0.971	0.976	0.944	0.911	0.968	0.931	0.988	
codim18	0.992	0.988	0.971	0.976	0.964	0.938	0.967	0.895	0.985	
codim19	0.990	0.991	0.976	0.980	0.971	0.947	0.973	0.913	0.976	
codim20	0.990	0.993	0.976	0.981	0.974	0.942	0.975	0.917	0.978	
codim21	0.990	0.989	0.977	0.981	0.971	0.945	0.976	0.963	0.986	
codim22	0.989	0.993	0.987	0.989	0.985	0.934	0.982	0.952	0.979	
codim23	0.990	0.999	0.993	0.996	0.992	0.978	0.991	0.950	0.987	
codim24	0.989	0.991	0.979	0.983	0.977	0.951	0.973	0.967	0.968	
codim25	0.990	0.996	0.985	0.990	0.984	0.962	0.986	0.966	0.975	
codim26	0.991	0.988	0.970	0.975	0.968	0.937	0.963	0.882	0.975	
codim27	0.989	0.994	0.988	0.991	0.987	0.969	0.983	0.969	0.982	
codim28	0.990	0.995	0.978	0.985	0.976	0.948	0.980	0.933	0.976	
codim29	0.988	0.997	0.992	0.994	0.990	0.978	0.991	0.938	0.987	
codim30	0.982	0.988	0.972	0.977	0.965	0.935	0.968	0.945	0.975	
average	0.990	0.991	0.979	0.984	0.975	0.946	0.976	0.940	0.984	3

Table B.23 FSIM$_C$ evaluation of test bed demosaicking algorithms over the ARRI Image Set

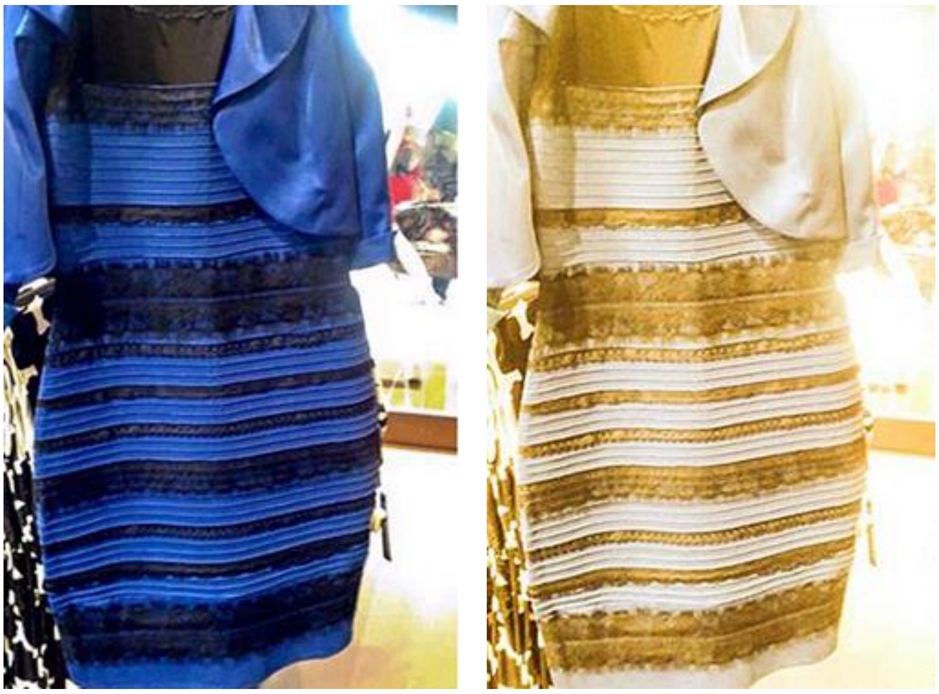
	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
arri_im1	0.989	0.991	0.999	1.000	0.997	0.989	0.998	0.989	0.996	
arri_im2	0.989	0.991	1.000	0.999	0.997	0.972	0.998	0.990	0.997	
arri_im3	0.989	0.991	1.000	0.990	0.994	0.975	0.999	0.997	0.997	
arri_im4	0.990	0.990	0.999	0.997	1.000	0.998	0.999	0.971	0.999	
arri_im5	0.991	0.991	1.000	0.998	1.000	0.999	1.000	0.998	0.999	
arri_im6	0.990	0.992	0.999	0.998	0.997	0.994	0.998	0.996	0.997	
arri_im7	0.990	0.999	0.999	0.999	0.999	0.998	0.999	0.995	0.999	
arri_im8	0.991	0.999	1.000	0.999	0.999	0.993	0.999	0.991	0.999	
arri_im9	0.990	0.998	0.999	0.998	0.998	0.788	0.998	0.994	0.997	
arri_im10	0.990	0.996	0.999	0.998	0.999	0.997	0.999	0.997	0.999	
arri_im11	0.990	0.997	1.000	0.998	0.998	0.989	0.999	0.956	0.998	
arri_im12	0.991	0.998	0.999	0.999	0.998	0.991	0.998	0.980	0.997	
average	0.990	0.994	0.999	0.998	0.998	0.972	0.999	0.988	0.998	3

Table B.24 FSIM$_C$ evaluation of test bed demosaicking algorithms over the Custom Image Set

	CDBI	EDI	MHC	Wang	ESFBI	MGBI	ACR	EDCR	Prop	Rank
cusim01	0.991	0.988	0.996	0.998	0.994	0.988	0.986	0.990	0.992	
cusim02	0.993	0.988	0.995	0.997	0.994	0.984	0.990	0.983	0.992	
cusim03	0.997	0.989	0.997	0.998	0.996	0.975	0.990	0.987	0.994	
cusim04	0.998	0.989	0.998	0.999	0.998	0.995	0.998	0.986	0.996	
cusim05	0.983	0.987	0.994	0.997	0.993	0.981	0.993	0.938	1.000	
cusim06	0.989	0.988	0.996	0.997	0.994	0.982	0.995	0.987	0.993	
cusim07	0.989	0.988	0.996	0.997	0.995	0.986	0.991	0.983	0.993	
cusim08	0.989	0.987	0.995	0.997	0.988	0.939	0.993	0.991	0.991	
cusim09	0.989	0.986	0.993	0.996	0.973	0.917	0.990	0.985	0.991	
cusim10	0.989	0.986	0.995	0.997	0.989	0.929	0.995	0.970	0.999	
cusim11	0.989	0.987	0.995	0.997	0.993	0.985	0.995	0.994	0.999	
cusim12	0.990	0.987	0.993	0.996	0.991	0.969	0.994	0.964	0.999	
cusim13	0.989	0.988	0.994	0.997	0.994	0.984	0.995	0.982	0.997	
cusim14	0.990	0.989	0.997	0.998	0.996	0.982	0.998	0.997	0.997	
cusim15	0.990	0.989	0.997	0.999	0.996	0.956	0.998	0.996	0.994	
average	0.990	0.988	0.995	0.997	0.992	0.970	0.993	0.982	0.995	2

Appendix C: Image Sets and Camera Resolution Chart

The images provided below have been employed in the analysis of this work and are readily available online or upon e-mail request to the author via the following addresses: kinyua.wachira@students.uonbi.ac.ke or kinyua.wachira@gmail.com.



(a) (b)

Figure C.1 The Dress image where (a) the blue and black version is the actual dress and (b) the white and gold version was the image posted online (source [21])

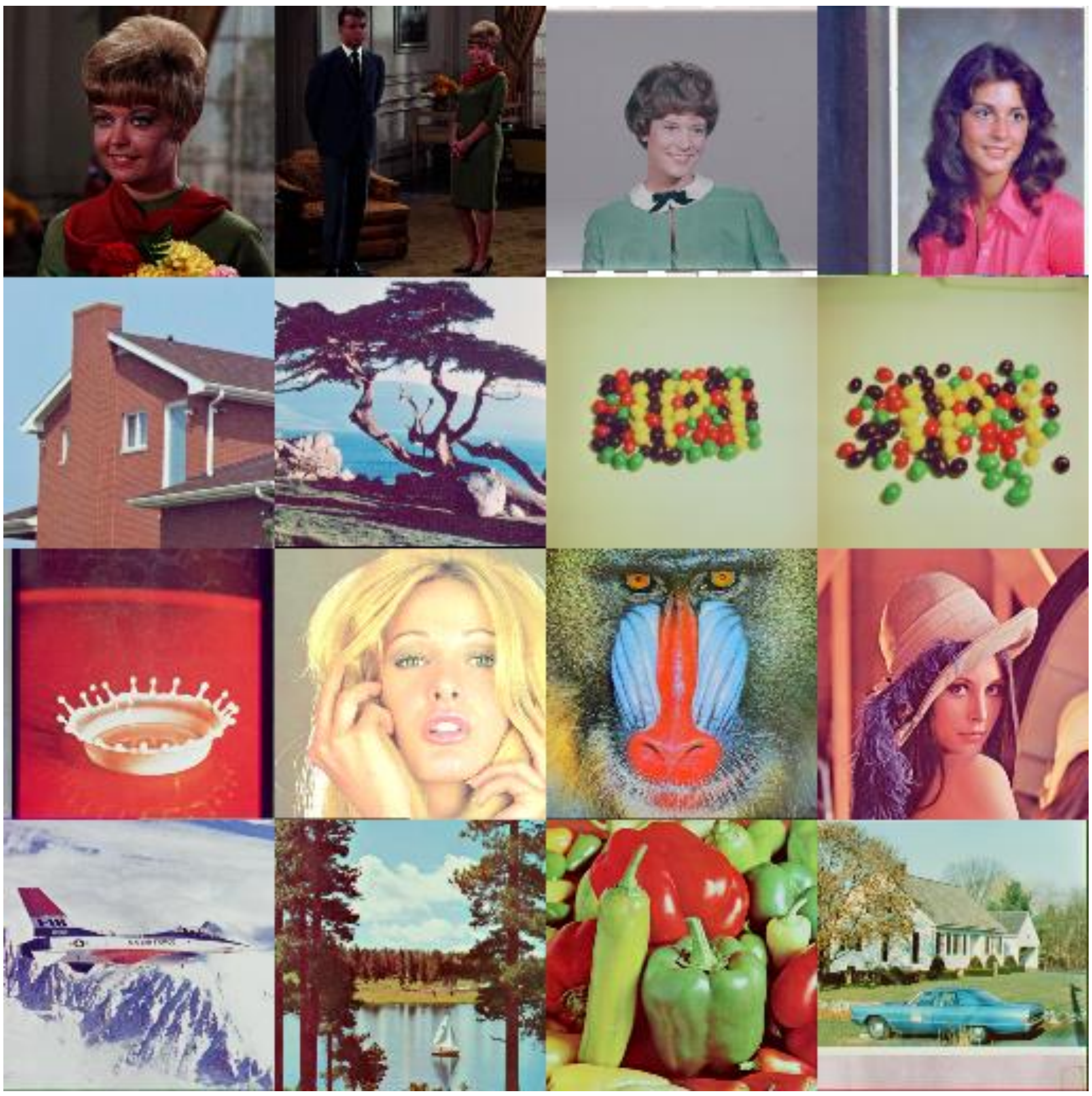


Figure C.2 The USC-SIPI Image Set: sipi_im01 to sipi_im16; viewed from top to bottom, left to right (source [128])

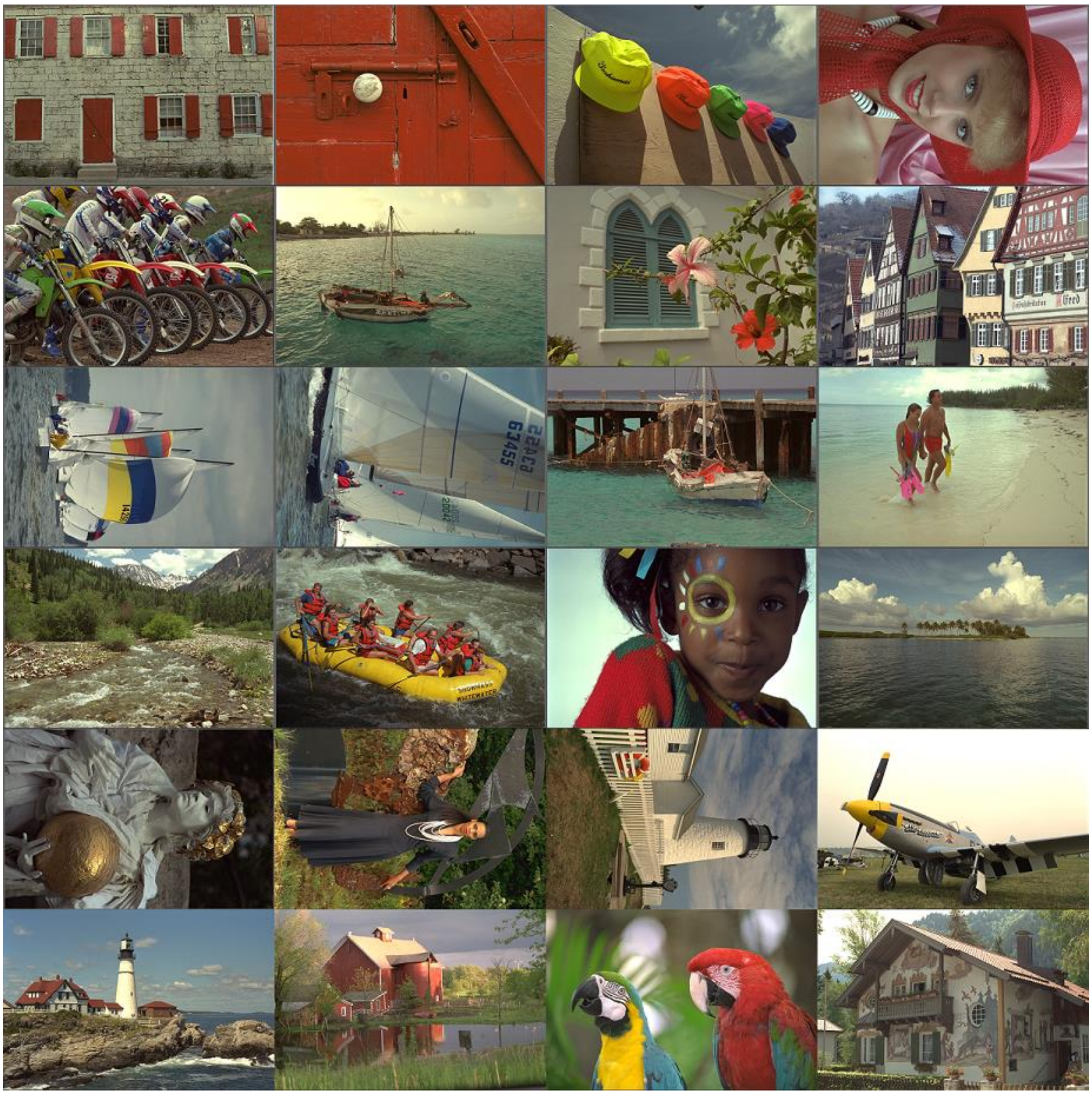


Figure C.3 The Kodak Set: kodim01 to kodim24; viewed from top to bottom, left to right (source [123])

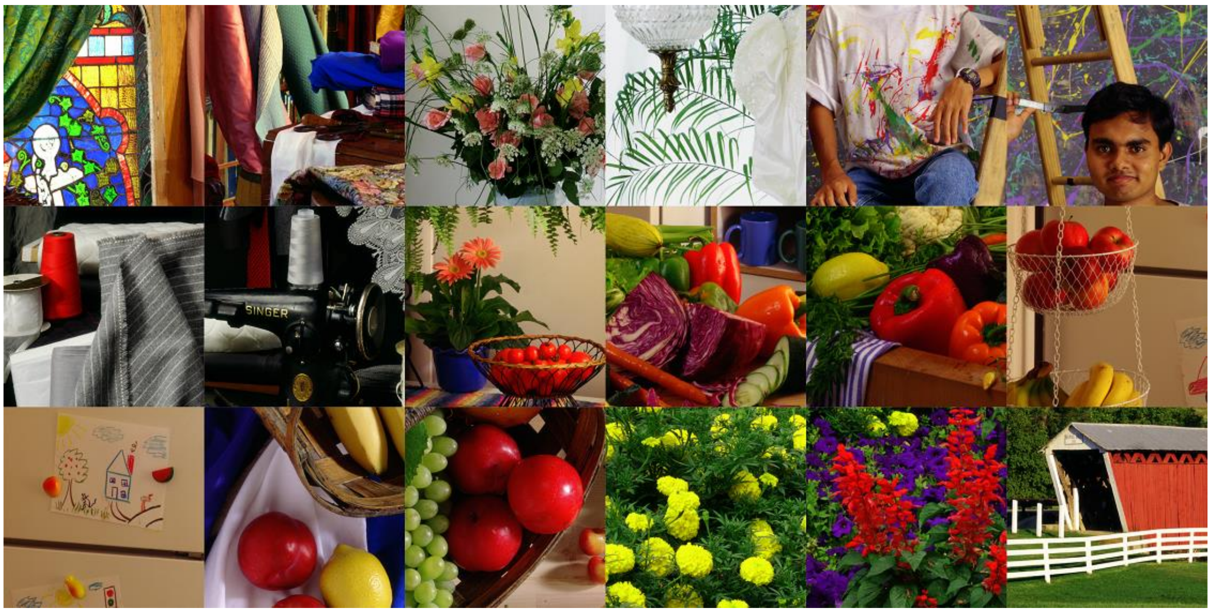


Figure C.4 McMaster-IMAX Set: mcm01 to mcm18; viewed from top to bottom, left to right (source [124])

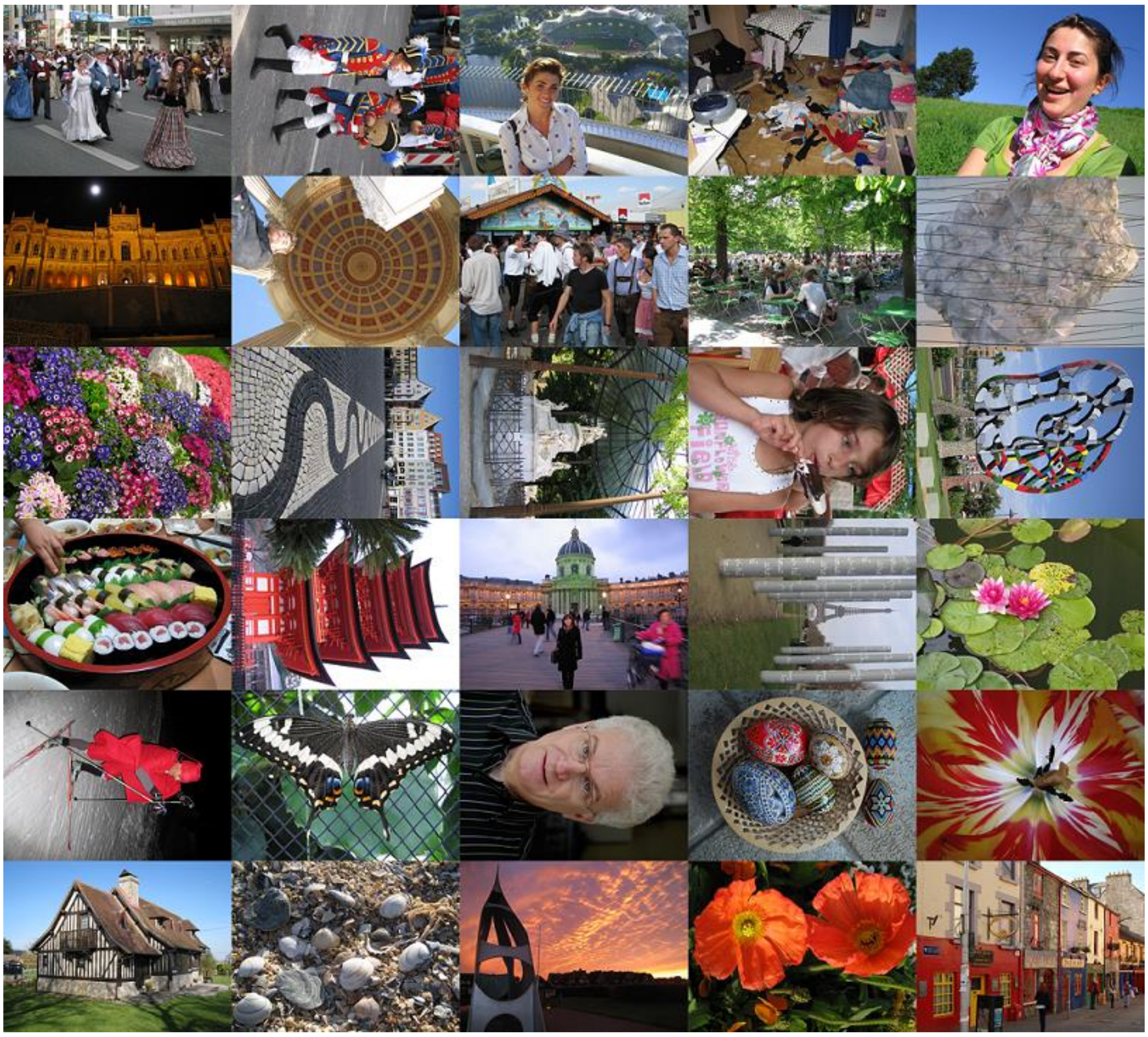


Figure C.5 Condat Set: codim01 to codim30; viewed from top to bottom, left to right (source [125])

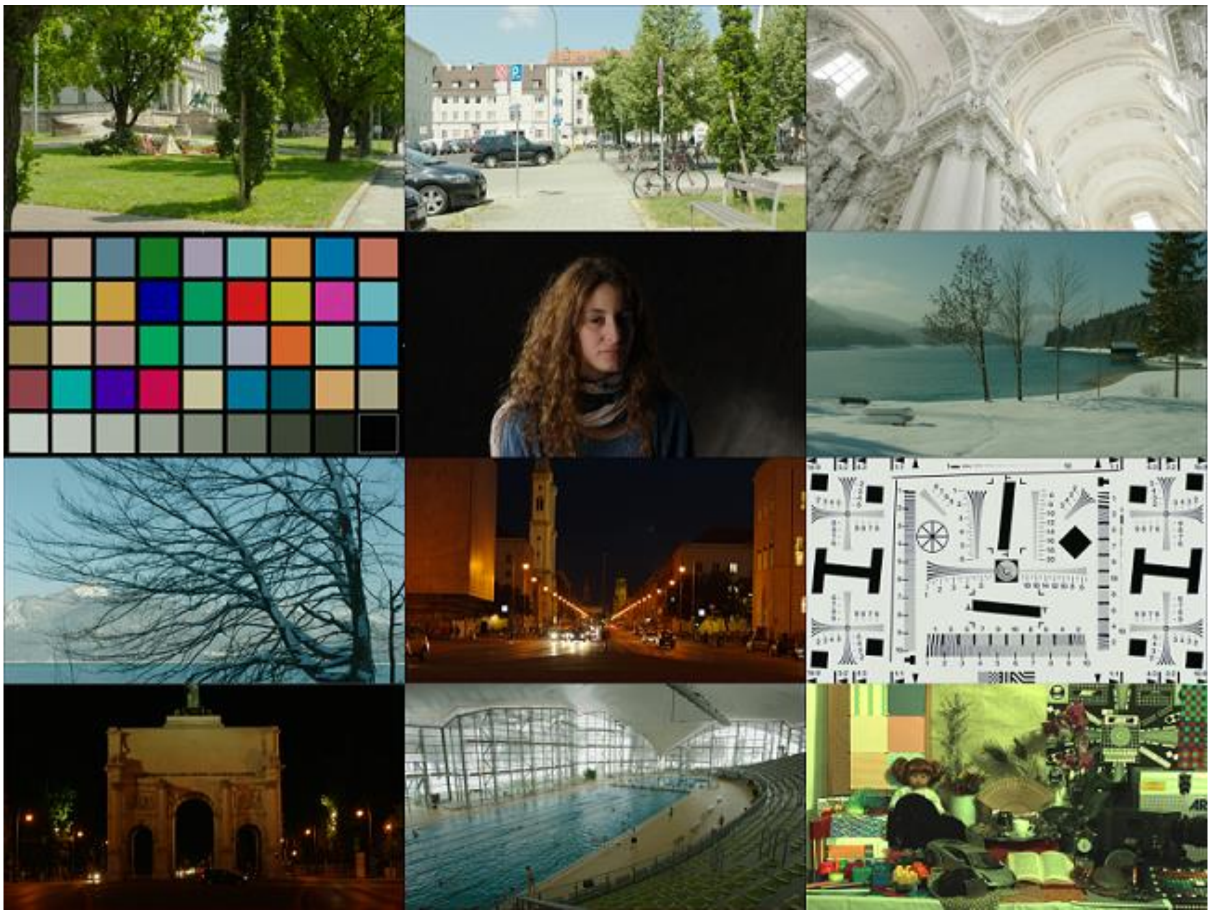


Figure C.6 The ARRI Set: arri_im01 to arri_im12; viewed from top to bottom, left to right (source [130])

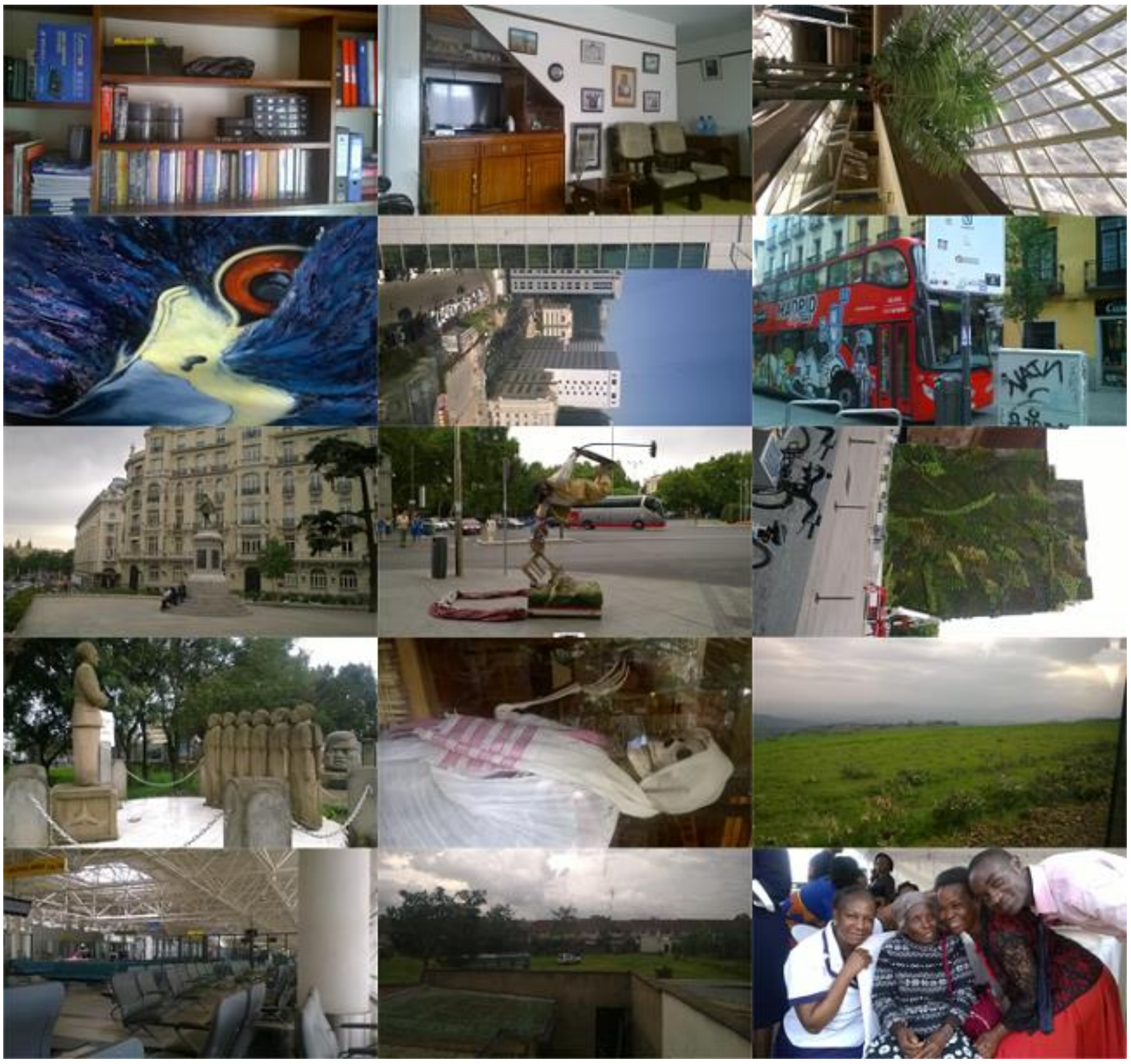


Figure C.7 A Custom Image Set: cusim01 to cusim15; viewed from top to bottom, left to right; developed by the author.

Table C.1 Resolution Specification Chart

Digital Camera Resolution Chart								
Capture Resolution	**Video Display***	**Print Size*****						
		2x3"	**4x5"/4x6"**	**5x7"**	**8x10"**	**11x14"**	**16x20"**	**20x30"**
320x240	Acceptable	Good	Acceptable	Poor	Poor	Poor	Poor	Poor
640x480 (0.3 Megapixel)	Good	Excellent	Good	Poor	Poor	Poor	Poor	Poor
800x600	Excellent	Photo Quality	Very Good	Acceptable	Poor	Poor	Poor	Poor
1024x768	Excellent	Photo Quality	Excellent	Good	Acceptable	Poor	Poor	Poor
1280x960 (1 Megapixel)	Excellent	Photo Quality	Photo Quality	Very Good	Good	Poor	Poor	Poor
1536x1180	Excellent**	Photo Quality	Photo Quality	Excellent	Very Good	Acceptable	Poor	Poor
1600x1200 (2 Megapixel)	Excellent**	Photo Quality	Photo Quality	Photo Quality	Very Good	Acceptable	Acceptable	Poor
2048x1536 (3 Megapixel)	Excellent**	Photo Quality	Photo Quality	Photo Quality	Excellent	Good	Acceptable	Acceptable
2240x1680 (4 Megapixel)	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Very Good	Good	Acceptable
2560x1920 (5 Megapixel)	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Excellent	Very Good	Very Good
3032x2008 (6 Megapixel)	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Excellent	Very Good
3072x2304 (7 Megapixel)	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Excellent	Excellent
3264x2448 (8 Megapixel)	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Excellent
10 Megapixel and Above	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality

* *A television or computer display*

** *Will produce an excessively large file size that would be inappropriate for web applications*

*** *Using a typical Photo Quality Desktop printer*

Where:

i. **Poor**: the image is noticeably pixelated
ii. **Acceptable**: only coarse details are visible in the image
iii. **Good**: coarse and fine details are visible in the image
iv. **Very Good**: image is an adequate scene representation for most people
v. **Excellent**: to the human eye, the image is indistinguishable from the original scene at a normal viewing distance
vi. **Photo Quality**: on a photo-quality printer, to the human eye, the image is indistinguishable from the original scene at a normal viewing distance

Appendix D: Human Trichromacity

The spectral curves derived from Hunt [16] are presented in Figures D.1 and D.2 for reference.

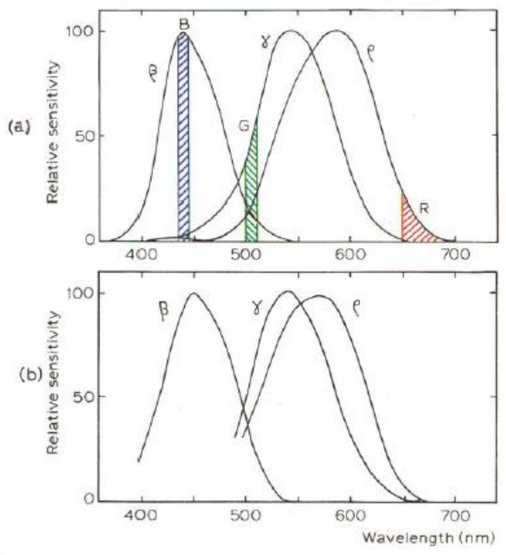
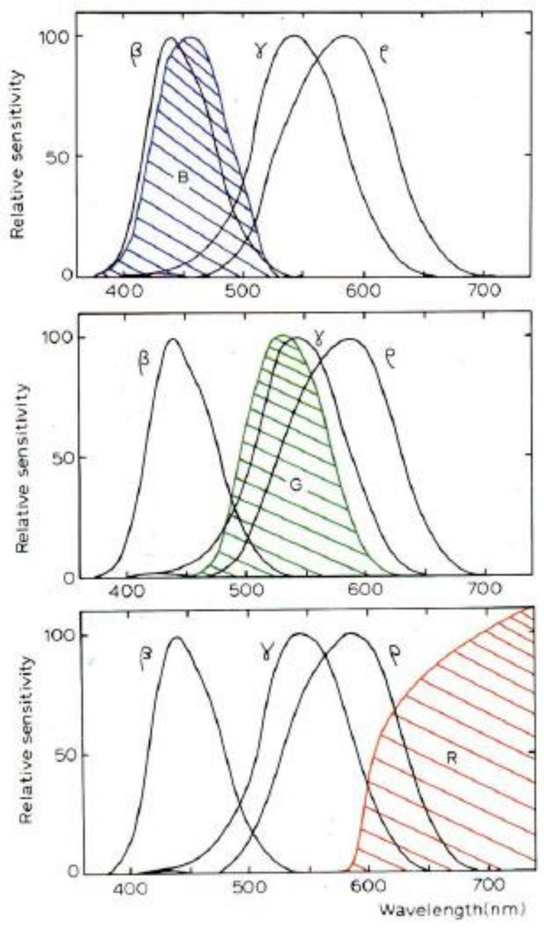
Figure D.1	Figure D.2
	

Figure D.1 (a) The probable sensitivity curves β, γ, and ρ determined by indirect methods together with the spectral quality points of R, G, and B (b) Spectral sensitivity curves found from bleaching experiments on pigments in the human retina

Figure D.2 The ρ, γ, β sensitivity curves and the spectral powers of light transmitted by red, green and blue filters typically used in additive colour reproduction

Appendix E: Visual Artefacts and Aberrations

E.1 Selected Optical Effect Artefacts

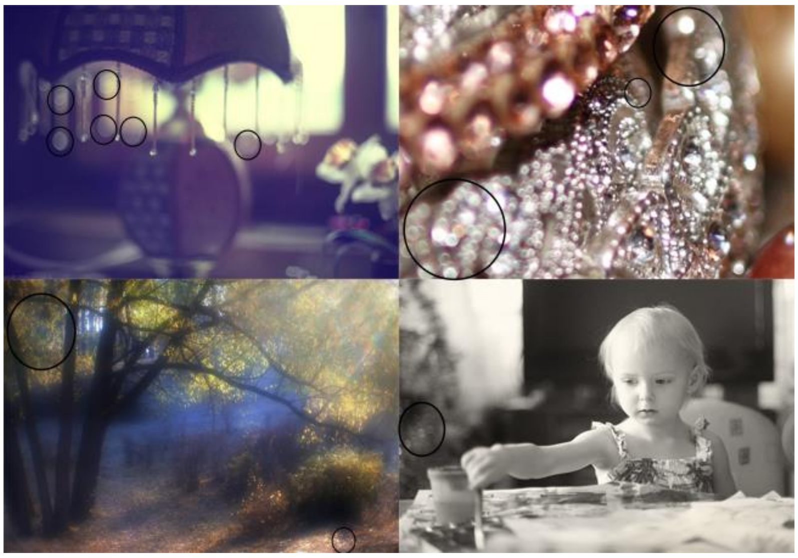


Figure E.1 Selected images highlighting spherical aberration

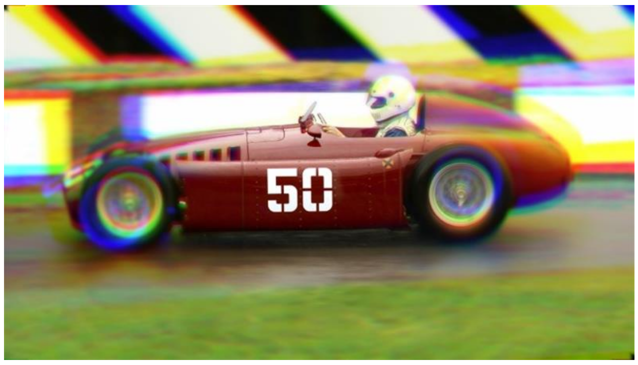


Figure E.2 Image illustrating chromatic aberration

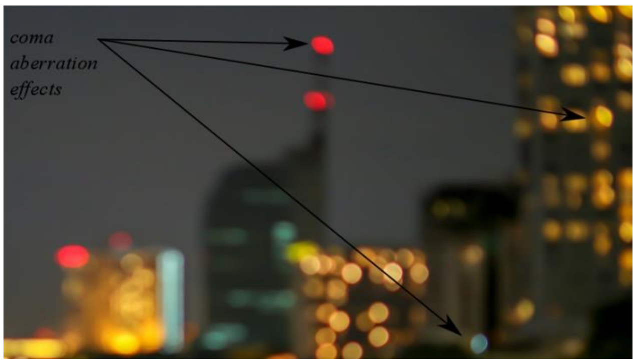


Figure E.3 Image illustrating comatic aberration

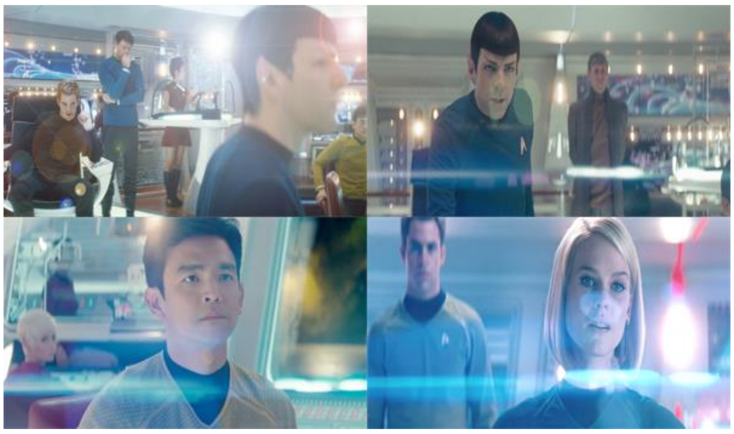


Figure E.4 Camera stills from the Star Trek films illustrating lens flare phenomena

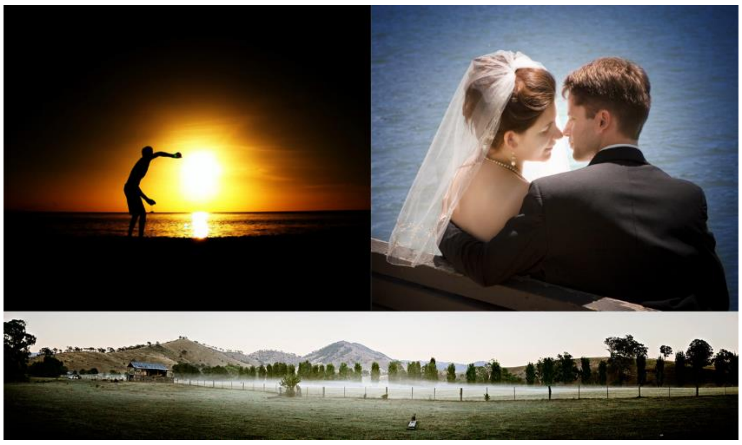


Figure E.5 Selected images illustrating vignette effects

E.2 Image Noise Effects

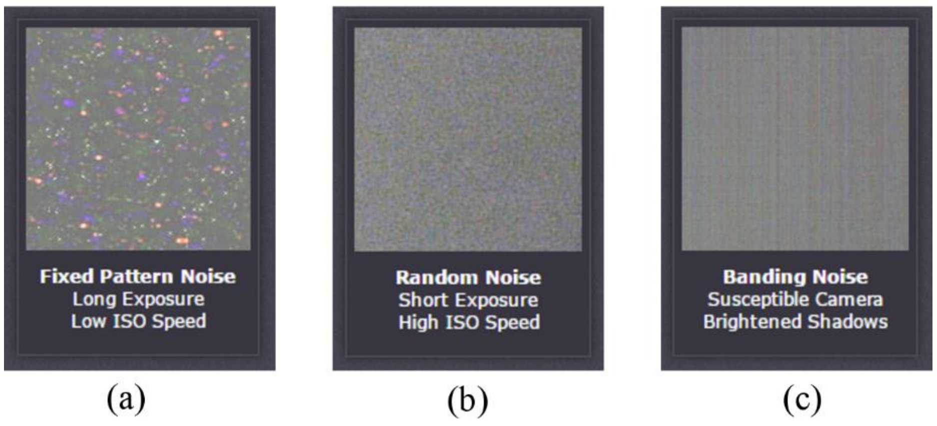


Figure E.6 Types of image noise: (a) fixed pattern (b) random and (c) banded

E.3 Demosaicking Artefacts

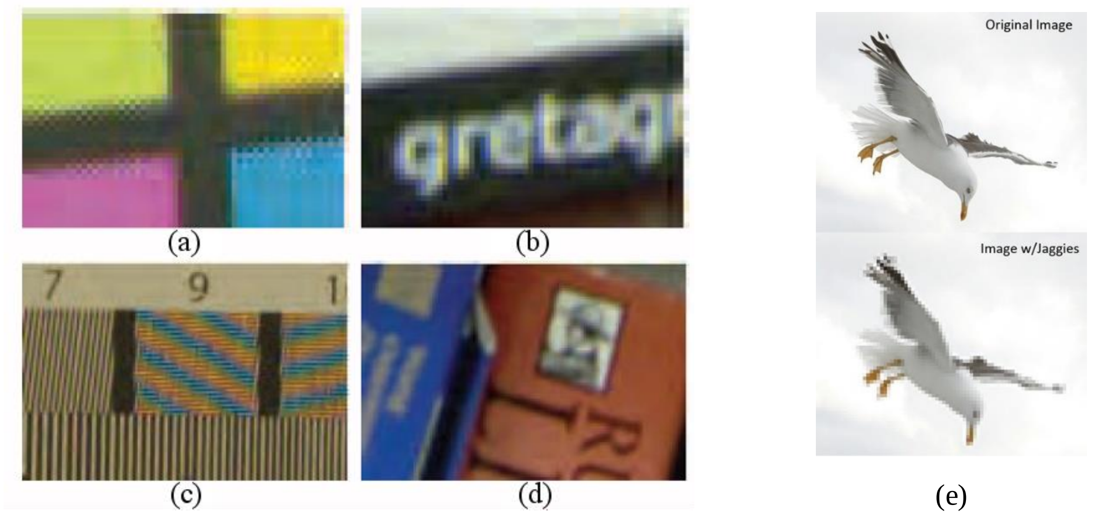


Figure E.7 Selected images showing (a) Zipper effect (b) Colour Shifts (c)Moiré effect, (d) Blurring and (e) Jaggies (source: [33])

E.4 Coloration and Exposure Shifts

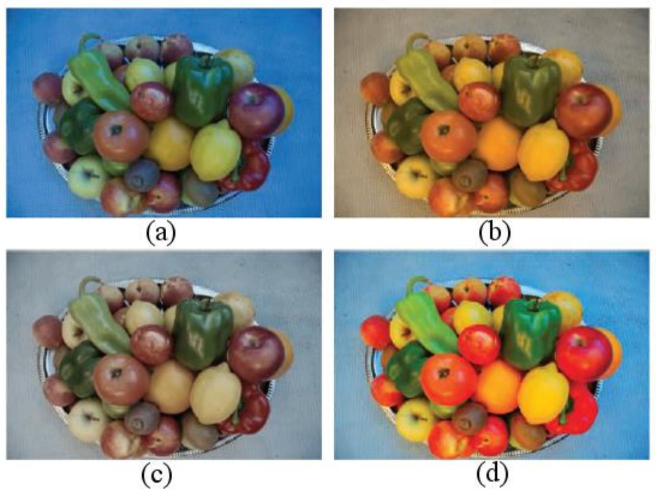


Figure E.8 Image showing coloration shifts: (a) cool appearance, (b) warm appearance, (c) grey appearance and (d) saturation effects (source [33])

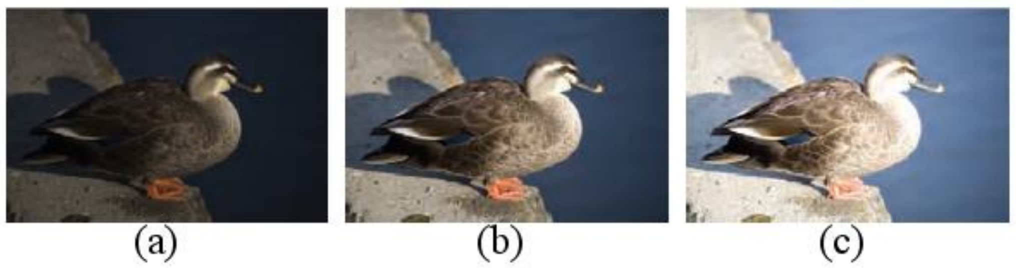


Figure E.9 Image showing exposure shifts: (a) underexposure, (b) normal exposure and (c) overexposure (source [33])

Appendix F: Publication Statistics

A general term search analysis was performed to identify the level of active interest in single sensor image demosaicking. The three main image processing repositories: the IEEE Xplore library [141], the Springer Link repository [142] and the SPIE Digital library [143] were queried to find the number of times the terms '*demosaicking'* and *'demosaicing'* appeared in titles of journals and conference papers. The results are shown below.

Table F.1 Search term statistics for the words 'demosaicking' and 'demosaicing' in a publication's title

Search Term	IEEE Xplore Library	Springer Link Repository	SPIE Digital Library
'demosaicking'	258	44	37
'demosaicing'	252	93	31

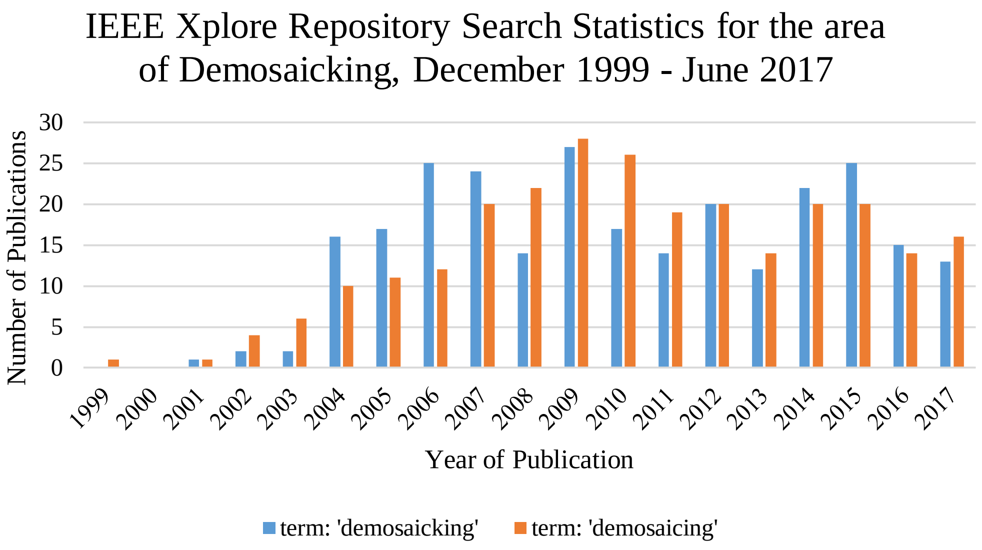


Figure F.1 Demosaicking publication trend in the IEEE Xplore repository

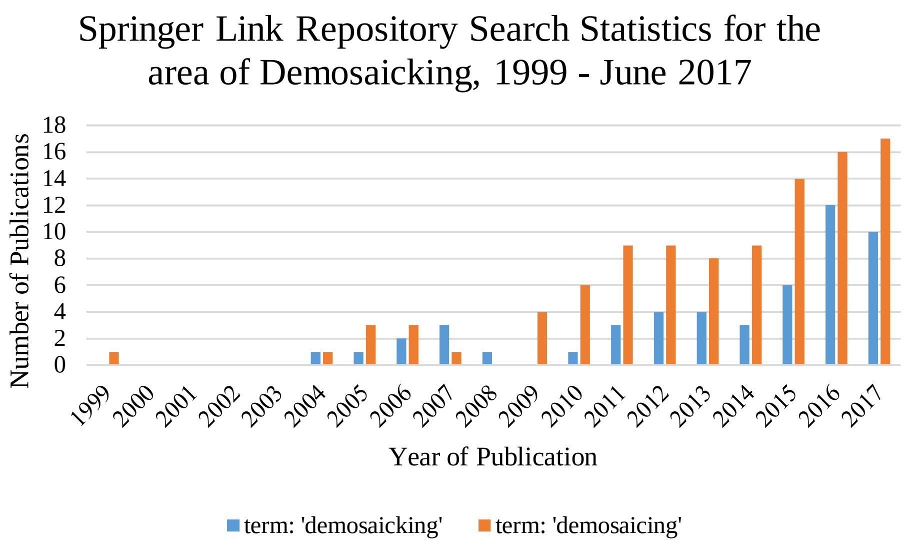


Figure F.2 Demosaicking publication trend in the Springer Link repository

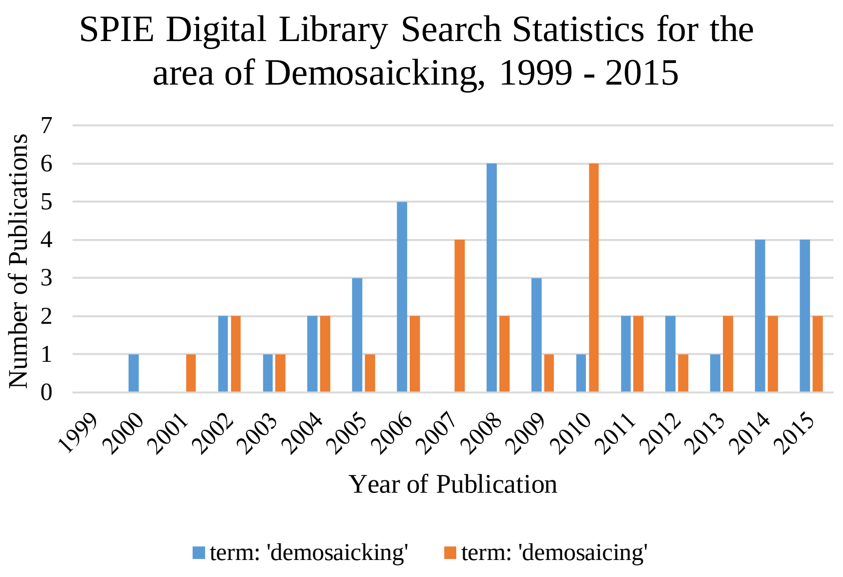


Figure F.3 Demosaicking publication trend in the SPIE Digital Library repository

Appendix G: Author's Publications

G.1 First Publication – IEEE SPICES 2015

A Cardinal-Direction Quincunx Based Interpolation Technique with Non-Uniform Inter-Plane Weighting for Bayer CFA Demosaicking

Kinyua Wachira and Elijah Mwangi
School of Engineering,
University of Nairobi,
Kenya
Email: kinyua.wachira@students.uonbi.ac.ke

***Abstract*—This paper presents a new weighted interpolation technique to address the Bayer Color Filter Array (CFA) demosaicing problem. The proposed technique provides two contributions that have not been reported in conventional interpolation methods. Firstly, it exploits the quincuncial nature of the green component of the Bayer CFA. The second contribution treats each of the three planes or lattices of the CFA mosaic as distinct and is weighted uniquely. The proposed technique is compared with other interpolation-based demosaicking algorithms and a significant improvement in performance has been noted through experimental results. In addition, to provide objectivity in analysis, three test measures have been used - CPSNR, CIELAB and CIEDE2000.**

Keywords*—*Demosaicking, inter-plane weighting, positive contributors, negative contributors, quincunx.

I. Introduction

The digital still camera has become an ubiquitous feature in human society. It is found either as a stand-alone object or integrated into another device. The process of image capture consists of several stages such as focus and exposure control, white balance adjustment, demosaicking, color transformation, correction [1]. This paper concerns itself with the demosaicking stage.

To produce a color image, there should be three color samples per sampling point; usually red, green and blue [2]. To achieve this, the digital camera has either a three color sensor regime or a single color sensor type that incorporates a technique to establish the unsampled points [3]. The second method is preferred as it reduces the camera's cost and complexity. A Color Filter Array (CFA) is used to perform the discrimination of the incident light from the lens into a one color per pixel sensor. The particular CFA in Figure 1 is the most commonly used and documented; the Bayer CFA, developed in 1976 [4].

The image on camera sensor follows the orientation set by the CFA. A demosaicing algorithm is used to reconstruct each of the constituent color planes from the CFA representation. Due to the popularity of CFA based cameras, the demosaicking problem has been and still is an area of extensive study. Li *et al.* [3] made a recent survey and classified demosaicking algorithms as either motivated in the spatial-domain, the frequency-domain such as in [5] or a hybrid of the two [6]. This paper is biased towards spatial domain based demosaicking techniques

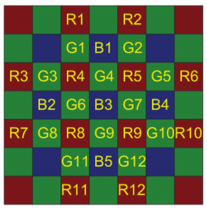


Fig. 1. The Bayer CFA arrangement in a 7-by-7 window

as they produce good, robust results and they are of moderate computational complexity.

In the simplest case, spatial demosaicking can be implemented by Bilinear Interpolation (BI). However, the high correlation between different color planes is not exploited. This results in color abnormalities in the final reconstructed image [7]. The constant hue based interpolation techniques assume that the hue content in an image is roughly constant. This assumption has been exploited for non-green planes in [8]. The main methods here are the Constant Difference Based Interpolation (CDBI) and Constant Ratio Based Interpolation (CRBI) [7].

Edge Directed Interpolation (EDI) introduced by Laroche *et al.* [9] is an adaptive method that uses neighbor information to determine which edge to interpolate along. Hamilton and Adams [10] and Chang *et al.* [11] extend this method by using gradients as correction terms in the red and blue planes. A more recent variant of EDI is Direction Weighted Interpolation (DWI) technique [12]. This method not only collects edge information but each edge is weighted by a gradient-based factor. This method in turn has been extended recently in various ways such as the Malvar-He-Cutler (MHC) algorithm in [13], the Wang method [14] and the Multi-Directional Weighted Interpolation (MDWI) [15].

Our proposed algorithm extends the preliminary section of the MDWI method by offering two novel contributions: an exploitation of the quincunx arrangement of the green plane and non-uniform inter-plane weighting.

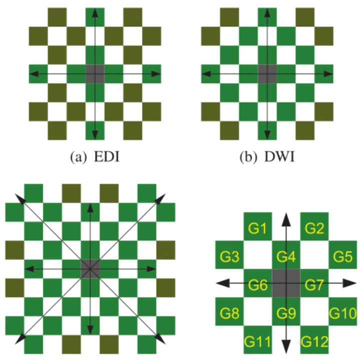
The remainder of this paper is divided as follows. Section II explains the two new contributions in detail before the entire algorithm is presented. Section III shows the objective evaluation mechanisms chosen and the rationale behind them. Finally Section IV briefly presents the conclusions that are drawn from experimental results.

II. Proposed Technique

A. Full Exploitation of the Quincunx Arrangement

The proposed technique offers two novel properties to conventional weighted interpolation. The first is exploitation of the full quincunx of the window bounding our pixel of interest. This arrangement is particular to the Bayer CFA and its direct variants.

Figure 2 shows some demosaicking techniques and their quincunx arrangements in the desired pixel's region of interest. Used points are marked in green, unused ones are shown in olive and the desired pixel point is marked in gray. In EDI, DWI and MDWI cases, not all the green pixels in the quincunx arrangement of the bounding window are used. In these methods the window is made large to give a better estimation in the interpolation direction. Size is crucial because a small window gives insufficient points and a large one introduces errors due to distance from the desired pixel. The effect of inaccurate points in a large window can be reduced by forcing them to make a small contribution. However that, in the limit, leads back to the small window problem.



(a) EDI (b) DWI

(c) MDWI (d) Proposed

Fig. 2. A comparison of effective pixel usage in the green quincunx plane

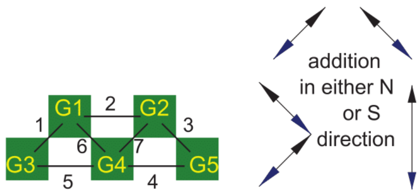
The proposed method instead picks all members of the quincunx arrangement in a 5-by-5 bounding window and classifies them in the cardinal directions. In Figure 2(d) pixels G1,G2,G3,G4,G5 contribute to the north direction since they all appear north of the desired pixel. By the same token G2,G5,G7,G10 and G12 are used for the east direction. A measure of quincunx use is given by equation 1. In terms of η_Q, EDI has 0.333; DWI has 0.667; MDWI has 0.8 and the proposed method has a $\eta_Q = 1$.

$$\eta_Q = (G_{used})/(G_{total}) \tag{1}$$

Conventional weighted techniques have the gradient made up of a sum of positive differences as the pixels chosen already conform to the desired direction. In the proposed technique, a sufficient weighted sum will consist of positive (G_{diff+}) and negative (G_{diff-}) terms. This is because there are some green pixel differences in line with the desired edge while others are in directional opposition to the edge.

$$\triangledown G = \sum_m G_{diff+} + \sum_n G_{diff-} \tag{2}$$

Consider the green pixels in Figure 3(a). There are 7 paths between closest neighbors between the 5 pixels. Paths 1,3,6 and 7 are diagonal. From a 'pseudo-vector' viewpoint shown in Figure 3(b), the overall direction follows a vertical path. Therefore, differences along these paths are taken as positive.



(a) Subset from Figure 2(d) (b) Pseudo-vector addition

Fig. 3. Quincunx paths

Paths 2,4 and 5 are all horizontal. This direction being opposite to the vertical lets us treat the paths as negative contributors. So in equation 2 considering the North direction, $\sum_m G^N_{diff+} = (|G4-G1|+|G4-G2|+|G3-G1|+|G5-G2|)$ and $\sum_n G^N_{diff-} = (|G4-G3| + |G4-G5| + |G1-G2|)$.

To further refine equation 2, two considerations are made. First, the term of path 2 is ignored due to its distance from the desired pixel point. Second, it is undesirable to interpolate across edges [7]. As such the negative contributors are weighted with a factor that is smaller than that used by the positive contributors. For the North direction example:

$$\triangledown G^N = \{c_1 \sum_m G^N_{diff_+}\} - \{c_2 \sum_{n-1} G^N_{diff_-}\} \tag{3}$$

where $c_2 < c_1$. It was determined empirically that $c_1 = 1$ and $c_2 = 0.707$ gave good results.

B. Non-uniform Inter-plane Weighting

A Bayer CFA is a mosaic of 3 separate color planes [4], that are essentially superimposed on one another. If a section of Figure 1 is split plane-wise, Figure 4 is obtained.

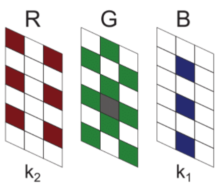


Fig. 4. A Bayer CFA portion split into its constituent planes

Consider the process of determining the pixel point marked gray in Figure 4. The desired information is the green content in a blue pixel point. Conventional interpolation techniques employ the following general equation:

$$\triangledown = \triangledown G + \triangledown B + \triangledown R \tag{4}$$

and $\triangledown X = \sum X_{diff}$ where $X \subset G, B, R$.

From equation 4, it is noted that all planes are treated as equal or $k_1 = k_2 = 1$ in Figure 4. The authors propose that if k_1 and k_2 are taken as variable, there are three possible inter-plane relations. Either $k_1 = k_2 = 1$ or $k_1 = k_2 \neq 1$ or $k_1 \neq k_2 \neq 1$.

In terms of weight priority in Figure 4, intuitively green is first, followed by blue then red. The green plane is where the missing color resides. The blue plane shares the same pixel location as the desired color. The red plane does not contain either the missing color or the pixel point and should contribute the least. The proposed algorithm quantifies this priority by using non-uniform inter-plane weighting where $k_2 < k_1 < 1$. This is a subset of the relation $k_1 \neq k_2 \neq 1$. From empirical testing, it was established that $k_1 = 0.8$ and $k_2 = 0.7$ gave good overall results. This generalizes equation 4 to:

$$\triangledown = \triangledown G + \triangledown B' + \triangledown R' \tag{5}$$

where $\triangledown B' = k_1 \times \triangledown B$ and $\triangledown R' = k_2 \times \triangledown R$.

C. Interpolation of Missing Components in the Green Plane

Consider establishing the green content value at pixel B3 of the Bayer CFA in Figure 1. Initial estimates are first made in the cardinal directions:

$$\begin{aligned} G^N_{B3} &= G4 + \{(k_1k_2)(B3 - B1)\} \\ G^W_{B3} &= G6 + \{(k_1k_2)(B3 - B2)\} \\ G^E_{B3} &= G7 + \{(k_1k_2)(B3 - B4)\} \\ G^S_{B3} &= G9 + \{(k_1k_2)(B3 - B5)\} \end{aligned} \tag{6}$$

Inter-plane weighting is used as the estimates will in the final analysis will be weighted by both red and blue planes. The gradients in each cardinal directions are then established following equation 4. Consider the West direction:

$$\triangledown^W_{B3} = \triangledown G^W_{B3} + \triangledown B^W_{B3} + \triangledown R^W_{B3} \tag{7}$$

The green contribution of the gradient will use the quincunx arrangement while the red and blue contributions of the gradient will use the inter-plane weighting concept.

$$\triangledown G^W_{B3} = c_1 \sum G^W_{B3+} - c_2 \sum G^W_{B3-} \tag{8}$$

where $\sum G^W_{B3+} = (|G6 - G3| + |G6 - G8| + |G1 - G3| + |G11 - G8|)$ and $\sum G^W_{B3-} = (|G6 - G1| + |G6 - G11|)$.

$$\triangledown B^W_{B3} = k_1(|B3 - B2|) \tag{9}$$

$$\triangledown R^W_{B3} = k_2(|R4 - R3| + |R8 - R7|) \tag{10}$$

$\triangledown^N_{B3}$, $\triangledown^E_{B3}$ and $\triangledown^S_{B3}$ are found in a similar manner. Once all the gradients are established, the weights are determined as:

$$w^k_{B3} = \frac{1}{\triangledown^k_{B3}} \tag{11}$$

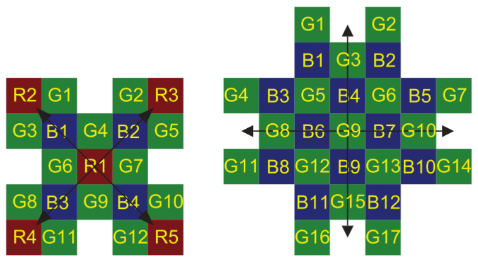
where $k = \{N, W, E, S\}$. Finally, the value of the green content in pixel B3 is given as:

$$G_{B3} = \frac{\sum_{k=\{N,W,E,S\}}[w^k_{B3}G^k_{B3}]}{\sum_{k=\{N,W,E,S\}} w^k_{B3}} \tag{12}$$

When establishing the green content in a red pixel point, the above relations from equation 6 to 12 should hold. Only the positions of the red and blue pixels are interchanged.

D. Interpolation of Missing Components in the Red/Blue Planes

When looking for missing components in these planes, two possibilities arise. The component resides in the opposing plane, that is, blue in a red pixel point or red in a blue pixel point. The second is that it is in a green pixel point. Both possibilities are illustrated in Figure 5.



(a) The Opposing Red Plane (b) The Green Plane

Fig. 5. Establishing the missing blue content in different color pixel points

1) Opposing Plane Interpolation: Let us consider the case of finding the missing content in an opposing plane. Using the example of Figure 5(a), assume the blue content in the red pixel R1 needs to be found. As the green plane has already been fully populated, a difference based interpolation solution similar to that seen in [15] is used. The initial difference based estimates taken in the NW, NE, SE and SW directions are then found as:

$$\begin{aligned} \beta^{NW}_{BG} &= B1 - G_{B1}, \beta^{NE}_{BG} = B2 - G_{B2} \\ \beta^{SE}_{BG} &= B4 - G_{B4}, \beta^{SW}_{BG} = B3 - G_{B3} \end{aligned} \tag{13}$$

Next the gradients are established and in each direction, there will be three type of contributors. The green pixels that lie exactly in the desired direction, the green pixels that are outliers of the desired direction and third the blue pixels that lie in the desired direction. They will follow that order of precedence:

$$\phi_{dir} = \sum G_{dir} + \sum G_{out} + \sum B_{dir} \tag{14}$$

To provide appropriate weighting to implicitly put the above priority into our weights, k_1 and k_2 are used. This can be envisioned intuitively as if the three contributors themselves are different pseudo-planes in this difference based relation of equation 13. For the North West direction, from equation 14, $\sum G^{NW}_{dir} = |G_{R1} - G_{B1}| + |G_{B1} - G_{R2}|$,

$\sum G_{out}^{NW} = |G4 - G1| + |G6 - G3|$ and $\sum B_{dir}^{NW} = |B4 - B1|$. The same technique is used to establish ϕ_{NE}, ϕ_{SE} and ϕ_{SW}.

The weights are then determined in a manner similar to equation 11:

$$\Phi_{k=\{NW,NE,SE,SW\}} = \frac{1}{\phi_k} \tag{15}$$

The value of the blue content in the red pixel point is then determined as follows:

$$B_{R1} = G_{R1} + \frac{\sum_{k=\{NW,NE,SE,SW\}}[\Phi_k \beta_{BG}^k]}{\sum_{k=\{NW,NE,SE,SW\}} \Phi_k} \tag{16}$$

2) Green Plane Interpolation: Let us consider interpolating the blue content G9 in Figure 5(b). The blue plane is now quincunx in nature. This allows the use of closer proximity pixels for the interpolation process. The initial estimates are determined as in equation 6. An example is given for the North direction in equation 17,

$$B_{G9}^N = B4 + k1|G9 - G3| \tag{17}$$

It should be noted that only k_1 is used in equation 17. This is because only two planes are under consideration. The gradients here are obtained in a similar manner as when the green plane was interpolated.

$$\triangledown = \triangledown B + \triangledown G' \tag{18}$$

where $\triangledown G' = k_1 \times \triangledown G$.

For the North direction example, the blue and green gradient terms are established by:

$$\triangledown B_{G9}^N = |B6 - B1| + |B7 - B2| + |B4 - B1| + |B4 - B2| \tag{19}$$

$$\triangledown G'^N_{G9} = k_1(|G5 - G1| + |G6 - G2| + |G9 - G3|) \tag{20}$$

The above two equation combined result in the directional form of equation 18

$$\triangledown_{G9}^N = \triangledown B_{G9}^N + \triangledown G'^N_{G9} \tag{21}$$

The other gradients $\triangledown_{G9}^W$,$\triangledown_{G9}^E$ and $\triangledown_{G9}^S$ are found. The weights w_{G9}^N,w_{G9}^E w_{G9}^S and w_{G9}^W are obtained and finally:

$$B_{G9} = \frac{\sum_{k=\{N,W,E,S\}}[w_{G9}^k B_{G9}^k]}{\sum_{k=\{N,W,E,S\}} w_{G9}^k} \tag{22}$$

The treatment presented here is for the missing component in the blue plane. The red plane situation is treated in the exact same manner.

III. Experimental Results

A. Choice of Image Set

The Kodak image set [16] was used. This is because most of the comparison techniques used to gauge the proposed method used that set.

B. Evaluation Measures Used

Three objective measures were used. These are the color peak signal-to-noise ratio (CPSNR) and two of the International Commission on Illumination (CIE) measures: CIELAB ($\triangle E_{ab}^*$) and CIEDE2000 ($\triangle E_{00}$). CPSNR is defined in the RGB color space and is an extension of conventional PSNR. It is an absolute measure that does not take into consideration the human visual system that follows a hue, saturation and lightness (HSL) based color space [1]. The larger the value of CPSNR the better the reconstructed image is.

$$CPSNR = 10log_{10}(\frac{255^2}{CMSE}) \tag{23}$$

$$CMSE = \frac{\sum_1^r \sum_1^c \sum_{k=R,G,B}(A_{i,j,k} - \tilde{A}_{i,j,k})}{3rc} \tag{24}$$

The second set of metrics are defined in a HSL-based color space by CIE called L*a*b* [17]. L* is luminosity information and a* and b* provide chromacity information. CIELAB was the initial difference equation. CIEDE2000 offers some improvements to measure differences in the L*a*b* space [18], [19]. The CIELAB difference equation between two pixel points is given by equation 25. The CIEDE2000 color difference formula used between two points is modestly given in equation 26. However a full representation is provided in [19].

$$\triangle E_{ab}^{*12} = \left[(L_1^* - L_2^*)^2 + (a_1^* - a_2^*)^2 + (b_1^* - b_2^*)^2\right]^{1/2} \tag{25}$$

$$\triangle E_{00}^{12} = \triangle E_{00}(L_1^*, a_1^*, b_1^*, L_2^*, a_2^*, b_2^*) \tag{26}$$

To apply equations 25 and 26, all pixel point differences must be summed over the entire image region. The smaller the value of the CIELAB and CIEDE2000 differences, the closer the reconstructed image is to the original.

C. Performance Results

The performance of the proposed algorithm was compared with nine other interpolation based demosaicking techniques. These are BI, CDBI and CRBI [7]; EDI [9]; HA [10]; Chang [11]; MHC [13]; the Wang algorithm [14] and MDWI [15]. To validate the proposed algorithm, the authors conducted simulations using MATLAB R2013a on an Intel(R) Core(TM)2 Duo CPU E7500 @2.93 GHz processor. It should be noted that no post-processing was implemented in any of the algorithms. This was because the authors wanted to determine the performance prior to post-processing. Also no formal algorithm complexity analysis was performed. Complexity tends to increase with the number of descriptors. In general BI, CDBI and CRBI have a low algorithmic complexity. EDI, HA, Chang and MHC are moderate while Wang and MDWI have the highest complexity of the chosen algorithms. The proposed algorithm is between a moderate and a high complexity.

First, the complete Kodak image set was applied to each of the demosaicking algorithms in turn. At each stage, the measures of CPSNR, CIELAB and CIEDE2000 were noted. An arithmetic mean (denoted M_A) of the set was done for each algorithm. All the results were then tabulated in Tables I to III with the proposed algorithm results in the Prop. column.

TABLE I. CPSNR RESULTS FOR DIFFERENT DEMOSAICKING TECHNIQUES ON THE KODAK IMAGE SET (IN DB)

Image	BI	CDBI	CRBI	Chang	EDI	HA	MHC	Wang	MDWI	Prop.
01	26.37	29.15	28.94	24.30	29.57	32.61	32.04	32.04	32.87	33.78
02	33.48	36.06	33.72	15.95	35.95	38.34	38.05	38.12	38.57	39.17
03	34.68	36.95	36.43	21.17	37.13	39.94	39.39	38.38	39.98	40.05
04	33.88	36.41	33.60	21.18	35.97	38.83	39.15	37.39	38.75	38.88
05	26.82	29.58	28.97	25.02	29.95	32.63	33.22	31.54	34.03	33.22
06	28.03	30.55	30.43	25.61	30.49	33.90	33.28	33.16	34.12	34.71
07	33.63	36.39	35.89	24.55	37.03	39.92	39.36	39.13	40.14	39.60
08	23.82	26.69	26.61	24.83	28.26	28.83	29.07	30.63	31.61	31.85
09	32.64	35.31	35.18	27.80	36.21	38.57	38.09	38.79	39.30	39.34
10	32.63	35.21	35.02	29.37	35.70	38.10	38.61	36.24	39.15	39.11
11	29.41	32.00	31.64	27.08	32.21	34.86	34.79	34.65	35.66	36.28
12	33.57	36.13	35.96	26.63	36.55	39.80	38.78	39.32	39.69	40.38
13	24.05	26.47	26.36	24.86	25.84	28.60	29.49	27.73	29.27	30.28
14	29.53	32.05	31.39	21.04	32.05	34.40	34.27	33.59	35.19	34.68
15	32.49	35.06	33.24	20.72	35.00	37.12	37.39	36.44	37.66	37.81
16	31.51	33.99	33.88	29.53	34.04	37.93	36.62	36.89	37.39	38.33
17	32.27	34.81	34.60	29.72	34.71	36.96	37.69	36.50	37.77	38.06
18	28.14	30.68	30.36	24.03	30.07	32.61	33.48	31.88	33.22	33.59
19	28.24	31.14	30.89	24.48	33.19	33.86	33.66	35.68	36.06	36.27
20	31.83	34.29	34.08	11.85	34.84	37.20	36.99	36.54	37.03	38.31
21	28.72	31.33	31.15	25.25	31.22	34.47	34.14	33.55	34.80	35.24
22	30.70	33.21	32.96	23.99	33.17	35.06	35.32	34.77	35.92	35.90
23	35.27	38.24	37.39	19.35	38.23	40.29	40.95	39.67	40.91	39.84
24	26.87	29.33	29.21	16.98	28.76	30.73	31.68	29.99	31.44	32.00
M_A	30.36	32.96	32.41	23.55	33.17	35.65	35.65	35.11	36.27	**36.53**

TABLE II. CIELAB $\triangle E^*_{ab}$ RESULTS FOR DIFFERENT DEMOSAICKING TECHNIQUES ON THE KODAK IMAGE SET

Image	BI	CDBI	CRBI	Chang	EDI	HA	MHC	Wang	MDWI	Prop.
01	9.159	7.173	7.390	10.396	6.305	4.597	5.054	4.910	4.312	4.110
02	3.783	3.014	4.222	26.908	2.735	2.268	2.419	2.451	2.147	2.203
03	3.050	2.537	2.732	9.664	2.189	1.688	1.791	2.014	1.665	1.660
04	3.871	3.079	4.068	13.739	2.956	2.348	2.301	2.721	2.209	2.223
05	8.276	6.576	7.040	8.833	5.790	4.125	4.206	4.700	3.806	3.916
06	6.836	5.501	5.557	7.634	4.866	3.339	3.875	3.736	3.418	3.241
07	3.421	2.806	2.987	7.876	2.282	1.821	1.996	2.061	1.739	1.900
08	11.120	8.699	8.818	9.490	6.346	5.708	6.184	4.998	4.717	4.600
09	3.802	3.131	3.245	4.395	2.640	2.139	2.242	2.267	2.030	2.015
10	3.749	3.101	3.219	4.414	2.660	2.083	2.161	2.356	1.995	1.984
11	5.713	4.570	4.790	6.591	4.054	2.981	3.293	3.237	2.897	2.745
12	3.477	2.843	2.961	6.696	2.439	1.861	2.062	2.068	1.837	1.803
13	11.693	9.327	9.422	10.560	9.678	6.740	6.587	7.681	6.575	5.731
14	6.254	4.968	5.347	9.666	4.592	3.338	3.517	3.830	3.202	3.234
15	4.000	3.236	3.965	9.814	2.935	2.415	2.405	2.681	2.215	2.282
16	4.837	3.923	3.986	5.334	3.452	2.392	2.826	2.736	2.533	2.383
17	4.009	3.268	3.418	4.699	2.962	2.270	2.334	2.542	2.209	2.172
18	6.761	5.430	5.627	8.985	5.512	4.039	3.823	4.544	3.789	3.683
19	6.131	4.891	5.066	8.719	3.852	3.393	3.525	3.246	2.897	2.816
20	3.779	3.120	3.211	20.594	2.715	2.170	2.271	2.368	2.101	1.973
21	6.053	4.868	4.995	8.486	4.582	3.328	3.489	3.759	3.240	3.061
22	5.033	4.133	4.270	9.049	4.010	3.111	3.045	3.498	2.935	2.900
23	2.582	2.176	2.415	11.135	1.976	1.705	1.621	1.868	1.597	1.642
24	6.824	5.445	5.558	11.140	5.073	3.801	3.783	4.253	3.564	3.418
M_A	5.592	4.492	4.763	9.784	4.025	3.069	3.200	3.355	2.901	**2.821**

TABLE III. CIEDE2000 $\triangle E_{00}$ RESULTS FOR DIFFERENT DEMOSAICKING TECHNIQUES ON THE KODAK IMAGE SET

Image	BI	CDBI	CRBI	Chang	EDI	HA	MHC	Wang	MDWI	Prop.
01	3.577	2.820	2.858	3.180	2.441	1.728	1.795	1.846	1.686	1.600
02	1.579	1.260	1.728	8.127	1.130	0.923	0.943	0.993	0.885	0.896
03	1.175	0.981	1.029	2.391	0.831	0.630	0.643	0.745	0.628	0.628
04	1.638	1.301	1.755	4.219	1.246	0.969	0.900	1.121	0.921	0.915
05	3.801	3.044	3.217	2.870	2.666	1.797	1.725	2.083	1.729	1.699
06	2.522	2.024	2.002	1.959	1.776	1.189	1.294	1.340	1.254	1.188
07	1.390	1.159	1.188	1.971	0.912	0.696	0.755	0.787	0.675	0.734
08	4.335	3.402	3.422	2.946	2.480	2.152	2.216	1.918	1.860	1.785
09	1.424	1.171	1.187	1.229	0.954	0.764	0.771	0.796	0.727	0.727
10	1.438	1.193	1.218	1.232	1.002	0.764	0.765	0.861	0.740	0.739
11	2.399	1.930	1.975	1.993	1.705	1.217	1.264	1.324	1.210	1.121
12	1.156	0.938	0.956	1.499	0.793	0.596	0.627	0.664	0.598	0.591
13	4.724	3.791	3.769	3.154	3.926	2.645	2.424	3.069	2.704	2.312
14	2.690	2.158	2.278	2.635	1.980	1.399	1.380	1.608	1.370	1.346
15	1.707	1.382	1.722	3.118	1.265	1.009	0.938	1.126	0.945	0.947
16	2.106	1.726	1.715	1.612	1.488	1.000	1.128	1.141	1.089	1.001
17	1.740	1.435	1.464	1.453	1.294	0.960	0.934	1.072	0.950	0.909
18	3.049	2.493	2.523	2.697	2.522	1.767	1.588	2.013	1.710	1.621
19	2.371	1.914	1.936	2.309	1.586	1.307	1.230	1.303	1.166	1.095
20	1.520	1.262	1.264	5.989	1.100	0.831	0.818	0.919	0.819	0.750
21	2.482	2.007	2.019	2.325	1.879	1.315	1.304	1.496	1.314	1.221
22	1.928	1.595	1.616	2.298	1.541	1.144	1.076	1.302	1.102	1.080
23	1.007	0.850	0.935	2.738	0.748	0.636	0.585	0.694	0.593	0.613
24	2.615	2.102	2.116	3.304	1.919	1.394	1.349	1.572	1.335	1.289
M_A	2.266	1.831	1.912	2.802	1.633	1.201	1.185	1.325	1.167	**1.117**

Table I highlights the CPSNR performance. Table II and III show the CIELAB and CIEDE2000 values providing a measure for human vision. A marked improvement over established methods is noted as the proposed algorithm had the highest mean CPSNR and the lowest mean CIELAB and CIEDE2000 values. This is an objective indication of image acuity.

IV. CONCLUSION

A new interpolation based demosaicking technique was proposed with two new contributions. To provide objectivity in analysis, three performance metrics: CPSNR, CIELAB and CIEDE2000 were used. Through mean value information, the proposed technique showed a marked improvement over the selected interpolation-based demosaicking techniques. Furthermore, the two contributions offered by the proposed technique are deemed as transferable meaning established techniques can also incorporate them to yield better results.

REFERENCES

[1] R. Ramanath, W. E. Snyder, Y. Yoo, and M. S. Drew, “Color image processing pipeline,” *IEEE Signal Process. Mag.*, vol. 22, pp. 34–43, Jan. 2005.

[2] M. Vrhel, E. Saber, and H. J. Trussel, “Color image generation and display technologies,” *IEEE Signal Process. Mag.*, vol. 22, pp. 23–33, Jan. 2005.

[3] X. Li, B. K. Gunturk, and L. Zhang, “Image demosaicking: A systematic survey,” in *Proc. SPIE 6822, Visual Communications and Image Processing 2008, SPIE-IS&T Electronic Imaging*, San Jose, CA, Jan. 2008.

[4] B. E. Bayer, “Color imaging array,” U.S. Patent 3 971 065, Jul. 20, 1976.

[5] D. Alleysson, S. Süsstrunk, and J. Hérault, “Linear demosaicing inspired by the human visual system,” *IEEE Trans. Image Process.*, vol. 14, no. 4, pp. 1–12, Apr. 2005.

[6] K. Hirakawa and P. J. Wolfe, “Spatio-Spectral color filter array design for optimal image recovery,” *IEEE Trans. Image Process.*, vol. 17, no. 10, pp. 1876–1890, Oct. 2008.

[7] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, “Demosaicking: Color filter array interpolation,” *IEEE Signal Process. Mag.*, vol. 22, pp. 44–54, Jan. 2005.

[8] W. T. Freeman, “Median filter for reconstructing missing color samples,” U.S. Patent 4 724 395, Feb. 9, 1988.

[9] C. A. Laroche and M. A. Prescott, “Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients,” U.S. Patent 5 373 322, Dec. 13, 1994.

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

[11] E. Chang, S. Cheung, and D. Pan, “Color filter array recovery using a threshold-based variable number of gradients,” in *Proc. SPIE 3650, IS&T/SPIE Conference on Sensors, Cameras and Applications for Digital Photography*, San Jose, CA, Jan. 1999.

[12] W. Lu and Y.-P. Tan, “Color filter array demosaicking: New method and performance measures,” *IEEE Trans. Image Process.*, vol. 12, no. 10, pp. 1194–1210, Oct. 2003.

[13] P. Getreuer, “Malvar-He-Cutler linear image demosaicking,” *Image Processing On Line (IPOL)*, vol. 1, Aug. 2011.

[14] W. Jin, “Improved color interpolation method based on bayer image,” in *Proc. SPIE 8420, 6th International Symposium on Advanced Optical Manufacturing and Testing Technologies*, 2012.

[15] X. Chen, G. Jeon, J. Jeong, and L. He, “Multi-directional weighted interpolation and refinement method for bayer pattern cfa demosaicking,” *IEEE Trans. Circuits Syst. Video Technol.*, submitted for publication in 2014.

[16] (2014). [Online]. Available: http://r0k.us/graphics/kodak/

[17] *ISO 11664-4:2008(E)/CIE S 014-4/E:2007 Joint ISO/CIE Standard: Colorimetry Part 4: CIE 1976 L*a*b* Colour Space*, ISO/CIE Std., 2008.

[18] *ISO/CIE 11664-6:2014(E): Colorimetry - Part 6: CIEDE2000 Colour-Difference Formula*, ISO/CIE Std., 2008.

[19] G. Sharma, W. Wu, and E. N. Dalal, “The CIEDE2000 Color-Difference Formula: Implementation notes, supplementary test data, and mathematical observations,” *Color Research and Application*, vol. 30, no. 1, pp. 21–30, Feb. 2005.

G.2 Second Publication - IEEE ICTRC 2015

2015 International Conference on Information and Communication Technology Research (ICTRC2015)

A Multi-variate Weighted Interpolation Technique with Local Polling for Bayer CFA Demosaicking

Kinyua Wachira and Elijah Mwangi
School of Engineering,
University of Nairobi,
P.O. Box 30197 Nairobi 00100,
Kenya
Email: kinyua.wachira@students.uonbi.ac.ke

***Abstract*—Gradient-based Spatial demosaicking techniques have gained prominence in literature for their superior reconstruction capabilities. This paper presents a novel algorithm in this class with several key contributions. It introduces variables operating at various lattice levels of the Color Filter Array (CFA) data. It also employs a Square-On-Point (SoP) neighborhood, a corrective term and localized polling to reduce reconstruction errors thus improving image perception. The proposed algorithm is compared to other contemporary methods and an appreciable improvement in performance has been noted through Matlab simulation. To provide a robust analysis, two performance metrics (CPSNR and SSIM) are used over two distinct image sets.**

Keywords*—CFA, Demosaicking, Difference terms, Inter- and Intra-Lattice factors, Residual terms, Square-on-Point.

I. Introduction

Pictures are an essential medium for relaying information. For this reason, the digital still camera (DSC) and the color photograph have become an all pervasive in society. Human vision, by nature, is trichromatic [1]. As such, a color image is composed of three base colors. Common image sensors, however, are monochrome devices that cannot capture color information. To ensure a cost effective and robust design, most DSCs employ a spectrally selective arrangement termed a color filter array/mosaic (CFA or CFM) to discriminate colors at the pixel level prior to passing the resultant light to a single monochrome sensor. Each pixel then contains light intensity information from only one prescribed color wavelength. Different CFA regimes exist; the most commonly documented, shown in Figure 1, is the Bayer CFA patented in 1976 [2]. It has red, green and blue as its base colors.

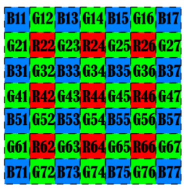

B11 G12 B13 G14 B15 G16 B17
G21 R22 G23 R24 G25 R26 G27
B31 G32 B33 G34 B35 G36 B37
G41 R42 G43 R44 G45 R46 G47
B51 G52 B53 G54 B55 G56 B57
G61 R62 G63 R64 G65 R66 G67
B71 G72 B73 G74 B75 G76 B77

Fig. 1. A 7-by-7 window of the Bayer CFA Pattern (US Patent 3971605)

The process of reconstructing a full three color per pixel image from the sub-sampled CFA raw data is called demosaicking. Several recent surveys [3]–[5] broadly classify demosaicking algorithms into either spatially or spectrally motivated methods. This paper limits itself to the spatial based variants as these schemes deal with pixel data directly. Their complexity is also low enough to allow for real time computation. Edge Directed Interpolation (EDI), Hamilton-Adams method (HA) [5]; and the Malvar-He-Cutler algorithm [6] are adaptive techniques that rely on local neighborhood information for interpolation. Modern spatial algorithms extend the adaptability of edge methods by employing gradients in their interpolation process [7]–[12]. Each gradient is made of a number of difference-term pairs. These gradients are weighted so that those falling in a direction sympathetic to the desired interpolation movement are of a larger weight. Some recent variants developed by Kiku et al. [9], [13] use residual-terms rather than difference-term pairs.

The proposed algorithm extends the modern difference-pair gradient techniques by providing several novel contributions in the form of a Square-On-Point (SoP) neighborhood and lattice variables. It also uses a corrective constant and localized polling to improve performance. This paper is divided as follows: Section II highlights the contributions while section III details the proposed algorithm. Section IV explains the data sets and performance metrics citing the motivation behind their selection. It also analyses the results observed from experimental testing. Finally Section V briefly presents the conclusions drawn from the overall process.

II. Contributions Offered

A. *Inter-Lattice Factors, ω_1 and ω_2*

A Bayer CFA is a lattice of three different color planes [2] overlaid on one another. Figure 2 shows this lattice in a dissociated form.

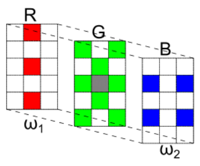


Fig. 2. Bayer CFA section illustrating the inter-lattice weighting concept

Consider the process of determining the green content in the gray pixel point in Figure 2. Reasoning from a weights point of view, neighborhood pixels from different planes should be factored differently. The green plane should have the most influence. This is the plane with the pixel whose color is being found. The red plane follows in importance as it shares the same pixel location as the desired point. Finally, the blue plane neither contains the required pixel point nor the color and should contribute the least. The weight factors in this argument would then follow the trend in equation 1. From empirical determination, it was found that $\omega_1 = 0.8$ and $\omega_2 = 0.7$ gave the best results when green is weighted by 1.

$$\omega_2 < \omega_1 < 1 \tag{1}$$

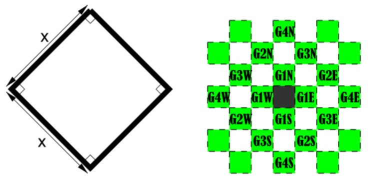
B. Intra-Lattice Factors, δ_1 and δ_2

Neighborhood pixels are grouped in difference pairs whose sum forms the directional gradient. It stands to reason that some difference pairs have an orientation that is more sympathetic to the interpolation direction and should be given more prominence. This implies that members of the internal lattice should be variably weighted to provide a more accurate gradient.

Two factors are introduced. An in-line factor, denoted δ_1, weighs the prominent difference pairs. An outlier factor, denoted δ_2, is used with the non prominent pairs. In the proposed method the authors use $\delta_1 = 2$ and $\delta_2 = 1$ for the red or blue planes. In the green plane: $\delta_1 = 3$ and $\delta_2 = 1$. δ_1 is higher in the green plane since it is sampled at a higher rate. As such more confidence can be placed in the in-line emphasis factor.

C. Square-on-Point (SoP) Usage in the Green Plane

A Square-On-Point (SoP) arrangement in a 7-by-7 CFA shown in Figure 3 region is used as the green pixel neighborhood. It can be decomposed into 4 distinct sub-neighborhoods that do not overlap. Each sub-neighborhood is composed of 4 pixels in a specific cardinal direction. The West sub-neighborhood, for instance, is made of G1W, G2W, G3W and G4W.



(a) Diagram (b) Green pixel neighborhood

Fig. 3. Square-on-Point Concept

D. A Corrective Constant, γ

In regions of homogeneity, the sum of difference pairs may result in a zero gradient. This will lead to an erroneous estimation of the missing color. Some algorithms [7], [8] add a constant, usually 1, to provide correction. Empirically, the authors found that this corrective constant is roughly equal to the number of interpolations directions chosen. In the algorithm presented, a corrective constant of $\gamma = 4$ was used.

III. Proposed Algorithm

The general gradient equation is of the form shown in equation 2; having contributions from all three color planes.

$$\psi = \psi_G + \psi_R + \psi_B + \gamma \tag{2}$$

A. Determination of Missing Components in the Green Plane

Consider trying to establish the green color content at the pixel marked R44 in the Bayer arrangement of Figure 1. The four cardinal direction based estimates can be determined as:

$$\begin{aligned}
\tilde{G}^N_{R44} &= G34 + [(\omega_1\omega_2)(R44 - R24)] \\
\tilde{G}^W_{R44} &= G43 + [(\omega_1\omega_2)(R44 - R42)] \\
\tilde{G}^S_{R44} &= G54 + [(\omega_1\omega_2)(R44 - R64)] \\
\tilde{G}^E_{R44} &= G45 + [(\omega_1\omega_2)(R44 - R46)]
\end{aligned} \tag{3}$$

The gradients in each cardinal direction will follow of the form of equation 2. In the East direction:

$$\psi^E_{R44} = \psi^E_{G,R44} + \psi^E_{R,R44} + \psi^E_{B,R44} + \gamma \tag{4a}$$

$$\begin{aligned}
\psi^E_{G,R44} &= \delta_1|G45 - G47| + \delta_2(|G45 - G36| + |G45 - G56|) \\
\psi^E_{R,R44} &= \omega_1(|R44 - R46| + |R24 - R26| + |R64 - R66|) \\
\psi^E_{B,R44} &= \omega_2(|B35 - B37| + |B55 - B57|)
\end{aligned} \tag{4b}$$

Gradients ψ^N_{R44}, ψ^S_{R44} and ψ^W_{R44} are generated in a similar manner. The inverse of the gradients in each direction form the weights to be used on the estimates. As such:

$$\Psi^n_{R44} = \frac{1}{\psi^n_{R44}} \tag{5}$$

where n corresponds to each of the cardinal directions. As a refinement step, local polling maps such as those developed by Chen et al. [7] are used to find the final value. Equations 6a or 6b indicate a predominantly binary direction sense. If no clear interpolation line is sensed, equation 6c is employed. A similar treatment is used to determine green color content in blue pixel locations.

$$\hat{G}_{R44} = \frac{\sum_{n\in\{E,W\}}[\Psi^n_{R44}\tilde{G}^n_{R44}]}{\sum_{n\in\{E,W\}}\Psi^n_{R44}} \tag{6a}$$

$$\hat{G}_{R44} = \frac{\sum_{n\in\{N,S\}}[\Psi^n_{R44}\tilde{G}^n_{R44}]}{\sum_{n\in\{N,S\}}\Psi^n_{R44}} \tag{6b}$$

$$\hat{G}_{R44} = \frac{\sum_{n\in\{N,E,W,S\}}[\Psi^n_{R44}\tilde{G}^n_{R44}]}{\sum_{n\in\{N,E,W,S\}}\Psi^n_{R44}} \tag{6c}$$

B. Determination of Missing Components in the Blue and Red Plane

Pixel color determination falls into two categories. It ether involves finding color content in an opposing plane or determining color data from the green plane.

1) Opposing Plane Considerations: Consider determining the blue color in the pixel R44 from Figure 1. As the green content in each pixel has been determined a priori, a difference based solution similar to Chen et al. [8] is used. Initial estimates are taken in the ordinal directions:

$$\begin{aligned} \tilde{\mu}_{R44}^{NW} &= B33 - \hat{G}_{B33}, \tilde{\mu}_{R44}^{NE} = B35 - \hat{G}_{B35} \\ \tilde{\mu}_{R44}^{SE} &= B55 - \hat{G}_{B55}, \tilde{\mu}_{R44}^{SW} = B53 - \hat{G}_{B53} \end{aligned} \tag{7}$$

The gradients are found using a variant of equation 2

$$\psi^k = \psi_{G,in}^k + \psi_{G,out}^k + \psi_B^k + \gamma \tag{8}$$

where k denotes ordinal directions. Taking the NW direction, the components of equation 8 are:

$$\begin{aligned} \psi_{G,in}^{NW} &= \delta_1(|\bar{G}_{R44} - \bar{G}_{B33}| + |\bar{G}_{B33} - \bar{G}_{R22}|) \\ \psi_{G,out}^{NW} &= \delta_2(|G43 - G32| + |G34 - G23|) \\ \psi_B^{NW} &= \omega_2(|B33 - B55|) \end{aligned} \tag{9}$$

After ψ_{R44}^{NW}, ψ_{R44}^{NE}, ψ_{R44}^{SE} and ψ_{R44}^{SW} are established, the weights are found using equation 5, replacing the cardinal directions with ordinal ones. For this consideration, the blue content in the red pixel location is given by:

$$\hat{B}_{R44} = \hat{G}_{R44} + \frac{\sum_{k \in \{NW,NE,SW,SE\}} [\Psi_{R44}^k \tilde{\mu}_{R44}^k]}{\sum_{k \in \{NW,NE,SW,SE\}} \Psi_{R44}^k} \tag{10}$$

2) In-Green Pixel Consideration: Consider looking for missing red content in pixel G45. At this stage both the red and green planes are quincuncial and are of the same size. Making an initial estimate of the North direction:

$$\tilde{R}_{R45}^N = \hat{R}_{B35} + [(\omega_1 \omega_2)(G45 - G25)] \tag{11}$$

After finding $\tilde{R}_{R45}^W$, $\tilde{R}_{R45}^S$ and $\tilde{R}_{R45}^E$, the same mechanic of establishing the gradients, weights and the final value is followed with minor modification. Taking the North direction for the G45 example:

$$\begin{aligned} \psi_{G45}^N &= \psi_{R,G45}^N + \psi_{G,G45}^N + \gamma \\ \psi_{R,G45}^N &= \delta_1(|\hat{R}_{B35} - \hat{R}_{B15}| + |\hat{R}_{B35} - \hat{R}_{B55}|) \\ &\quad + \delta_2(|\hat{R}_{B35} - R24| + |\hat{R}_{B35} - R26|) \\ \psi_{G,G45}^N &= (|G45 - G25| + |G34 - G14| + |G36 - G16|) \end{aligned} \tag{12}$$

After ψ_{G45}^S ψ_{G45}^W and ψ_{G45}^E are determined, equation 5 is used to find the weights. The final value is given by:

$$\hat{R}_{G45} = \frac{\sum_{n \in \{N,E,W,S\}} [\Psi_{G45}^n \tilde{R}_{G45}^n]}{\sum_{n \in \{N,E,W,S\}} \Psi_{G45}^n} \tag{13}$$

The same treatment is applied to determine blue content as well.

IV. Simulation Results and Analysis

A. Choice of Image Sets and Metrics

The proposed method was tested using 12 randomly selected images from each of the commonly documented Kodak [14] and McMaster [15] image sets shown in Figure 4.

(a) Kodak Images (1-12)

(b) McMaster Images (1-12)

Fig. 4. Image Sets Used

Two objective measures were used: the Color Peak Signal-to-Noise Ratio (CPSNR) is as it is a common metric. However as the human visual system is more attuned to objects than absolute pixel values [16], the Structure SIMilarity Index (SSIM) [17] provides a measure in line with subjective comparisons.

B. Testing Methodology and Performance

The proposed algorithm was compared with five recent state-of-the-art algorithms: the Wang method [11], ESFB [12], MDWI [8], MGBI [10] are difference-term gradient algorithms. MLRI [13] is a residual-term gradient algorithm. Simulations were conducted using MATLAB R2013a on an Intel(R) Core(TM)2 Duo CPU E7500 @ 2.93GHz processor. Firstly, the 12 images of the Kodak set were exposed to each of the test algorithms in turn. At each point, the values of CPSNR and SSIM were noted and tabulated in Tables I and II. In addition, two ranking systems were used. The DT Rank indicates the best difference-term based method. The best all around technique is shown by the overall rank. The McMaster images are treated in the same manner and their results are recorded in Tables III and IV. The average CPSNR and SSIM results are then combined for all the test images to gauge the overall robustness of the methods chosen. This is shown in Table V with bold font indicating best performance.

TABLE I. CPSNR Evaluation of Several Demosaicking Techniques on the Kodak Image Set (in dB)

Image	Wang	ESFB	MDWI	MGBI	MLRI	Proposed	DT Rank	Overall Rank
1	41.83	42.06	41.88	42.10	41.31	43.03	1	1
2	42.71	44.38	42.97	44.49	42.95	44.78	1	1
3	41.23	42.54	41.80	42.55	41.97	42.88	1	1
4	43.24	44.14	43.25	43.98	42.74	44.31	1	1
5	42.59	43.81	42.57	43.75	42.38	44.12	1	1
6	39.68	40.59	39.59	42.36	39.42	40.84	1	1
7	34.56	37.08	34.94	38.38	33.20	35.94	1	1
8	40.82	41.78	41.32	40.48	39.48	42.07	1	1
9	39.73	40.07	39.75	42.72	40.07	41.19	1	1
10	41.15	41.74	41.28	34.20	40.71	42.90	1	1
11	39.05	40.38	39.54	41.14	38.65	40.66	1	1
12	44.05	44.11	43.50	43.17	43.75	44.92	1	1
Avg.	40.89	41.89	41.03	41.61	40.55	**42.30**	1	1

Tables I and III highlight the CPSNR performance over the two image sets. In the Kodak set, it is observed that

TABLE II. SSIM EVALUATION OF SEVERAL DEMOSAICKING TECHNIQUES ON THE KODAK IMAGE SET

Image	Wang	ESFB	MDWI	MGBI	MLRI	Proposed	DT Rank	Overall Rank
1	0.978	0.975	0.970	0.971	0.991	0.977	2	3
2	0.980	0.984	0.980	0.983	0.995	0.986	1	2
3	0.981	0.984	0.979	0.981	0.994	0.985	1	2
4	0.988	0.986	0.982	0.983	0.997	0.986	2	3
5	0.980	0.981	0.975	0.975	0.994	0.981	2	3
6	0.978	0.980	0.975	0.981	0.994	0.982	1	2
7	0.956	0.977	0.962	0.975	0.986	0.974	3	4
8	0.983	0.986	0.985	0.970	0.991	0.987	1	2
9	0.976	0.979	0.973	0.980	0.992	0.980	1	2
10	0.972	0.954	0.967	0.851	0.991	0.973	1	2
11	0.974	0.977	0.973	0.975	0.990	0.979	1	2
12	0.985	0.984	0.981	0.973	0.995	0.985	2	3
Avg.	0.978	0.979	0.975	0.966	**0.993**	0.981	1	2

TABLE III. CPSNR EVALUATION OF SEVERAL DEMOSAICKING TECHNIQUES ON THE MCMASTER IMAGE SET (IN DB)

Image	Wang	ESFB	MDWI	MGBI	MLRI	Proposed	DT Rank	Overall Rank
1	37.90	37.31	38.40	37.28	35.07	38.18	2	2
2	36.24	36.74	37.26	37.10	33.95	37.65	1	1
3	38.71	38.36	40.65	38.64	37.77	40.53	2	2
4	39.37	37.50	40.67	37.50	38.25	40.06	2	2
5	39.61	38.61	40.59	38.49	36.48	40.25	2	2
6	40.54	39.33	41.03	39.22	38.69	40.86	3	3
7	41.37	40.57	42.11	40.24	39.95	42.03	2	2
8	42.96	41.43	42.64	40.99	40.65	43.15	1	1
9	41.42	40.57	41.62	40.16	38.84	41.50	2	2
10	36.92	36.33	38.00	36.41	35.14	37.68	2	2
11	36.27	35.14	37.96	35.10	32.63	37.16	2	2
12	37.83	37.22	38.89	37.56	36.08	38.96	1	1
Avg.	39.09	38.26	**39.99**	38.23	36.96	39.83	2	2

TABLE IV. SSIM EVALUATION OF SEVERAL DEMOSAICKING TECHNIQUES ON THE MCMASTER IMAGE SET

Image	Wang	ESFB	MDWI	MGBI	MLRI	Proposed	DT Rank	Overall Rank
1	0.973	0.967	0.972	0.964	0.983	0.971	3	4
2	0.972	0.971	0.967	0.966	0.991	0.973	1	2
3	0.972	0.968	0.961	0.960	0.997	0.967	3	4
4	0.969	0.949	0.960	0.947	0.991	0.966	2	3
5	0.970	0.968	0.972	0.963	0.988	0.972	1	2
6	0.980	0.976	0.980	0.972	0.992	0.981	1	2
7	0.974	0.966	0.974	0.962	0.992	0.973	3	4
8	0.961	0.958	0.956	0.950	0.985	0.960	2	3
9	0.979	0.972	0.969	0.967	0.987	0.974	2	3
10	0.942	0.927	0.955	0.926	0.988	0.952	2	3
11	0.962	0.946	0.972	0.943	0.983	0.966	2	3
12	0.968	0.965	0.970	0.964	0.988	0.972	1	2
Avg.	0.968	0.961	0.967	0.957	**0.989**	0.969	1	2

TABLE V. AVERAGE CPSNR AND SSIM PERFORMANCE

method	CPSNR	SSIM
Wang	39.09	0.973
ESFB	40.08	0.970
MDWI	40.51	0.972
MGBI	39.92	0.962
MLRI	38.76	**0.991**
Proposed	**41.07**	0.975

the proposed technique exceeds the established methods. For example, from the average data in Table I the proposed method exceeds MDWI [8] by 0.41dB. In the McMaster set, it is second behind MDWI. The SSIM performance in both images sets shows the proposed algorithm is the best difference-term gradient method and the second best method overall. This is from an observation of the mean data in Tables II and IV. It should noted that each method performs differently for different image sets over different metrics. Table V gives an overall view of performance over all images for the two evaluation metrics. It shows the proposed technique has a better CPSNR performance than all the tested methods. It exceeds the second best technique by 0.56dB. It also exhibits a good average SSIM index second only to MLRI [13]. No formal complexity analysis was done. The proposed technique, from inspection, is of a higher computational complexity than MLRI. It however is comparable with the other constant difference based techniques presented.

V. CONCLUSION

A new adaptive constant difference gradient based spatial demosaicking technique has been proposed. Several novel concepts in the form of lattice relations and use of a SoP neighborhood have been advanced. Objectivity in analysis has been provided through the use of two standard image sets and two well documented performance criteria. From the mean value data, the proposed technique was found to be comparable to existing methods; even exceeding members of its sub-class. Furthermore, all the contributions offered by the proposed algorithm are viewed as flexible and transferable. They can be incorporated into existing techniques in order to provide further improvement.

REFERENCES

[1] A. Koschan and M. A. Abidi, *Digial Color Image Processing*. Hoboken, NJ: John Wiley and Sons Inc., 2008.

[2] B. E. Bayer, "Color imaging array," U.S. Patent 3 971 065, Jul. 20, 1976.

[3] D. Menon and G. Calvagno, "Color image demosaicking: An overview," *Signal Processing: Image Communication*, vol. 26, May 2011.

[4] X. Li, B. K. Gunturk, and L. Zhang, "Image demosaicking: A systematic survey," in *Proc. SPIE 6822, Visual Communications and Image Processing 2008, SPIE-IS&T Electronic Imaging*, San Jose, CA, Jan. 2008.

[5] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demosaicking: Color filter array interpolation," *IEEE Signal Process. Mag.*, vol. 22, pp. 44–54, Jan. 2005.

[6] H. S. Malvar, L. W. He, and R. Cutler, "High-quality linear interpolation for demosaicking of bayer-patterned color images," in *Proc. IEEE 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, Montreal, Quebec, Canada, May 2004.

[7] X. Chen, G. Jeon, and J. Jeong, "Voting based directional method and its application to still color image demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 2, pp. 255–262, Feb. 2014.

[8] X. Chen, G. Jeon, J. Jeong, and L. He, "Multi-directional weighted interpolation and refinement method for bayer pattern cfa demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, submitted for publication in 2014.

[9] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Residual interpolation for color image interpolation," in *2013 IEEE International Conference on Image Processing (ICIP 2013)*, Melbourne, Australia, Sep. 2013.

[10] I. Pekkucuksen and Y. Altunbasak, "Multiscale gradients-based color filter array interpolation," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 157–165, Jan. 2013.

[11] W. Jin, "Improved color interpolation method based on bayer image," in *Proc. SPIE 8420, 6th International Symposium on Advanced Optical Manufacturing and Testing Technologies*, 2012.

[12] I. Pekkucuksen and Y. Altunbasak, "Edge strength filter based color filter array interpolation," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 393–397, Jan. 2012.

[13] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Minimized-laplacian residual interpolation for color image demosaicking," in *Proc. SPIE-IS&T 9023, Digital Photography X*, Mar. 2014.

[14] (2015). [Online]. Available: http://r0k.us/graphics/kodak/

[15] (2015). [Online]. Available: http://www4.comp.polyu.edu.hk/~cslzhang/

[16] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it?" *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009.

[17] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

G.3 Third Publication – IEEE EUROCON 2015

Corrective Term Usage in the Improvement of Gradient-Based Bayer CFA Demosaicking Algorithms

Kinyua Wachira
School of Engineering,
University of Nairobi,
P.O. Box 30197 Nairobi 00100, Kenya
Email: kinyua.f.wachira@ieee.org, kinyua.wachira@students.uonbi.ac.ke

***Abstract*—This paper presents a novel scrutiny into the role of corrective terms in gradient-based demosaicking algorithms and how they can be used to improve image reconstruction. The terms analysed are the non-zero corrective term ϵ and a new inter-plane weighting term β. Two new techniques are proposed using these terms to highlight their contribution. Four performance metrics have been used (PSNR, CPSNR, SSIM and FSIM) for testing over four different image sets. Comparison of one the proposed algorithms is made with established current techniques and an improvement in image fidelity is observed over several image sets.**

***Keywords*—CFA, corrective and directional terms, demosaicking, homogeneity and near-homogeneity, inter-plane weighting, non-zero correction.**

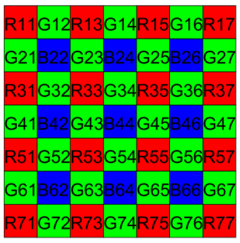
I. Introduction

Images have become a pertinent feature of modern society. Colour images are the most common form and the digital still cameras (DSC) that produce them are often found either within other devices or as standalone elements.

Due to complexities in hardware design and cost, generation of colour images is treated as a sub-sampling process. Three base colours are needed to generate a full colour image due to the trichromacy of human vision. However, at each sampling point of a conventional DSC, only one base colour is sampled. This is achieved by means of a specialised filter, called a Colour Filter Array or Mosaic (CFA/CFM), that selectively discriminates colours in certain sample points. The sensor receives a one colour per sample point lattice representing the captured scene. A software process is then used to reconstruct the full colour image from the CFA data. This process is called demosaicking.

Different manufacturers have developed various filter regimes. The most documented of these is the Bayer CFA [1] shown in Figure 1 where the base colours are red, green and blue. The green channel is sampled at twice the rate of the red and blue channels because the human eye is most sensitive to the mid-wavelengths of the visible light spectrum [2].

In a recent survey [3], demosaicking techniques are broadly classified into spectral and spatial-based methods. This paper is biased toward spatial techniques because, in general, they deal with pixel information directly and are of a low computational complexity to allow for real time processing. Most demosaicking algorithms that operate in the spatial domain use a gradient analysis [4]–[16]. Each gradient is made up of a number of directional and corrective terms corresponding to a direction of interpolation. Directional terms help in determination of the predominant interpolation direction while the corrective terms are mainly used to mitigate some undesirable visual artefacts that may be introduced. The choice of terms determines the image fidelity in reconstruction. Some methods such as the Multi-Gradient Based Interpolation (MGBI) method [12] and the Wang technique [11] only exploit the directional terms. More recent methods such as the Edge-Strength Filter Based Interpolation method (ESFB) [14] and the methods developed by Chen et al. [4], [5], [7] have a small error term for correction.


R11 G12 R13 G14 R15 G16 R17
G21 B22 G23 B24 G25 B26 G27
R31 G32 R33 G34 R35 G36 R37
G41 B42 G43 B44 G45 B46 G47
R51 G52 R53 G54 R55 G56 R57
G61 B62 G63 B64 G65 B66 G67
R71 G72 R73 G74 R75 G76 R77
Fig. 1. The Bayer CFA arrangement in a 7-by-7 window

In most algorithms, directional terms are given dominance. The corrective terms are treated as an afterthought. This paper demonstrates that a well chosen corrective term is comparable to one or more directional terms.

The remainder of this paper is divided as follows. Section II explains the elements that constitute a pure gradient-based demosaicking algorithm. Section III highlights the two corrective terms that provide the enhancement potential. Section IV explains the data and performance metrics and the rationale behind their selection. Section V highlights important results. Finally Section VI briefly presents the conclusions drawn from the presented results.

978-1-4799-8569-2/15/$31.00 ©2015 IEEE

II. A Generic Gradient-Based Demosaicking Algorithm

Effective demosaicking involves choosing a sufficient number of descriptors. Based on the relative direction from a pixel of interest, these descriptors can adequately predict the missing colour component(s) in that pixel using neighbourhood pixel statistics.

Gradient based demosaicking algorithms can generally be broken down into four stages:

1) Initial estimates of the desired pixel in various directions are generated

$$\tilde{D}_P^k = D_{P-1}^k + m_1(X_P^k - X_{P-2}^k) \tag{1}$$

where P is the pixel point, D is the desired plane, X is the opposing plane, k is the direction of interpolation and m_1 is a constant.

2) A directional gradient based on a sum of differences from existing pixel points, in the three colour planes, in the region of interest is determined

$$\phi_P^k = \phi_{PG}^k + \phi_{PB}^k + \phi_{PR}^k + \epsilon \tag{2}$$

3) Weight factors are obtained from the gradients

$$\Phi_P^k = \frac{1}{\phi_P^k} \tag{3}$$

4) Finally, the missing colour content in the pixel of interest is generated

$$\bar{D}_P = \frac{\sum_k \Phi_P^k \tilde{D}_P^k}{\sum_k \Phi_P^k} \tag{4}$$

Variations to this process exist. Some algorithms use pixel differences rather than absolute pixel values [14]. Others manipulate gradients using mathematical processes such as integration. [13].

III. Corrective Terms To Be Analysed

All the terms in the general gradient relation can be viewed as directional terms, corrective terms or a combination of both. Equation 2 can be rewritten as:

$$\phi_P^k = \phi_{Pdir}^k + \phi_{Pcor}^k + \phi_{Pdir+cor}^k \tag{5}$$

where $\phi_{Pdir}^k = \phi_{PG}^k + \phi_{PB}^k + \phi_{PR}^k$ and $\phi_{Pcor}^k = \epsilon$.

A. Non-zero Corrective Term ϵ

This paper analyses two terms. The first is the non-zero wholly corrective term introduced in the state-of-the-art demosaicking methods by Pekkucuksen et al. [14] and Chen et al. [4], [5].

When establishing gradients using equation 2, the above methods include a non-zero corrective term ϵ to avoid a zero gradient and undefined weight sizes. However, in all the reviewed gradient algorithms no mention is made as to an appropriate size for ϵ. There is also no information regarding the degradation of the image should ϵ be omitted.

From an inspection of equations 2 and 3, if the corrective term ϵ is a small value, then in homogeneous or near-homogeneous regions of the image, the pixel to be determined would be undefined. Should the ϵ term be made larger, it begins to override the effect of the directional gradients. In the limiting case, as $\epsilon \to \infty$, $\Phi \to 0$ and the demosaicking process does not occur at all. From this preliminary observation, a closer analysis of this term behaviour is warranted.

B. Pseudo-Corrective Term β

The Bayer CFA is a mosaic of 3 separate colour planes superimposed on one another. If we split a portion of Figure 1 plane-wise, we obtain Figure 2.

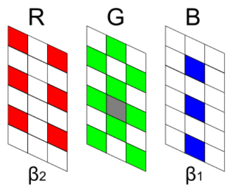


Fig. 2. A Bayer CFA section split into its component planes

Consider the problem of determining the green channel content in the gray pixel point marked in Figure 2 that only contains blue channel information. From equation 2, it is noted that the directional terms contributed from each of the colour planes are weighted equally. Mathematically, this can be represented by interplane weights, β_1=β_2=1. If it is assumed that the two β values are variable, 3 possible scenarios arise:

$$\beta_1 = \beta_2 = 1 \tag{6a}$$
$$\beta_1 = \beta_2 \neq 1 \tag{6b}$$
$$\beta_1 \neq \beta_2 \neq 1 \tag{6c}$$

From intuitive reasoning, the different component planes should have varying priorities. The explanation for this in the scenario above is as follows: the green plane is where the missing colour resides and holds the most importance. The blue plane shares the same pixel location as the desired point. The red plane neither contains the colour or pixel point desired and should contribute the least. This variable interplane weighting can be seen as:

$$\beta_2 < \beta_1 < 1 \tag{7}$$

which is a subset of the relation in equation 6c.

Equation 2 can then be abstracted to the following form:

$$\phi_P^k = \phi_{PG}^k + \rho_{PB}^k + \rho_{PR}^k + \epsilon \tag{8}$$

where $\rho_{PB}^k = \beta_1 \phi_{PB}^k$ and $\rho_{PR}^k = \beta_2 \phi_{PR}^k$

Two of the previously directional terms in equation 5, ϕ_{PB} and ϕ_{PR} have been corrected by β_1, β_2 to mitigate any error brought about by uniform weighting. As such the β terms can be seen as pseudo-corrective terms.

IV. Proposed Methodology

To adequately test the afforementioned corrective terms and avoid misinterpretation of results, two generic gradient-based demosaicking algorithms were created having different complexities. This was to give a more complete analysis of the corrective terms. The terms ϵ and β were be tested. Four image sets and four objective performance metrics were used to quantify analysis objectively.

Algorithm I carries out the demosaicking process only in the green plane. It is deliberately made simple in order to provide an analysis of the corrective terms on a basic demosaicking level. Algorithm II is a more complete technique that performs a complete demosaicking process for all three base colour planes. It is a method of comparable complexity with established methods.

A. Algorithm I

Assume the green content in pixel $B44$ of Figure 1 is to be determined. Initial estimates are first made in the cardinal directions:

$$\begin{aligned}\tilde{G}^N_{B44} &= G34 + 0.5\{B44 - B24\}\\ \tilde{G}^S_{B44} &= G54 + 0.5\{B44 - B64\}\\ \tilde{G}^W_{B44} &= G43 + 0.5\{B44 - B42\}\\ \tilde{G}^E_{B44} &= G45 + 0.5\{B44 - B46\}\end{aligned} \tag{9}$$

The gradients in each direction are then determined from the gradient equation. The directional descriptors must conform to the general direction of interpolation. In Algorithm I, as the green plane is considered in isolation, equation 2 reduces to

$$\phi^k_P = \phi^k_{PG} + \epsilon \tag{10}$$

and considering two of the cardinal directions, say the East and North; for Algorithm I

$$\begin{aligned}\phi^E_{B44} &= \{|G45 - G36| + |G45 - G56|\} + \epsilon\\ \phi^N_{B44} &= \{|G34 - G23| + |G34 - G25|\} + \epsilon\end{aligned} \tag{11}$$

ϕ^S_{B44} and ϕ^W_{B44} are found in a similar manner. The weights are then established as:

$$\Phi^k_{B44} = \frac{1}{\phi^k_{B44}} \tag{12}$$

where $k = \{N, E, W, S\}$. Finally the value of the green content in pixel $B44$ is given as:

$$\bar{G}_{34} = \frac{\sum_{k=\{N,E,W,S\}} \Phi^k_{B44}\tilde{G}^k_{34}}{\sum_{k=\{N,E,W,S\}} \Phi^k_{B44}} \tag{13}$$

When establishing green content in a red channel pixel, the above relations from equation 9 to 13 still hold, but the positions of the red and blue pixels are interchanged.

B. Algorithm II

Assume that the green content in pixel $B44$ of Figure 1 was required. The initial estimates are derived as in equation 9 except the inter-plane weighting concept is exploited. For example in the North direction,

$$\tilde{G}^N_{B44} = G34 + [(\beta_1\beta_2)\{B44 - B24\}] \tag{14}$$

The other initial estimates are derived in a similar manner. The gradient terms will be in the form of equation 2 abstracted to equation 8. Considering the North direction,

$$\begin{aligned}\phi^N_{B44} &= \phi^N_{B44G} + \beta_1\phi^N_{B44B} + \beta_2\phi^N_{B44R} + \epsilon\\ \phi^N_{B44G} &= |G34 - G23| + |G34 - G25| + |G34 - G14|\\ \phi^N_{B44B} &= |B44 - B24| + |B42 - B22| + |B46 - B26|\\ \phi^N_{B44R} &= |R33 - R13| + |R35 - R15|\end{aligned} \tag{15}$$

Taking the East direction as another example:

$$\begin{aligned}\phi^E_{B44G} &= |G45 - G36| + |G45 - G56| + |G45 - G47|\\ \phi^E_{B44B} &= |B44 - B46| + |B24 - B26| + |B64 - B66|\\ \phi^E_{B44R} &= |R35 - R37| + |R55 - R57|\end{aligned} \tag{16}$$

The weight factors are found using equation 12. For the final value, a refinement of equation 13 is used. Polling maps such as those used by Chen et al. [4] are developed. If the desirable interpolation direction is East-West, equation 17a is used. If a North-South direction is preferable, equation 17b is adopted. Finally, if the direction of interpolation is not apparent the default scheme of equation 17c is used.

$$\bar{G}_{34} = \frac{\sum_{k=\{E,W\}} \Phi^k_{B44}\tilde{G}^k_{34}}{\sum_{k=\{E,W\}} \Phi^k_{B44}} \tag{17a}$$

$$\bar{G}_{34} = \frac{\sum_{k=\{N,S\}} \Phi^k_{B44}\tilde{G}^k_{34}}{\sum_{k=\{N,S\}} \Phi^k_{B44}} \tag{17b}$$

$$\bar{G}_{34} = \frac{\sum_{k=\{N,E,W,S\}} \Phi^k_{B44}\tilde{G}^k_{34}}{\sum_{k=\{N,E,W,S\}} \Phi^k_{B44}} \tag{17c}$$

Equations 14 through 17 explain the interpolation process in the green plane. The red and blue planes are reconstructed in the same manner using the same type of relations bound by the general forms in Section II and the exception of a change in the terms depending on the pixel being determined.

C. Choice of Image Sets

Four image sets were used. Two standard image sets: the Kodak image set [17] because of its widespread use in publications and the McMaster set [18] as its demosaicked images' colour quality closely resemble real sensor data. Two non-standard image sets: the McGill University Land and Water set [19] and the Sample Microsoft Windows 7 set [20] were added to further test and validate the terms under scrutiny. All four image sets are freely available online.

D. Choice of Performance Metrics

Four objective measures were selected: the standard and colour peak signal-to-noise ratios (PSNR and CPSNR), the Structure SIMilarity index (SSIM) [21] and the Feature SIMilarity index (FSIM) [22]. PSNR and CPSNR were chosen as they are the traditional metrics for signal performance. However, due to the fact that the human visual system considers objects rather than absolute pixel values [23], the SSIM index provides a measure more consistent with subjective evaluation. Finally, FSIM is also presented to extend SSIM concepts by considering low level features rather than structures. Regular

FSIM and its chrominance variant FSIMc were used depending on whether a single plane or all three planes were under consideration.

In performance, the higher the value of PSNR and CPSNR, the closer the image represents the original. Using the modern SSIM and FSIM indicators, image reconstruction is measured from 0 to 1. A value equal to zero indicates a complete mismatch and a perfect match occurs when SSIM and FSIM are equal to 1.

V. Experimental Results

To validate the analysis, simulation of all the algorithms was done using MATLAB R2013a on an Intel(R) Core(TM)2 Duo CPU E7500 @2.93GHz processor.

A. A Subjective Evaluation of ϵ on Image Fidelity

An empirical analysis using Algorithm I from Section IV was performed on the 4th image from the McMaster image set and a cropped version of the 21st image in the Kodak set. Five values of ϵ were chosen ($\epsilon = 0, 1, 10, 100, 1000$) to observe whether a trend existed by varying ϵ. The resulting images are shown in Figure 3 and 4. These agree with the initial inspection of equations 2 and 3. When ϵ is zero, in both images, the homogeneous regions are not reconstructed properly. This is seen in the blotches in Figures 3(b) and 4(b). Demosaicking improves as ϵ is increased. However at very large values, the non-zero corrective term overpowers the directional terms in the gradient occasionally leading to no reconstruction occurring at pixel points with missing values. This is seen in the darkening of images observed at $\epsilon = 1000$ seen in Figures 3(f) and 4(f). This empirical analysis indicated there must be an optimum value for the non-zero corrective term. This point would be found between the range of $\epsilon = 1$ and $\epsilon = 100$.

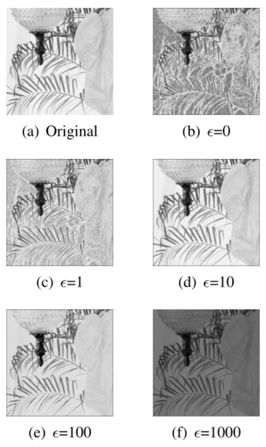


Fig. 3. Visual comparison of various ϵ values using McMaster Image 4

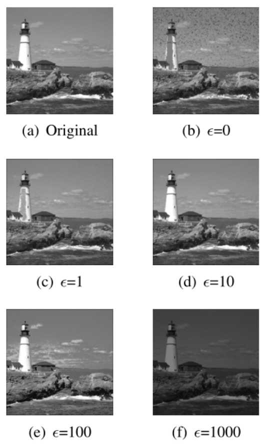


Fig. 4. Visual comparison of various ϵ values using Kodak Image 21

B. Effect of Variation of ϵ on Image Fidelity

To determine the optimum ϵ value, using Algorithm I from Section IV, each image in the Kodak set was tested with 33 different values of ϵ: from 0 to 14 in steps of 1 and from 15 to 100 in steps of 5. At each step, the values of $PSNR$, $SSIM$ and $FSIM$ were noted. A geometric mean of the images' results was then taken for each performance metric at each ϵ value and the resulting trend was plotted in Figures 5 to 7. The above process was repeated for the other three image sets so that a more complete representation of results was found. This gave an indication of how the performance metric changed for different ϵ values.

In all the three performance metrics shown in Figures 5 through to 7, for all the image sets, the non-zero corrective term peak occurred at ϵ=4.

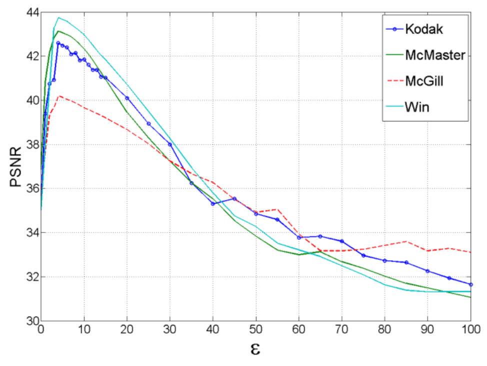


Fig. 5. Variation of PSNR for different ϵ values

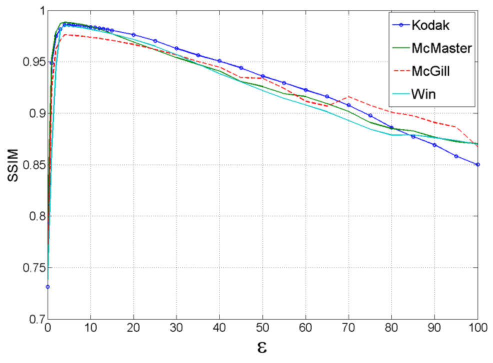


Fig. 6. Variation of SSIM for different ϵ values

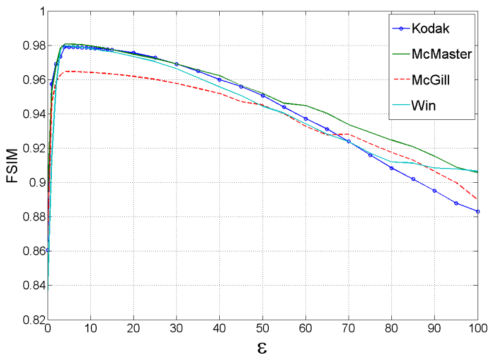


Fig. 7. Variation of FSIM for different ϵ values

C. Comparison of ϵ over Different Interpolation Direction Combinations

The singular peaking exhibited by the non-zero corrective term at $\epsilon = 4$ was explained by observing the behaviour of ϵ for different interpolation combinations to perform equation 13. When only two directions are considered the peaks occur at $\epsilon = 2$ and with three directions, the peak is at $\epsilon = 3$. These findings are presented in Table I. It should be noted that while the accuracy of the values presented is at most to 3 decimal places, increasing the accuracy leads to the trend highlighted in bold in the table. This implied that the peak value is related to the number of directions chosen to do the final interpolation step of the gradient algorithm that is equation 4.

D. A General Analysis of β on Image Fidelity

To analyse the effect of the pseudo-corrective terms β_1 and β_2, Algorithm II from Section IV was used. The three possibilities of equation 6 were selected when several images were picked at random and exposed to β_1 and β_2 varied from 0 to 2 in steps of 0.1. The values shown are the optimal variants for each of the possible combinations of equation 6. In each image set, the geometric means of $CPSNR$, $SSIM$ and $FSIM_c$ for all the four image sets were obtained for three different β_1, β_2 combinations and the results were tabulated in Table II. It was determined that optimal values of β occur at $\beta_1 = 0.8$ and $\beta_2 = 0.7$. This is in line with the expectations of Section III-B. A value of $\epsilon = 4$ was used as the non-corrective term as the algorithm uses all the cardinal directions.

TABLE I. Performance Metric Variations For Different ϵ Values Using The Kodak Image Set In Algorithm I For Various Interpolation Direction Combinations

ϵ	NS	NW	NE	SW	SE	EW	NSW	NSE	SWE	NWE	NWES
					$PSNR$						
0	36.75	36.39	36.40	36.38	36.44	36.94	36.10	36.13	36.16	36.14	35.41
1	39.15	38.53	38.48	38.48	38.55	39.48	39.45	39.46	39.54	39.51	39.39
2	**40.54**	**39.81**	**39.75**	**39.73**	**39.82**	**40.96**	39.98	39.99	40.08	40.04	40.75
3	40.40	39.70	39.64	39.62	39.70	40.81	**41.41**	**41.43**	**41.53**	**41.49**	40.93
4	40.28	39.59	39.54	39.52	39.59	40.66	41.32	41.35	41.44	41.41	**42.59**
5	40.13	39.47	39.42	39.39	39.47	40.49	41.21	41.23	41.32	41.29	42.48
6	39.98	39.36	39.30	39.28	39.35	40.33	41.10	41.11	41.21	41.19	42.39
7	39.71	39.13	39.08	39.05	39.12	40.03	40.83	40.84	40.93	40.91	42.08
8	39.65	39.09	39.04	39.01	39.08	39.96	40.83	40.84	40.92	40.91	42.14
9	39.37	38.85	38.80	38.78	38.85	39.67	40.54	40.56	40.63	40.62	41.81
					$SSIM$						
0	0.786	0.785	0.777	0.778	0.786	0.793	0.754	0.754	0.756	0.755	0.731
1	0.967	0.966	0.961	0.960	0.966	0.971	0.960	0.960	0.960	0.960	0.949
2	**0.979**	**0.978**	**0.973**	**0.973**	**0.978**	**0.983**	0.977	0.977	0.977	0.977	0.975
3	0.979	0.977	0.973	0.972	0.977	0.982	**0.983**	**0.983**	**0.984**	**0.984**	0.982
4	0.978	0.977	0.972	0.972	0.976	0.982	0.983	0.982	0.983	0.983	**0.986**
5	0.978	0.976	0.971	0.971	0.976	0.981	0.982	0.982	0.983	0.983	0.986
6	0.977	0.975	0.970	0.970	0.975	0.980	0.982	0.982	0.982	0.982	0.986
7	0.976	0.974	0.970	0.969	0.974	0.979	0.981	0.981	0.982	0.982	0.985
8	0.975	0.973	0.969	0.968	0.973	0.978	0.981	0.981	0.981	0.981	0.985
9	0.974	0.972	0.968	0.967	0.972	0.977	0.980	0.980	0.981	0.981	0.984
					$FSIM$						
0	0.901	0.892	0.892	0.891	0.893	0.901	0.876	0.877	0.876	0.876	0.860
1	0.965	0.961	0.960	0.960	0.960	0.967	0.962	0.961	0.961	0.961	0.957
2	**0.976**	**0.971**	**0.970**	**0.970**	**0.970**	**0.977**	0.969	0.969	0.969	0.970	0.969
3	0.976	0.971	0.970	0.969	0.970	0.977	**0.977**	**0.976**	**0.976**	**0.977**	0.973
4	0.975	0.971	0.969	0.969	0.970	0.977	0.977	0.976	0.976	0.977	**0.979**
5	0.975	0.971	0.969	0.969	0.970	0.977	0.977	0.976	0.976	0.976	0.979
6	0.975	0.970	0.969	0.969	0.970	0.976	0.976	0.976	0.976	0.976	0.979
7	0.975	0.970	0.969	0.968	0.970	0.976	0.976	0.976	0.976	0.976	0.979
8	0.974	0.970	0.968	0.968	0.969	0.976	0.976	0.976	0.976	0.976	0.979
9	0.974	0.970	0.968	0.968	0.968	0.975	0.976	0.976	0.975	0.976	0.979

TABLE II. Performance Metric Variations For Different β_1, β_2 Combinations

Set	$\beta_1 = 1, \beta_2 = 1$	$\beta_1 = 0.8, \beta_2 = 0.8$	$\beta_1 = 0.8, \beta_2 = 0.7$
		$CPSNR$	
Kodak	38.469	41.111	**41.356**
McMaster	37.661	39.407	**39.756**
McGill	34.595	35.701	**35.850**
Win7	38.612	39.942	**40.137**
		$SSIM$	
Kodak	0.9721	0.9813	**0.9818**
McMaster	0.9512	0.9652	**0.9676**
McGill	0.9305	0.9466	**0.9484**
Win7	0.9435	0.9601	**0.9630**
		$FSIM_c$	
Kodak	0.9690	0.9720	**0.9723**
McMaster	0.9655	0.9684	**0.9688**
McGill	0.9413	0.9462	**0.9467**
Win7	0.9623	0.9650	**0.9653**

E. A Comparison of Algorithm II with Established Methods

The collective contributions of the directional and correction terms in Algorithm II are compared with current established gradient algorithms that are largely directional based. These are the Wang Method [11] and ESFB [14] developed in 2012; MDWI [7] and MGBI [12] presented in 2013.

All four image sets were exposed to each algorithm in turn. For each algorithm, the geometric means of the $CPSNR$,

$SSIM$ and $FSIM_c$ for each image set were noted and tabulated in Tables III, IV and V respectively. In the proposed Algorithm II, a value of $\epsilon = 4$ was used as the non-corrective term and $\beta_1 = 0.8, \beta_2 = 0.7$ for the pseudo-corrective inter-plane terms.

TABLE III. $CPSNR$ EVALUATION OF SEVERAL DEMOSAICKING TECHNIQUES ON FOUR IMAGE SETS (IN dB)

Set	Wang [11]	ESFB [14]	MDWI [7]	MGBI [12]	Alg. II
Kodak	39.987	41.153	40.135	**41.534**	41.356
McMaster	38.962	38.437	39.750	38.434	**39.756**
McGill	35.403	35.490	35.717	35.543	**35.850**
Win7	39.619	39.321	39.998	39.251	**40.137**
Mean	38.448	38.545	38.855	38.631	**39.219**

TABLE IV. $SSIM$ EVALUATION OF SEVERAL DEMOSAICKING TECHNIQUES ON FOUR IMAGE SETS

Set	Wang [11]	ESFB [14]	MDWI [7]	MGBI [12]	Alg. II
Kodak	0.9780	0.9805	0.9759	0.9710	**0.9818**
McMaster	**0.9689**	0.9607	0.9678	0.9559	0.9676
McGill	0.9429	0.9423	0.9444	0.9344	**0.9484**
Win7	0.9650	0.9561	**0.9660**	0.9505	0.9630
Mean	0.9636	0.9598	0.9635	0.9529	**0.9651**

TABLE V. $FSIM_c$ EVALUATION OF SEVERAL DEMOSAICKING TECHNIQUES ON FOUR IMAGE SETS

Set	Wang [11]	ESFB [14]	MDWI [7]	MGBI [12]	Alg. II
Kodak	**0.9859**	0.9800	0.9531	0.9476	0.9723
McMaster	**0.9843**	0.9765	0.9542	0.9525	0.9688
McGill	**0.9688**	0.9580	0.9250	0.9192	0.9467
Win7	**0.9815**	0.9743	0.9514	0.9481	0.9653
Mean	**0.9801**	0.9722	0.9458	0.9418	0.9632

Table III highlights the $CPSNR$ performance. The generic Algorithm II was the best performing method in three of the image sets. In Table IV, Algorithm II is comparable with the Wang and MDWI methods by the $SSIM$ metric that analyses structure rather than pixel statistics. It performs best overall for this metric Finally, when considering low-level features, Algorithm II ranks third behind the ESFB and Wang methods as shown by the $FSIM_c$ metric in Table V.

In general, Algorithm II is comparable and in some cases even exceeds some established techniques. This can be attributed to the contributions made by optimising the corrective terms of the gradient equation and combining them with the directional terms.

VI. CONCLUSION

A study of improving gradient based demosaicking was performed highlighting the influence of corrective terms. Two such terms: the non-zero corrective term ϵ and the pseudo-corrective term β were formally introduced and the rationale behind each was explained. Two generic algorithms using either one or both corrective terms were then developed. An objective performance metric analysis of the corrective terms was done. The proposed algorithm employing both corrective terms was compared with several recently established techniques. Through a geometric mean analysis over four separate image sets, the proposed generic algorithm was found to be comparable and sometimes even superior to the established techniques.

Of note, as the two corrective terms are general and derived from intuitive reasoning, they can be incorporated to existing methods for a boost in performance.

REFERENCES

[1] B. E. Bayer, "Color imaging array," U.S. Patent 3 971 065, Jul. 20, 1976.

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

[3] X. Li, B. K. Gunturk, and L. Zhang, "Image demosaicking: A systematic survey," in *Proc. SPIE 6822, Visual Communications and Image Processing 2008, SPIE-IS&T Electronic Imaging*, San Jose, CA, Jan. 2008.

[4] X. Chen, G. Jeon, and J. Jeong, "Voting based directional method and its application to still color image demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 2, pp. 255–262, Feb. 2014.

[5] X. Chen, , L. He, G. Jeon, and J. Jeong, "Local adaptive directional color filter array interpolationbased on inter-channel correlation," *Optics Communications (Elsevier)*, vol. 324, pp. 269–276, 2014.

[6] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Minimized-laplacian residual interpolation for color image demosaicking," in *Proc. SPIE-IS&T 9023, Digital Photography X*, Mar. 2014.

[7] X. Chen, G. Jeon, J. Jeong, and L. He, "Multi-directional weighted interpolation and refinement method for bayer pattern cfa demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, submitted for publication in 2014.

[8] K. Lee, S. Jeong, J. soo Choi, and S. Lee, "Multiscale edge-guided demosaicking algorithm," in *18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*, Jeju, Korea, Jun. 2014.

[9] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Residual interpolation for color image interpolation," in *2013 IEEE International Conference on Image Processing (ICIP 2013)*, Melbourne, Australia, Sep. 2013.

[10] D.-C. Sung and H.-W. Tsao, "A gradient based edge sensing scheme for color filter array demosaicking," in *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, Tokyo, Japan, Oct. 2013.

[11] W. Jin, "Improved color interpolation method based on bayer image," in *Proc. SPIE 8420, 6th International Symposium on Advanced Optical Manufacturing and Testing Technologies*, 2012.

[12] I. Pekkucuksen and Y. Altunbasak, "Multiscale gradients-based color filter array interpolation," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 157–165, Jan. 2013.

[13] K. H. Chung and Y. H. Chan, "Low-complexity color demosaicking algorithm based on integrated gradients," *Journal of Electronic Imaging*, vol. 19, no. 2, Apr. 2010.

[14] I. Pekkucuksen and Y. Altunbasak, "Edge strength filter based color filter array interpolation," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 393–397, Jan. 2012.

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

[16] W. Lu and Y.-P. Tan, "Color filter array demosaicking: New method and performance measures," *IEEE Trans. Image Process.*, vol. 12, no. 10, pp. 1194–1210, Oct. 2003.

[17] (2014). [Online]. Available: http://r0k.us/graphics/kodak/

[18] (2014). [Online]. Available: http://www4.comp.polyu.edu.hk/~cslzhang/

[19] (2014). [Online]. Available: http://tabby.vision.mcgill.ca/

[20] (2014). [Online]. Available: https://www.flickr.com/photos/37121025@N07/sets/72157616417074008/

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[22] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: A feature similarity index for image quality assessment," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2378–2386, Aug. 2011.

[23] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it?" *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009.

G.4 Fourth Publication – IEEE AFRICON 2015

A Pentomino-Based Path Inspired Demosaicking Technique for the Bayer Color Filter Array

Kinyua Wachira and Elijah Mwangi
School of Engineering,
University of Nairobi,
P.O. Box 30197 Nairobi 00100, Kenya
Email: kinyua.wachira@students.uonbi.ac.ke

Gwanggil Jeon
Department of Embedded Systems Engineering,
Incheon National University,
12-1 Songdo-dong Yeonsu-gu, Incheon 406-772, Republic of Korea
Email: gjeon@inu.ac.kr

***Abstract*—Spatial demosaicking methods are attractive due to their robust reconstruction capabilities and low computational load. This paper presents a new gradient-based spatial demosaicking algorithm with contributions from combinatorial geometry. It introduces the use of square tiling primitives called pentominoes to dictate the difference-term pairs combinations for gradient generation. The proposed algorithm is compared with current established methods. Reconstruction performance is objectively measured using three commonly documented metrics (MSE, SSIM and CPSNR). Three different image sets are used (Kodak, McMaster-IMAX and Condat) to evaluate demosaicking robustness. An significant improvement in performance over existing methods has been obtained through MATLAB simulation.**

***Keywords*—Color filter array, chirality, combinatorial geometry, demosaicking, difference-term pairs, pentominoes, polyominoes**

I. Introduction

Color images, along with the digital still camera (DSC), are an omnipresent feature of modern society. DSCs are found either as standalone elements or embedded within other consumer products. The generation of color images using DSCs is a multi-stage pipelined process composed of focus and exposure control, white balance adjustment, demosaicking, noise suppression and color alteration [1]. As human vision is trichromatic, a color image needs three base color samples at every sampling point. The common base colors used are red, green and blue as they roughly correspond to the short(S), medium(M) and long(L) wavelength spectral responses of the cones in the human eye [2]. Most camera sensors, however, are monochrome [3]. They record intensity per sample point and cannot capture color information. To generate the color content, a color filter array or mosaic (CFA/M) is used. It separates colors at the sampling level prior to passing a single color intensity per point to the image sensor. A software process is then used to reconstruct the full color information from the sub-sampled color data read from the sensor. This reconstruction process is called demosaicking. The approximation of the S, M and L spectral responses has led to the use of different base color combinations resulting in varying CFAs from camera manufacturers. The most commonly documented and analyzed is the Bayer CFA [4] shown in Figure 1. Green is sampled twice as much as red or blue as human eye is most sensitive to the M wavelength region. Recent literature surveys in the field [5]–[7] classify demosaicking methods as spatial-based, spectral-based or image model based. This paper is biased towards the spatially motivated methods because they directly deal with actual recorded intensity data, produce good results and are of a low enough computational complexity to allow for real time processing. The simplest form of spatial demosaicking is done by Bilinear Interpolation (BI) [7]. This method, however, does not consider the correlation between different color planes. Edge techniques such as Edge Directed Interpolation (EDI), Hamilton-Adams (HA) [7] and the Malvar-He-Cutler algorithm (MHC) [8] are adaptive methods that use local pixel neighborhood information to perform the reconstruction. Modern spatial algorithm extend the versatility of edge methods by using gradients in their interpolation process [9]–[14]. Each gradient is made up of a combination of difference-terms.

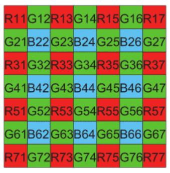

R11 G12 R13 G14 R15 G16 R17
G21 B22 G23 B24 G25 B26 G27
R31 G32 R33 G34 R35 G36 R37
G41 B42 G43 B44 G45 B46 G47
R51 G52 R53 G54 R55 G56 R57
G61 B62 G63 B64 G65 B66 G67
R71 G72 R73 G74 R75 G76 R77

Fig. 1. The Bayer CFA Pattern in a 7-by-7 region (US Patent 3971605)

The algorithm proposed extends the work of modern gradient-based spatial demosaicking techniques by presenting a new difference-term generation technique based on combinatorial geometry primitives for green plane interpolation. It also makes use of a corrective error constant and polling maps to improve reconstruction. The paper is divided as follows: Section II explains the elements that made a modern gradient-based demosaicking method. Section III draws attention to the new contributions before the entire algorithm is presented in Section IV. Section V explains the objective evaluation mechanisms and the image data sets along with the rationale behind their choice. It also presents the results of simulation. Finally Section VI provides in brief the conclusions drawn from the experimental results.

II. Generic Gradient Based Spatial Demosaicking Methodology

Spatial demosaicking is an adaptive interpolation process. A sufficient number of descriptors in various directions from a pixel of interest are chosen to adequately predict missing color content in that pixel. Gradient-based methods are a four stage process:

1) Directional estimates of the desired pixel are found,

$$\widetilde{D}_P^k = (D_{P-1}^k) + c_1(W_P^k - W_{P-2}^k) \quad (1)$$

where P is the pixel location, D is the desired color plane, W is a neighboring color plane, k is the interpolation direction and c_1 is a non-zero constant.

2) Directional gradients based on a sum of difference-term pairs from the three base color planes in the region of interest are established,

$$\Gamma_P^k = \Gamma_{G,P}^k + \Gamma_{B,P}^k + \Gamma_{R,P}^k + c_2 \quad (2)$$

where Γ indicates a sum of difference-terms and c_2 is a corrective constant.

3) Weight factors are obtained from the gradients:

$$\gamma_P^k = \frac{1}{\Gamma_P^k} \quad (3)$$

4) The missing color content in the pixel of interest is obtained from the estimates and weights derived:

$$\widehat{D}_P = \frac{\sum_k \gamma_P^k \widetilde{D}_P^k}{\sum_k \gamma_P^k} \quad (4)$$

III. Novel Contributions

The pixels for the gradients in equation 2 are often picked to fall wholly along the desired direction. This leads to very large bounding windows for interpolation due to the restrictive paths followed. Shrinking the region results in fewer descriptors being available leading to poor reconstruction. The proposed method looks at the pixel selection problem using combinatorial geometry. The square tiled CFA can be broken into smaller primitives. The primitives are then selected based on a set of constraints defined by the authors to determine pixels for gradient determination. This selection process is done for the green plane interpolation step as it contains most of the raw sampled sensor data.

A. Polyomino Theory and Pentomino Inspired Paths

The Bayer CFA is a square tiling in two dimensional Euclidean space. Polyominoes are shapes made by connecting a certain number of equal-sized squares, each joined together with at least one other square along an edge [15]. Polyominoes occur in various sizes depending on the number of interconnected squares. The most recognized use of polyominoes is in the popular game of Tetris - a puzzler using randomly generated tetrominoes (4-square polyominoes). A subset of polyominoes is required to ensure a sufficient number of descriptor pixels for establishing gradients. By the nature of the Bayer CFA, it was determined for n-square polyominoes:

- The maximum number of green pixels bounded by the polyomino, g, for odd or even valued n;

$$g_{odd} = (\frac{n+1}{2}), g_{even} = (\frac{n}{2}) \quad (5)$$

- The number of paths found within the polyomino, p;

$$p = \frac{g!}{(g-2)!2!} \quad (6)$$

Exceptions exist to the above rules. An optimal value allowing a good number of descriptors in our Bayer CFA problem was found to be n = 5. This is the pentomino subset of polyominoes. There are 12 unique pentomino shapes possible if it is assumed that the geometric processes of reflection and rotation are allowed. These are shown in Figure 2. Pentominoes with the chirality property are marked by a prime symbol.

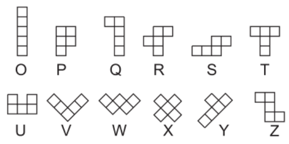


Fig. 2. The 12 pentomino constructs

Based on equations 5 and 6, each pentomino can have three green pixels and has three distinct paths between these pixels. Visual inspection determined that some of the paths were antagonistic to a desired interpolation direction. Due to this, the authors assigned the pixels and paths as shown in Figure 3 based on the following constraints:

1) A pixel of interest (marked gray) must share a vertex with an edge pixel.
2) Pentomino path-pairs should remain distinct from one another even after rotation.
3) Only two of the three possible paths are chosen to form a unique path-pair for each pentomino.

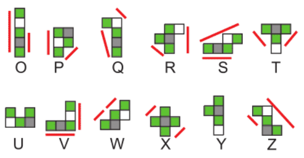


Fig. 3. Distinct pentomino path-pairs

From Figure 3, it can be seen that no new paths exist for the U and Y pentominoes. The three possible path-pairs for those configurations have been assigned to the P, R and T pentominoes.

B. Prioritized Weighting

Any CFA is a combination of several color planes interleaved to become one structure. The Bayer CFA is shown in its constituent form in Figure 4. When determining the pixel of interest (marked gray in Figure 4), the green plane has the most influence since the pixel resides within it. The blue plane is of secondary importance as this plane contains the recorded sensor data from the camera. Finally the red plane offers no position or intensity information and is of the least importance. This priority weighting is encoded by introducing two factors b_1 and b_2 such that the condition $b_2 < b_1 < 1$ is met. It was determined empirically that $b_2 = 0.7$ and $b_1 = 0.8$ gave best results.

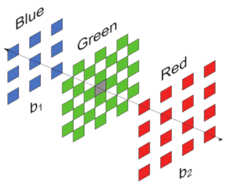


Fig. 4. The decomposed Bayer CFA of Figure 1

IV. Proposed Method

The authors opted to use the path-pairs of the O,Q and S pentominoes to determine gradients in the green plane. The elongated nature of these particular shapes bounds pixels along a desired direction more strictly than the P,R,T and X pentominoes. The W and Z pentominoes are not ideal for a N,W,E,S direction interpolation process. Finally, the V-pentomino has two cardinal directions and as such is also not ideal for determining a single direction.

A. Green Content Interpolation

Consider attempting to establish the green color content at the pixel marked B44 in the Bayer arrangement of Figure 1. The four cardinal direction based estimates can be determined as:

$$\begin{aligned}\tilde{G}^N_{B44} &= G34 + [(b_1 b_2)(B44 - B24)]\\ \tilde{G}^S_{B44} &= G54 + [(b_1 b_2)(B44 - B64)]\\ \tilde{G}^W_{B44} &= G43 + [(b_1 b_2)(B44 - B42)]\\ \tilde{G}^E_{B44} &= G45 + [(b_1 b_2)(B44 - B46)] \qquad (7)\end{aligned}$$

In finding the gradient all three color planes are used. For the green pixel contributions, the O,Q and S-pentominoes are used as in Figure 5, to generate six difference terms per cardinal direction. Considering the North,

$$\begin{aligned}\Gamma^N_{B44} &= \Gamma^N_{G,B44} + \Gamma^k_{B,B44} + \Gamma^N_{R,B44} + c_2\\ \Gamma^N_{G,B44} &= |path_{Q,N}| + |path_{O,N}| + |path_{S,N}|\\ \Gamma^N_{B,B44} &= b_1(|B44 - B24| + |B42 - R22| + |B46 - R26|)\\ \Gamma^N_{R,R44} &= b_2(|R35 - R15| + |R33 - R13|) \qquad (8)\end{aligned}$$

where $path_Q = (|G54 - G23| + |G34 - G23|)$, $path_O = (|G54 - G34| + |G54 - G14|)$ and $path_S = (|G54 - G34| + |G54 - G25|)$. The inter-plane weights are b_1 and b_2.

Gradients Γ^W_{B44}, Γ^S_{B44} and Γ^E_{B44} are generated in a similar manner and their O,Q,S paths are obtained by rotating the pentominoes anticlockwise by 90°, 180° and 270° respectively. The West paths for B44 are shown in Figure 5 for reference. The gradient weights are then found as:

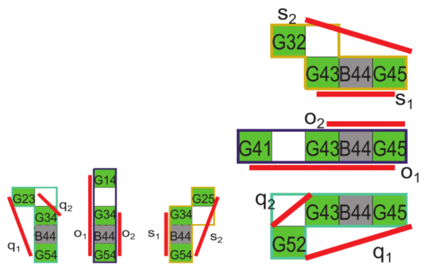
$$\gamma^k_{B44} = \frac{1}{\Gamma^k_{B44}} \qquad (9)$$

where k corresponds to each of the cardinal directions. Polling maps such as those developed by Chen et al. [16] are used to refine the final value. Equations 10a and 10b are used if interpolation follows a binary direction sense. Otherwise, equation 10c is employed. A similar treatment is used to determine green color content in red pixel locations.

$$\hat{G}_{B44} = \frac{\sum_{n\in\{N,S\}}[\gamma^k_{B44}\tilde{G}^k_{B44}]}{\sum_{n\in\{N,S\}}\gamma^k_{B44}} \qquad (10a)$$

$$\hat{G}_{B44} = \frac{\sum_{k\in\{W,E\}}[\gamma^k_{B44}\tilde{G}^k_{B44}]}{\sum_{k\in\{W,E\}}\gamma^k_{B44}} \qquad (10b)$$

$$\hat{G}_{B44} = \frac{\sum_{k\in\{N,S,W,E\}}[\gamma^k_{B44}\tilde{G}^k_{B44}]}{\sum_{k\in\{N,S,W,E\}}\gamma^k_{B44}} \qquad (10c)$$



(a) North Direction (b) West Direction

Fig. 5. The Q, O and S-Pentomino Green Paths for B44

B. Blue/Red Content Interpolation

Red or blue color determination either involves finding content in an opposing plane (red in blue plane or vice versa) or finding content from the green plane.

1) Opposing Plane Considerations: Consider determining the red color in the pixel B44 from Figure 1. As the green content in each pixel has already been determined, a difference-based solution similar to Chen et al. [10] is used. Diagonal direction estimates are taken:

$$\begin{aligned}\tilde{\rho}^{NW}_{B44} &= R33 - \hat{G}_{R33}, \tilde{\rho}^{NE}_{B44} = R35 - \hat{G}_{R35}\\ \tilde{\rho}^{SE}_{B44} &= R55 - \hat{G}_{R55}, \tilde{\rho}^{SW}_{B44} = R53 - \hat{G}_{R53} \qquad (11)\end{aligned}$$

Gradients are found using a variant of equation 2:

$$\Gamma^k = \Gamma^k_{G,in} + \Gamma^k_{G,out} + \Gamma^k_R + c_2 \qquad (12)$$

where k denotes ordinal directions. Taking the NW direction, the components of equation 12 are:

$$\begin{aligned}\Gamma^{NW}_{G,in} &= (2|\bar{G}_{B44} - \bar{G}_{R33}| + 2|\bar{G}_{R33} - \bar{G}_{B22}|)\\ \Gamma^{NW}_{G,out} &= (|G43 - G32| + |G34 - G23|)\\ \Gamma^{NW}_R &= b_2(|R33 - R55|) \qquad (13)\end{aligned}$$

The process is repeated for Γ_{R44}^{NE}, Γ_{R44}^{SE} and Γ_{R44}^{SW}. The weights are found by equation 9 and the final value is:

$$\widehat{R}_{B44} = \widehat{G}_{B44} + \frac{\sum_{k\in\{NW,NE,SW,SE\}}[\gamma_{B44}^{k}\widetilde{\rho}_{B44}^{k}]}{\sum_{k\in\{NW,NE,SW,SE\}}\gamma_{B44}^{k}} \tag{14}$$

2) In-Green Pixel Consideration: Consider trying to find blue content in pixel G45. The blue plane is now quincuncial and the green content interpolation steps can be used. Consider the North direction; its estimate and gradient are given by equations 15a and 15b. The final value is in equation 15c:

$$\widetilde{B}_{B45}^{N} = \widehat{B}_{R35} + b_1 b_2(G45 - G25) \tag{15a}$$

$$\begin{aligned}
\Gamma_{G45}^{N} &= \Gamma_{B,G45}^{N} + \Gamma_{G,G45}^{N} + c_2 \\
\Gamma_{B,G45}^{N} &= (3|\widehat{B}_{R35} - \widehat{B}_{R15}| + 3|\widehat{B}_{R35} - \widehat{B}_{R55}|) \\
&\quad + (|\widehat{B}_{R35} - B24| + |\widehat{B}_{B35} - B26|) \\
\Gamma_{G,G45}^{N} &= (|G45 - G25| + |G34 - G14| + |G36 - G16|)
\end{aligned} \tag{15b}$$

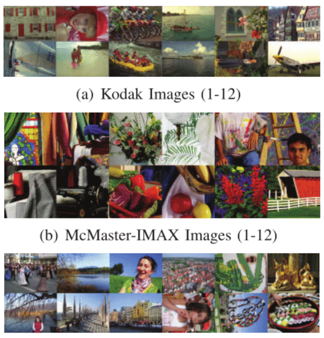
$$\widehat{B}_{G45} = \frac{\sum_{k\in\{N,E,W,S\}}[\gamma_{G45}^{k}\widetilde{B}_{G45}^{k}]}{\sum_{k\in\{N,E,W,S\}}\gamma_{G45}^{k}} \tag{15c}$$

The same treatment is applied to determine red content.

V. Results

A. Image Base Sets and Indices Used

The proposed algorithm made use of 36 randomly chosen images from three commonly documented image bases: the Kodak [17], McMaster-IMAX [18] and Condat bases [19]. Twelve images per base as shown in Figure 6. Three objective measures were used. For single color plane analysis, the Mean Square Error (MSE) and the Structure SIMilarity Index were used. For a full image, the Color Peak Signal-to-Noise Ratio (CPSNR) metric was employed.


(a) Kodak Images (1-12)

(b) McMaster-IMAX Images (1-12)

(c) Condat Images (1-12)

Fig. 6. Test Images Used

B. The Simulation Procedure and Performance

The experimental simulation was performed using MATLAB R2013a running on an Intel(R) Core(TM)2 Duo CPU E5700 @ 2.93 GHz processor. The method proposed was compared with seven spatial-based demosaicking algorithms. Three are classical edge methods: BI, EDI and HA [7]. The others are recently published gradient-based techniques: Wang method [9], ESFB [14], MDWI [10] and MGBI [12]. Testing was done in two stages. The first stage was to gauge the reconstruction the green plane interpolation and the second measured full demosaicking algorithm reconstruction. An analysis of the green plane was necessary because the pentomino paths are only used in the green content interpolation stage.

The Kodak images, after being converted to Bayer CFA equivalents, were passed to each of the algorithms in turn and the reconstruction of the green plane was done and measured using the MSE and SSIM indices. The results of this were tabulated in Tables I and II. An average of all the images, to show the overall robustness of the methods chosen, was taken and also noted. Bold font is used to indicated best performance and a ranking system shows the position of the proposed method compared to the other techniques. The McMaster-IMAX and Condat images were similarly treated and their results documented in Tables I and II.

TABLE I. MSE Evaluation Results Over Three Image Sets

Kodak Set									
Image	**BI**	**EDI**	**HA**	**WANG**	**ESFB**	**MDWI**	**MGBI**	**Proposed**	**Rank**
1	20.001	15.922	10.552	11.064	7.002	9.639	3.630	6.174	2
2	5.812	5.866	3.068	4.275	2.647	3.329	2.630	2.251	1
3	16.170	13.741	7.629	9.128	5.201	8.750	3.553	4.409	2
4	14.111	11.018	6.299	7.133	4.736	7.170	2.545	4.384	2
5	4.866	3.717	1.966	2.424	1.832	2.414	1.740	1.367	1
6	22.102	13.067	12.238	10.468	10.274	10.047	5.441	6.689	2
7	5.311	3.978	2.491	2.791	1.977	2.684	1.690	1.485	1
8	5.083	3.934	2.164	2.506	1.963	2.459	1.618	1.422	1
9	11.539	11.309	5.502	7.301	4.368	6.124	3.482	3.744	2
10	8.290	6.976	3.337	4.152	2.804	4.097	1.413	2.258	2
11	12.248	12.698	6.929	8.814	4.971	7.707	3.925	4.843	2
12	5.946	5.214	3.063	3.641	2.524	3.489	5.254	1.896	1
Mean	10.957	8.953	5.437	6.142	4.192	5.659	**3.077**	3.410	2
McMaster – IMAX Set									
1	16.417	14.491	13.689	13.478	17.582	11.766	17.164	12.451	2
2	7.562	6.335	4.825	5.332	7.000	4.209	6.334	4.260	2
3	15.532	12.528	10.850	11.195	10.470	8.371	8.775	8.211	1
4	8.829	4.935	5.233	6.049	7.586	2.587	6.962	4.560	2
5	6.968	5.429	5.635	5.753	8.276	4.242	7.752	5.067	2
6	3.442	3.160	2.868	3.794	7.628	2.041	7.222	2.805	2
7	11.501	12.093	6.726	8.657	3.963	7.131	2.828	4.327	2
8	6.222	4.961	3.756	3.743	2.945	3.387	2.413	2.718	2
9	3.950	3.650	2.358	2.937	4.893	2.070	4.625	2.236	2
10	3.016	2.710	2.174	2.475	3.515	1.965	3.283	2.168	2
11	7.243	6.987	6.397	8.049	13.042	4.842	13.002	6.647	2
12	11.923	9.394	7.012	8.254	8.966	6.509	8.246	5.504	1
Mean	8.550	7.223	5.960	6.643	7.989	**4.927**	7.384	5.080	2
Condat Set									
1	13.181	10.434	7.967	8.051	8.480	7.128	7.116	5.503	1
2	21.197	20.632	17.483	18.414	14.922	17.054	12.604	13.987	2
3	7.263	6.710	5.309	5.698	6.413	4.924	5.729	4.184	1
4	19.392	16.350	14.267	14.617	14.013	13.927	11.824	11.240	1
5	11.636	4.583	5.080	3.906	8.416	3.991	7.121	3.538	1
6	9.922	7.889	6.198	5.685	8.013	5.847	6.886	4.659	1
7	22.348	17.319	16.693	16.330	14.492	14.912	11.825	12.284	2
8	11.218	7.015	6.779	5.300	7.024	5.550	4.760	4.069	1
9	14.325	9.634	8.971	7.920	9.405	7.689	6.662	5.932	1
10	3.732	2.966	2.987	3.178	5.372	2.015	4.757	2.924	2
11	11.917	9.101	7.561	7.761	9.002	7.783	7.287	5.845	1
12	4.290	3.445	2.810	3.378	5.164	2.284	4.807	2.834	2
Mean	12.535	9.673	8.509	8.353	9.226	7.759	7.615	**6.417**	1

The image sets were then exposed to the full demosaicking process for each method and the average CPSNR results tabulated in Table III. An overall average of all the 36 images was then calculated and recorded. The rank of the proposed method relative to the others was also noted. Bold font indicated best performance.

TABLE II. SSIM EVALUATION RESULTS OVER THREE IMAGE SETS

Kodak Set									
Image	**BI**	**EDI**	**HA**	**WANG**	**ESFB**	**MDWI**	**MGBI**	**Proposed**	**Rank**
1	0.9635	0.9611	0.9784	0.9738	0.9827	0.9772	0.9859	0.9844	2
2	0.9664	0.9783	0.9856	0.9806	0.9839	0.9791	0.9813	0.9844	2
3	0.9533	0.9781	0.9859	0.9811	0.9851	0.9802	0.9836	0.9848	3
4	0.9255	0.9602	0.9792	0.9741	0.9796	0.9734	0.9563	0.9807	1
5	0.9833	0.9882	0.9904	0.9882	0.9864	0.9824	0.9826	0.9863	5
6	0.9253	0.9691	0.9771	0.9779	0.9801	0.9767	0.9805	0.9828	1
7	0.9669	0.9781	0.9833	0.9800	0.9807	0.9753	0.9746	0.9805	3
8	0.9588	0.9774	0.9873	0.9843	0.9852	0.9802	0.9849	0.9864	2
9	0.9515	0.9721	0.9825	0.9781	0.9806	0.9755	0.9780	0.9808	2
10	0.9386	0.9659	0.9810	0.9765	0.9779	0.9717	0.9795	0.9789	3
11	0.9494	0.9701	0.9810	0.9761	0.9841	0.9795	0.9841	0.9839	3
12	0.9635	0.9714	0.9749	0.9723	0.9545	0.9666	0.8510	0.9730	2
Mean	0.9538	0.9725	0.9822	0.9786	0.9801	0.9765	0.9685	**0.9822**	1
McMaster − IMAX Set									
1	0.9537	0.9517	0.9487	0.9466	0.9270	0.9498	0.9130	0.9444	6
2	0.9697	0.9745	0.9750	0.9732	0.9671	0.9732	0.9644	0.9711	5
3	0.9615	0.9739	0.9760	0.9734	0.9713	0.9690	0.9665	0.9733	3
4	0.9716	0.9747	0.9720	0.9715	0.9676	0.9619	0.9603	0.9669	6
5	0.9746	0.9679	0.9657	0.9626	0.9487	0.9602	0.9426	0.9572	6
6	0.9848	0.9763	0.9708	0.9692	0.9492	0.9681	0.9468	0.9659	6
7	0.9560	0.9699	0.9794	0.9747	0.9810	0.9715	0.9769	0.9780	3
8	0.9779	0.9839	0.9853	0.9837	0.9799	0.9796	0.9760	0.9803	4
9	0.9840	0.9849	0.9814	0.9800	0.9760	0.9804	0.9722	0.9813	4
10	0.9822	0.9814	0.9801	0.9795	0.9722	0.9728	0.9669	0.9735	5
11	0.9816	0.9722	0.9678	0.9623	0.9456	0.9725	0.9435	0.9659	5
12	0.9652	0.9693	0.9714	0.9676	0.9650	0.9702	0.9643	0.9715	1
Mean	0.9719	**0.9734**	0.9728	0.9704	0.9626	0.9691	0.9578	0.9691	5
Condat Set									
1	0.9538	0.9699	0.9743	0.9729	0.9740	0.9710	0.9535	0.9760	1
2	0.9026	0.9300	0.9423	0.9404	0.9495	0.9474	0.9387	0.9543	1
3	0.9710	0.9746	0.9742	0.9713	0.9699	0.9691	0.9628	0.9734	3
4	0.9471	0.9651	0.9695	0.9695	0.9692	0.9694	0.9686	0.9733	1
5	0.9803	0.9812	0.9742	0.9728	0.9614	0.9689	0.9605	0.9705	3
6	0.9663	0.9786	0.9750	0.9770	0.9661	0.9764	0.9623	0.9756	4
7	0.9291	0.9609	0.9655	0.9676	0.9721	0.9709	0.9742	0.9757	1
8	0.9644	0.9802	0.9803	0.9827	0.9779	0.9768	0.9748	0.9810	2
9	0.9534	0.9723	0.9732	0.9748	0.9691	0.9707	0.9697	0.9744	2
10	0.9918	0.9888	0.9849	0.9845	0.9788	0.9781	0.9748	0.9804	5
11	0.9800	0.9797	0.9718	0.9710	0.9686	0.9675	0.9661	0.9713	4
12	0.9855	0.9865	0.9850	0.9843	0.9784	0.9797	0.9745	0.9808	5
Mean	0.9604	0.9723	0.9725	0.9724	0.9696	0.9705	0.9650	**0.9739**	1

TABLE III. AVERAGE CPSNR EVALUATION RESULTS OVER SEVERAL DEMOSAICKING SCHEMES IN (DB) IN THREE IMAGE SETS

Image Set	**BI**	**EDI**	**HA**	**WANG**	**ESFB**	**MDWI**	**MGBI**	**Proposed**	**Rank**
Kodak	36.491	38.494	40.334	39.825	40.984	39.990	**41.791**	41.118	2
McMaster	37.313	38.177	38.449	38.268	37.950	**39.166**	38.062	39.125	2
Condat	36.021	37.317	37.568	37.739	37.095	38.044	37.456	**38.221**	1
Mean	36.609	37.996	38.784	38.611	38.677	39.067	39.103	**39.488**	1

Table I and II highlight the green content interpolation over all three image sets. In the Kodak set, it is observed that the proposed technique has the best SSIM performance and is second after MGBI [12] in the MSE metric. This is from an observation of the Kodak set mean value data and the rank system. In the McMaster-IMAX set, the proposed method comes second after MDWI [10] in the MSE analysis. In SSIM for this image set, the proposed method is perceived to perform poorly being ranked 5th but this is because the small variation change - the difference between the 1st and 6th ranks is 0.0043. The value of 0.9691 documented for the proposed method is still a high SSIM metric. Finally, in the Condat set, the proposed metric outperforms all the other methods in both test measures. Overall, the proposed technique shows very robust demosaicking performance in the green plane using the pentomino generated path concept. Table III shows the average CPSNR performance of all the tested methods over the three image sets. The proposed method performs second best in the Kodak and McMaster-IMAX, after MGBI [12] and MDWI [10] respectively. It is, however, the best performing algorithm in the Condat set and overall. This is observed from the ranking system in the table. This shows the proposed method is highly robust when operating over all the color planes and acting as a full demosaicking algorithm. It exceeds the second best performing algorithm by 0.385 dB from the mean value data.

VI. CONCLUSION

A novel gradient spatial-based demosaicking method has been proposed. The proposed technique's concept of pentomino inspired path has allowed it to be comparable and in some cases even exceed current existing methods. An objective verification of this was provided by the use of three well understood performance criteria over three commonly documented image sets. Furthermore, the contribution of prioritizing different planes in interpolation is a transferable concept that can be incorporated in other existing techniques. These concepts shall form the basis of future work.

REFERENCES

[1] R. Ramanath, W. E. Snyder, Y. Yoo, and M. S. Drew, "Color image processing pipeline," *IEEE Signal Process. Mag.*, vol. 22, pp. 34–43, Jan. 2005.

[2] R. Hunt, *The Reproduction of Color*, 6th ed. West Sussex, England: John Wiley and Sons Inc., 2004.

[3] R. Lukac, Ed., *Single Sensor Imaging Methods*. Boca Raton, FL: CRC Press, 2009.

[4] B. E. Bayer, "Color imaging array," U.S. Patent 3 971 065, Jul. 20, 1976.

[5] D. Menon and G. Calvagno, "Color image demosaicking: An overview," *Signal Processing: Image Communication (Elsevier)*, vol. 26, 2011.

[6] X. Li, B. K. Gunturk, and L. Zhang, "Image demosaicking: A systematic survey," in *Proc. SPIE 6822, Visual Communications and Image Processing 2008, SPIE-IS&T Electronic Imaging*, San Jose, CA, Jan. 2008.

[7] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demosaicking: Color filter array interpolation," *IEEE Signal Process. Mag.*, vol. 22, pp. 44–54, Jan. 2005.

[8] H. S. Malvar, L. W. He, and R. Cutler, "High-quality linear interpolation for demosaicking of bayer-patterned color images," in *Proc. IEEE 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, Montreal, Quebec, Canada, May 2004.

[9] W. Jin, "Improved color interpolation method based on bayer image," in *Proc. SPIE 8420, 6th International Symposium on Advanced Optical Manufacturing and Testing Technologies*, 2012.

[10] X. Chen, G. Jeon, J. Jeong, and L. He, "Multi-directional weighted interpolation and refinement method for bayer pattern cfa demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, submitted for publication in 2014.

[11] W. T. Huang, W. J. Chen, and S. C. Tai, "A sharp edge-preserving joint color demosaicking and zooming algorithm using integrated gradients and an iterative back-projection technique," *Digital Signal Processing (Elsevier)*, vol. 27, pp. 79–94, 2014.

[12] I. Pekkucuksen and Y. Altunbasak, "Multiscale gradients-based color filter array interpolation," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 157–165, Jan. 2013.

[13] D.-C. Sung and H.-W. Tsao, "A gradient based edge sensing scheme for color filter array demosaicking," in *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, Tokyo, Japan, Oct. 2013.

[14] I. Pekkucuksen and Y. Altunbasak, "Edge strength filter based color filter array interpolation," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 393–397, Jan. 2012.

[15] S. W. Golomb, *Polyominoes: Puzzles, Patterns, Problems and Packings*, 2nd ed. Princeton, New Jersey: Princeton University Press, 1994.

[16] X. Chen, G. Jeon, and J. Jeong, "Voting based directional method and its application to still color image demosaicking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 2, pp. 255–262, Feb. 2014.

[17] (2015). [Online]. Available: http://r0k.us/graphics/kodak/

[18] (2015). [Online]. Available: http://www4.comp.polyu.edu.hk/~cslzhang/

[19] (2015). [Online]. Available: http://www.gipsa-lab.grenoble-inp.fr/~laurent.condat/imagebase.html

G.5 Fifth Publication – IEEE AFRICON 2017

An Ordinal Direction Driven Gradient-Based RGBW CFA Demosaicking Technique using a Bayerisation Process and Polyomino Theory

Kinyua Wachira and Elijah Mwangi
School of Engineering,
University of Nairobi,
P.O. Box 30197-00100, Nairobi
Email: kinyua.wachira@students.uonbi.ac.ke

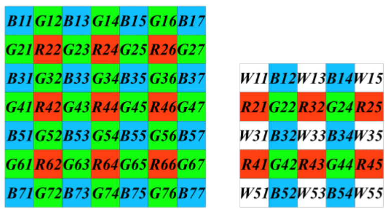
Gwanggil Jeon
Department of Embedded Systems Engineering,
Incheon National University,
Songdo-dong 119, Yeonsu-gu, Incheon, Republic of Korea
Email: gjeon@inu.ac.kr

***Abstract*—Recent studies have shown that panchromatic colour filter arrays possess better light intensity properties than the more prevalent Bayer array. This is an attractive property in integrated camera design. Demosaicking algorithms, however, focus primarily on Bayer-based cameras because of their simple demosaicking process. This paper presents a novel algorithm for use in the RGBW class of panchromatic arrays. It encodes light intensity information in a bayerisation conversion process and uses polyomino theory to define robust reconstruction interpolation weights. Performance of the proposed algorithm is assessed with established methods using two objective and commonly documented assessment measures (SSIM and CPSNR) over three (Kodak, USC-SIPI and Custom) image sets. An appreciable improvement is noted from MATLAB simulation, averaging 3dB above established techniques.**

***Keywords*—bayerisation, chroma, colour filter array (CFA), luma, ordinal direction, panchromatic, polyomino combinatorial geometry, quincunx, RGBW array**

I. Introduction

In recent years, a significant uptake in visual communication has been observed. Video and colour images have become widespread information descriptors and digital still cameras (DSCs) are integrated in most communication devices [1]. Human vision has two modes: an achromatic (colourless) mode that dominates in low light situations and a trichromatic (three base-colour) mode that is used under normal light conditions. The trichromatic mode is responsible for colour vision and results when incident light passes through three different base pigment photo-receptors in the retina. These base colours are recombined in the human visual system to produce colour images. When mimicking this process, cameras use a single sensor augmented with a colour filter array (CFA). The array sub-samples colour content in pixel points of the image scene before passing it to the sensor for recording. A colour interpolation process termed demosaicking is then performed in software along with other image enhancement techniques to complete reconstruction. This simplifies the hardware needed to sample all three base-colours and ensures camera robustness and affordability. The colour quality of an image is predominantly determined by the choice of CFA and the ability of the demosaicking algorithm. The Bayer CFA [2] is the most common type. It samples the red, green and blue colour wavelengths that roughly correspond to the retinal base pigments and gives good colour (chroma) approximation. The rigid sub-sampling, however, leads to a reduction in recorded light intensity (luma) characteristics [3]. In low resolution integrated DSCs, this leads to severe intensity loss and image reconstruction inaccuracies. A recent class of CFAs termed panchromatic [4] records both intensity and colour content by having some pixel points absorb full incident white light. This balanced luma-chroma characteristic makes sensor data more representative of the actual scene. Both CFA types are shown in Figure 1.


B11 G12 B13 G14 B15 G16 B17
G21 R22 G23 R24 G25 R26 G27
B31 G32 B33 G34 B35 G36 B37
G41 R42 G43 R44 G45 R46 G47
B51 G52 B53 G54 B55 G56 B57
G61 R62 G63 R64 G65 R66 G67
B71 G72 B73 G74 B75 G76 B77
W11 B12 W13 B14 W15
R21 G22 R32 G24 R25
W31 B32 W33 B34 W35
R41 G42 R43 G44 R45
W51 B52 W53 B54 W55

(a) 7-by-7 Bayer grid (b) 5-by-5 RGBW grid

Fig. 1. Bayer and RGBW Panchromatic Colour Filter Arrays

Current literature classifies demosaicking algorithms into spatial-heuristic, spectral-heuristic and optimisation methods [5]. This work is biased towards spatial-heuristic type because they have a low computational load and use direct sensor data. This provides real time processing advantages needed in integrated cameras. Early spatial methods were bilinear and constant hue-based interpolation [6]. These algorithms suffered from edge preservation limitations that led to the creation of edge-based spatial methods [7], [8]. Edge techniques query neighbourhood pixels to determine what direction to interpolate, thereby preserving edge information in the scene. Modern spatial techniques extend edge preservation by determining gradient descriptors to weight the interpolation process enhancing detail even further [9]–[11]. Gradient-based spatial techniques provide a high degree of calibration depending on how the gradient is generated. They can be enhanced by increasing interpolation directions or by using higher order gradients [12].

The work presented makes a contribution by creating a novel spatial-heuristic gradient-based demosaicking algorithm for the RGBW panchromatic CFA [13]. The remainder of this paper is divided as follows: Section II explains the bayerisation process and gradient-based demosaicking. Section III highlights interpolation weight generation by polyomino geometry that exploits the ordinal directions in the green plane. The proposed technique is presented in Section IV. Section V outlines the simulation and testing process before presenting the results found. Final conclusions drawn are provided in Section VI.

II. The Bayerisation Process and Gradient-Based Demosaicking

Most of the established techniques presented in Section I are designed for the Bayer CFA with one colour-per-pixel point regime in all pixels. Demosaicking in the RGBW panchromatic array shown in Figure 1(b) is complicated by the white pixel point regions that contain three colours-per-pixel. These sections allow the full light spectrum rather than distinct wavelengths to pass as in the case of the red, blue and green pixel regions making colour decomposition difficult.

A. The Bayerisation Process

RGBW demosaicking can be performed by converting white pixel points to green equivalents using neighbourhood averaging. Considering pixel $W33$ in Figure 1(b), the equivalent $\widehat{G}_{W33}$ is generated using (1). The light intensity is considered separate from demosaicking and handled at a later stage.

$$\widehat{G}_{W33} = 0.25(G22 + G24 + G42 + G44) \tag{1}$$

The averaging process above can be expressed as a filter, h_{basic}, shown in (2).

$$h_{basic} = \begin{bmatrix} 2 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & 2 \end{bmatrix} /8 \tag{2}$$

An alternative method was proposed by Chen et. al [14]. The RGBW CFA is converted to an equivalent Bayer CFA while simultaneously encoding the light intensity information within. This is achieved by using a modified averaging filter, h_{mod}, shown in (3). The authors have termed this conversion and encoding a bayerisation process.

$$h_{mod} = \begin{bmatrix} 0 & 0 & -3/2 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ -3/2 & 0 & 6 & 0 & -3/2 \\ 0 & 2 & 0 & 2 & 0 \\ 0 & 0 & -3/2 & 0 & 0 \end{bmatrix} /8 \tag{3}$$

The white pixel coefficients of h_{mod} introduce a light intensity term ρ to (1). If pixel $W33$ in Figure 1(b) is brighter or darker than its white pixel neighbours, this intensity is factored to the green pixel average. If the white pixels in the region are of equal value, h_{mod} reduces to h_{basic}. This is expressed in (4). Bayerisation conversion is attractive as it combines colour demosaicking and light intensity in a single step. It is also a satisfactory assumption considering ρ is much smaller than the green estimates in practice to adversely affect approximation. The proposed algorithm uses this conversion as its first step.

$$\widehat{G}_{W33} = 0.25(G22 + G24 + G42 + G44 + \rho) \tag{4}$$

B. Gradient Based Demosaicking

In a Bayer or converted equivalent array, gradient-based demosaicking is a four step exercise expressed from (5) to (8). It involves generating sufficient heuristic estimates that can predict the missing information in the three colour planes and allow accurate reconstruction. Consider a general pixel, X_P; where X is the desired colour plane, N is a separate plane, P is the pixel location and k is the interpolation direction.

1) Initial estimates of the desired pixel colour are found,

$$\widetilde{X}_P^k = (X_{P-1}^k) + s_1(N_P^k - N_{P-2}^k) \tag{5}$$

where s_1 is a fraction minimising neighbour effects.

2) Directional gradients are derived from the three colour planes based on a sum of absolute differences,

$$\Phi_P^k = \Phi_{Red,P}^k + \Phi_{Green,P}^k + \Phi_{Blue,P}^k + s_2 \tag{6}$$

where Φ indicates a sum of absolute differences and s_2 is a corrective constant.

3) The gradients are converted to weighting factors,

$$\phi_P^k = \frac{1}{\Phi_P^k} \tag{7}$$

4) Finally, the missing colour value in the general pixel X_P is found from the calculated estimates and weights,

$$\widehat{X}_P = \frac{\sum_k \phi_P^k \widetilde{X}_P^k}{\sum_k \phi_P^k} \tag{8}$$

Variations to this four step process exist in different gradient based algorithms but they follow the above trend.

III. New Concepts

Effective demosaicking is dependent on choosing pixels to accurately estimate missing colour content with minimal error. In the case of a gradient-based algorithm, the chosen pixels should be sufficient to generate good heuristic descriptors. If the descriptors are generated with too few pixels, they lead to an overly smooth image with blur effects in regions of fine detail. If too many pixels are used to form the descriptors, the outlier information from far away pixels introduces inaccuracies in reconstruction. It also increases the computational complexity of the algorithm. This work introduces an ordinal direction driven exploitation of the quincuncial green plane seen in the Bayer and converted equivalent arrays. It also introduces the use of $5tris$ polyomino blocks that create sufficient gradient heuristic descriptors to achieve optimal reconstruction. It should be noted that these ideas are applied in the proposed algorithm after the bayerisation process and form part of the gradient-based demosaicking technique. As such, they are considered in a Bayer or converted equivalent environment.

A. The Ordinal Nature of the Green Plane

The green plane is the most populous in a Bayer CFA. Proper demosaicking here improves overall image reconstruction. Established gradient-based methods often set a cardinal direction interpolation. However, a visual inspection reveals the green plane of the Bayer CFA has more pixels lying in ordinal directions than in the cardinal for the same size of bounding window due to the green plane quincunx. For any window of size $(2h+1)$ pixels, there are more ordinal-directed pixels present than cardinal ones. This is shown in Figure 2 with the N cardinal and NW ordinal directions.

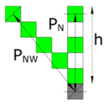


Fig. 2. Paths in the North and North-West directions

This leads to higher, more compact pixel packing along ordinal directions compared to cardinal ones. This is shown in Figure 3 for two different grid sizes, where P_C are the cardinal-directed pixels and P_O are ordinal-directed. This observation motivated the authors to propose an ordinal-only interpolation process.

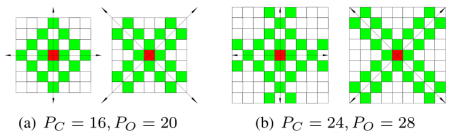


Fig. 3. Pixel packing comparison

B. $5tris$ Pentomino Blocks

The Bayer CFA is a tessellation with sets of square blocks fitting together to form a whole. This idea of smaller blocks constructing larger ones is the main design feature of recreational polyomino geometry games such as Tetris and its variants. The proposed algorithm uses a Tetris variant called $5tris$ developed by Zhen [15] with 5-square interconnected blocks called pentominoes. From polyomino theory, the total number of one-sided pentominoes (assuming that rotations are not considered unique) is 18. These are illustrated in Figure 4. Blocks that form chirals (non-superposable mirror images) are denoted by a letter with a prime symbol.

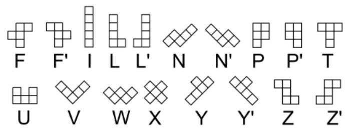


Fig. 4. The 18 one-sided pentomino blocks using Golomb naming [16]

Each pentomino block of Figure 4 is used to generate two paths. These paths will be used to select the difference terms for finding gradients in the proposed algorithm. These paths are shown in Figure 5. Chirals are ignored for ease of analysis. The following rules were set as guidelines to generate unique paths:

1) Two blocks forming a chiral pair will have their paths forming a chiral pair as well.
2) The pixel of interest must share a vertex with an edge square of the pentomino construct.

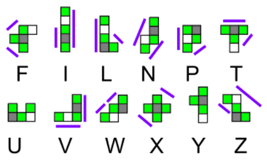


Fig. 5. The difference term paths

Considering ordinal directions and based on the paths generated, pentominoes N, T, W, X or Z are potential candidates for use. This is because I, L, P and V have cardinal paths; F has paths in two different ordinal directions thus it has no preferred direction and the U and Y pentominoes have no new paths that cannot be generated from F, P and T. The authors selected the N,W and Z pentominoes as gradient selection paths due to their significant ordinal bias.

C. Additional Contributions

As the Bayer CFA is made up of three colour planes as shown in Figure 6, the proposed algorithm encodes this distinction by introducing variable plane factors.

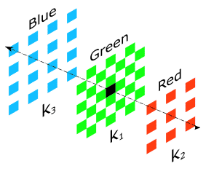


Fig. 6. The decomposed Bayer CFA

The black pixel point in Figure 6 is in the green plane but has a record of red colour data. The green plane has the largest impact (rank 1) as it is the physical location of the pixel point. The red plane follows in importance (rank 2) as it contains actual recorded colour data at that pixel point. The blue plane offers no contribution in location or colour and is the least important (rank 3). The authors denoted this impact using three factors $k_1 = 1$, $k_2 = 0.8$ and $k_3 = 0.7$ where the subscript denotes the plane rank. These values were empirically determined by the principal author in a separate study on corrective terms [9].

IV. Proposed Technique

The general directional gradient equation has contributions from all three colour planes:

$$\Phi_P^k = \Phi_{G,P}^k + \Phi_{B,P}^k + \Phi_{R,P}^k + c \tag{9}$$

where ψ indicates a sum of differences and c is a corrective constant to prevent an indeterminate gradient.

A. Green Lattice Reconstruction

Consider determining the green colour present in the pixel marked $R44$ in Figure 1(a). Initial estimates are established for the four ordinal directions:

$$\begin{aligned}
\tilde{G}_{R44}^{NW} &= 0.5(G34 + G43) + (k_2 k_3)(R44 - R22)\\
\tilde{G}_{R44}^{SW} &= 0.5(G54 + G43) + (k_2 k_3)(R44 - R62)\\
\tilde{G}_{R44}^{SE} &= 0.5(G45 + G54) + (k_2 k_3)(R44 - R66)\\
\tilde{G}_{R44}^{NE} &= 0.5(G45 + G34) + (k_2 k_3)(R44 - R26) \qquad (10)
\end{aligned}$$

The gradient in each ordinal direction is determined using all three colour planes. Using the South West as an example:

$$\begin{aligned}
\Phi_{R44}^{SW} &= \Phi_{G,R44}^{SW} + \Phi_{B,R44}^{SW} + \Phi_{R,R44}^{SW} + c\\
\Phi_{G,R44}^{SW} &= |path_N| + |path_W| + |path_Z|\\
\Phi_{R,R44}^{SW} &= k_2(|R44 - R62|)\\
\Phi_{B,R44}^{SW} &= k_3(|B53 - B71|) \qquad (11)
\end{aligned}$$

where $path_N = (|G43 - G52| + |G45 - G52|)$, $path_W = (|G34 - G43| + |G43 - G52|)$ and $path_Z = (|G54 - G63| + |G45 - G63|)$ shown in Figure 7. The variable plane factors are k_2 and k_3. Φ_{B44}^{NW}, Φ_{B44}^{NE} and Φ_{B44}^{SE} are generated in a similar manner by rotating the pentomino paths in Figure 7 counter-clockwise 90°, 180° and 270° respectively.

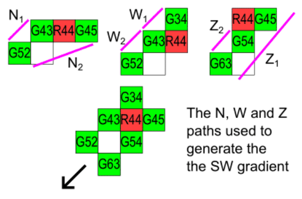


Fig. 7. The South West paths from R44 using N,W and Z pentominoes

The inverse of the gradients, Φ, provides the weighting factors ϕ. Polling maps were used to determine which of the final equations to invoke. Equations 12a and 12b are used if interpolation is predominant along a particular diagonal. Otherwise, equation 12c is employed. A similar treatment is used to determine green colour content in blue pixel locations.

$$\hat{G}_{R44} = \frac{\sum_{k\in\{NW,SE\}}[\phi_{R44}^k \tilde{G}_{R44}^k]}{\sum_{n\in\{NW,SE\}} \phi_{R44}^k} \tag{12a}$$

$$\hat{G}_{R44} = \frac{\sum_{k\in\{SW,NE\}}[\phi_{R44}^k \tilde{G}_{R44}^k]}{\sum_{k\in\{SW,NE\}} \phi_{R44}^k} \tag{12b}$$

$$\hat{G}_{R44} = \frac{\sum_{k\in\{NW,SW,SE,NE\}}[\phi_{R44}^k \tilde{G}_{R44}^k]}{\sum_{k\in\{NW,SW,SE,NE\}} \phi_{R44}^k} \tag{12c}$$

B. Blue and Red Lattice Reconstruction

Blue or red lattice reconstruction involves first establishing content in an opposing plane (red in blue plane or vice versa) and then finding missing red/blue content within the green plane.

1) Opposing Plane: When establishing blue content in $R44$ say, it is observed that at this stage the green content in each pixel has been determined. A difference-based solution is used where ordinal estimates are taken:

$$\begin{aligned}
\tilde{\gamma}_{R44}^{SW} &= B53 - \hat{G}_{B53}, \tilde{\gamma}_{R44}^{NW} = B33 - \hat{G}_{B33}\\
\tilde{\gamma}_{R44}^{NE} &= B35 - \hat{G}_{B35}, \tilde{\gamma}_{R44}^{SE} = B55 - \hat{G}_{B55} \qquad (13)
\end{aligned}$$

Gradients are found using a variant of equation 9 where both calculated and actual green plane content is used :

$$\Phi^k = \Phi_{G,calc}^k + \Phi_{G,act}^k + \Phi_B^k + c \tag{14}$$

where k denotes ordinal directions. Taking the NW direction, the components of equation 14 are:

$$\begin{aligned}
\Phi_{G,calc}^{NW} &= (|\hat{G}_{R44} - \hat{G}_{B33}| + |\hat{G}_{B33} - \hat{G}_{R22}|)\\
\Phi_{G,act}^{NW} &= (|G43 - G32| + |G34 - G23|)\\
\Phi_B^{NW} &= k_2(|B33 - B55|) \qquad (15)
\end{aligned}$$

The process is repeated for Φ_{R44}^{NE}, Φ_{R44}^{SE} and Φ_{R44}^{SW}. The weights, ϕ are found and the final value is:

$$\hat{B}_{R44} = \hat{G}_{R44} + \frac{\sum_{k\in\{NW,SW,SE,NE\}}[\phi_{R44}^k \tilde{\gamma}_{R44}^k]}{\sum_{k\in\{NW,SW,SE,NE\}} \phi_{R44}^k} \tag{16}$$

2) Within Green Plane: Consider trying to find red content in the green plane. The procedure is similar to interpolation during green lattice reconstruction: obtain initial ordinal estimates, determine the weights and establish the final value. Equations 17a, 17b and 17c demonstrate the work flow for the NW direction of the pixel $G45$. The same treatment is applied to determine blue content.

$$\tilde{R}_{G45}^{NW} = 0.25(\hat{R}_{B35} + R24 + \hat{R}_{B33} + R44) \tag{17a}$$

$$
\begin{aligned}
\Phi^{NW}_{G45} &= \Phi^{NW}_{R,G45} + \Phi^{NW}_{G,G45} + c \\
\Phi^{NW}_{R,G45} &= (|\widehat{R}_{B5} - R24| + |\widehat{R}_{B33} - R44|) \\
\Phi^{NW}_{G,G45} &= (|G45 - G34| + |G34 - G23|)
\end{aligned} \tag{17b}
$$

$$
\widehat{R}_{G45} = \frac{\sum_{k\in\{NW,SW,SE,NE\}}[\phi^k_{G45}\widetilde{R}^k_{G45}]}{\sum_{k\in\{NW,SW,SE,NE\}}\phi^k_{G45}} \tag{17c}
$$

V. Simulation Results And Discussion

A. Choice of Image Sets and Quality Metrics

Twenty four test images were randomly selected from two commonly reported databases readily available online: USC-SIPI and Kodak [17], [18]. Eight additional custom images were also used. These are shown in Figure 8. For objective assessment, the Colour Peak Signal-to-Noise Ratio (CPSNR) was used to measure individual pixel reconstruction acuity and the Structural Similarity Index (SSIM) indicated the strength of reconstruction of objects in the image.

(a) USC-SIPI Set (1-12)

(b) Kodak Set (1-12)

(c) Custom Set (1-8)

Fig. 8. Test Images Used

B. Simulation Methodology

Simulation of the demosaicking process for purposes of analysis was performed using MATLAB R2015b running on an Intel(R) Core(TM) i5-6200 CPU @ 2.39 GHz processor. The proposed method was compared with six established spatial algorithms: Four methods were state of the art gradient based Bayer methods: MHC [8], Wang [19], ESFBI [11] and MGBI [10]. The remaining two are RGBW techniques that function without encoding luma information in a bayerisation process: ACR and EDCR [20].

C. Experimental Results

For each image set, every image was decomposed to a Bayer CFA equivalent (or RGBW CFA equivalent in the case of the ACR and EDCR methods). The image was passed to each of the algorithms in turn and the CPSNR and SSIM values were determined and recorded in Tables I and II respectively.

TABLE I
CPSNR Metric Evaluation Over Selected Image Sets

Image	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Proposed**	*Rank*
				USC − SIPI Set				
01	29.100	28.787	34.242	34.252	34.306	33.438	34.626	
02	37.534	36.438	39.380	39.186	41.906	38.834	42.349	
03	33.030	32.514	36.915	36.607	38.181	37.571	37.856	
04	37.704	37.499	40.454	39.958	40.826	41.458	42.264	
05	37.017	36.486	39.791	39.732	38.620	38.456	40.121	
06	36.697	37.693	39.867	39.464	38.833	37.408	40.801	
07	31.835	31.815	35.943	35.824	36.217	35.284	36.363	
08	34.087	34.204	37.565	37.441	37.314	37.514	38.048	
09	33.298	33.322	37.354	37.100	37.223	37.659	37.792	
10	32.685	32.015	36.678	36.719	36.222	33.783	36.563	
11	34.085	34.668	37.773	37.487	36.477	37.004	38.500	
12	25.197	24.616	32.273	32.390	31.781	32.245	32.273	
Mean	33.322	33.120	37.279	37.114	37.234	36.636	**38.020**	*1*
				Kodak Set				
01	39.388	38.377	41.380	40.495	39.544	38.983	43.254	
02	33.216	31.536	39.699	40.846	34.395	33.790	39.864	
03	29.075	30.629	36.530	38.661	32.669	31.642	40.292	
04	37.962	38.757	41.212	42.337	38.490	38.545	41.547	
05	38.779	39.323	39.583	43.652	38.944	38.447	42.445	
06	29.491	27.729	37.082	38.384	32.758	32.546	37.091	
07	37.498	36.472	41.029	42.437	38.923	39.446	42.065	
08	33.643	35.670	39.070	42.718	35.528	35.165	39.250	
09	34.142	33.549	40.336	41.646	35.822	36.206	39.924	
10	40.946	39.673	42.109	42.169	41.112	41.684	43.351	
11	31.684	29.993	39.254	37.040	35.459	35.435	39.635	
12	38.048	38.115	41.062	41.097	37.948	38.409	41.058	
Mean	35.112	34.750	39.828	**40.911**	36.704	36.573	40.777	*2*
				Custom Set				
01	41.463	41.005	44.620	47.872	40.332	40.521	44.853	
02	38.644	38.943	43.166	44.576	37.796	37.396	41.474	
03	40.149	39.843	43.626	38.601	38.686	37.757	43.078	
04	38.315	37.597	39.831	33.473	36.853	35.995	41.130	
05	38.275	36.967	42.233	39.278	37.317	37.615	41.303	
06	41.497	39.908	42.652	47.696	40.365	39.637	44.836	
07	39.375	39.917	43.693	47.864	38.156	38.118	42.426	
08	40.425	39.491	41.805	43.478	39.158	38.527	42.776	
Mean	39.749	39.189	42.681	42.552	38.564	38.174	**42.712**	*1*

Geometric means of the two objective metrics in each image set were also calculated and recorded in the aforementioned tables. The averages of the CPSNR geometric means over the three image sets are highlighted in Table III.

D. Discussion

A ranking system showing the overall position of the proposed algorithm in relation to the other established methods was introduced to assess algorithm robustness over various image sets. In CPSNR, the proposed technique ranked first in the USC-SIPI and Custom image sets and second in the Kodak set. The average value data in Table III shows the proposed method's CPSNR exceeded all other algorithms by at least 0.311 dB. This strong performance is in contrast with the best performing established technique that is the MGBI method that ranks first in the Kodak set, but third in the Custom set and fourth in USC-SIPI images. A similar trend was observed in the SSIM index values. The proposed method ranked either first or second in the three image sets. Its worst performance

TABLE II
SSIM METRIC EVALUATION RESULTS OVER SELECTED IMAGE SETS

Image	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Proposed**	*Rank*
USC − SIPI Set								
01	0.933	0.930	0.934	0.930	0.937	0.930	0.932	
02	0.971	0.966	0.964	0.964	0.970	0.959	0.967	
03	0.942	0.937	0.929	0.911	0.953	0.951	0.936	
04	0.973	0.968	0.963	0.954	0.969	0.953	0.967	
05	0.970	0.972	0.966	0.957	0.969	0.966	0.963	
06	0.955	0.965	0.955	0.947	0.967	0.963	0.961	
07	0.934	0.932	0.931	0.927	0.940	0.936	0.936	
08	0.952	0.950	0.952	0.946	0.954	0.949	0.950	
09	0.950	0.959	0.945	0.950	0.954	0.951	0.964	
10	0.977	0.974	0.970	0.959	0.972	0.957	0.965	
11	0.950	0.955	0.973	0.967	0.950	0.967	0.978	
12	0.911	0.905	0.908	0.912	0.885	0.856	0.907	
Mean	0.951	0.951	0.949	0.944	0.951	0.944	**0.952**	*1*
Kodak Set								
01	0.982	0.980	0.984	0.983	0.968	0.964	0.984	
02	0.982	0.981	0.985	0.984	0.954	0.942	0.982	
03	0.972	0.978	0.980	0.980	0.923	0.907	0.977	
04	0.980	0.981	0.982	0.979	0.965	0.958	0.980	
05	0.982	0.984	0.985	0.985	0.958	0.953	0.984	
06	0.964	0.956	0.977	0.975	0.899	0.883	0.969	
07	0.984	0.986	0.986	0.980	0.971	0.921	0.982	
08	0.974	0.976	0.979	0.980	0.942	0.929	0.975	
09	0.976	0.975	0.979	0.977	0.952	0.941	0.975	
10	0.985	0.985	0.984	0.973	0.980	0.965	0.983	
11	0.978	0.977	0.982	0.947	0.948	0.937	0.979	
12	0.975	0.978	0.975	0.971	0.943	0.921	0.968	
Mean	0.978	0.978	**0.981**	0.976	0.950	0.935	0.978	*2*
Custom Set								
1	0.991	0.993	0.992	0.989	0.992	0.985	0.990	
2	0.994	0.996	0.994	0.987	0.993	0.988	0.993	
3	0.992	0.995	0.984	0.950	0.991	0.988	0.990	
4	0.990	0.991	0.922	0.860	0.981	0.985	0.988	
5	0.993	0.994	0.988	0.947	0.991	0.974	0.992	
6	0.996	0.997	0.996	0.994	0.996	0.995	0.996	
7	0.994	0.966	0.994	0.992	0.993	0.986	0.994	
8	0.998	0.998	0.998	0.951	0.998	0.997	0.998	
Mean	**0.994**	0.991	0.983	0.958	0.992	0.987	0.993	*2*

TABLE III
AVERAGE CPSNR EVALUATION VALUES OVER SEVERAL DEMOSAICKING SCHEMES IN (DB) IN SELECTED IMAGE SETS

Image Set	**MHC**	**Wang**	**ESFBI**	**MGBI**	**ACR**	**EDCR**	**Proposed**	*Rank*
USC-SIPI	33.322	33.120	37.279	37.114	37.234	36.636	**38.020**	*1*
Kodak	35.112	34.750	39.828	**40.911**	36.704	36.573	40.777	*2*
Custom	39.749	39.189	42.681	42.552	38.564	38.174	**42.712**	*1*
Average	35.961	35.687	39.929	40.192	37.501	37.128	**40.503**	*1*

was in the Kodak set where it had a SSIM geometric mean of 0.978 compared to the ESFBI algorithm with 0.981.

Comparing the proposed method with the two established RGBW based algorithms, it was observed that the proposed technique had superior CPSNR and SSIM characteristics. In particular, averaging a CPSNR improvement of more than 3dB over the ACR and EDCR methods shown in Table III.

VI. CONCLUSION

A novel RGBW panchromatic array gradient-based spatial demosaicking algorithm was proposed. It works on a bayerisation process coupled with a gradient weight generation system that exploits the quincunical green plane and uses polyomino blocks. It was found to exhibit robust performance characteristics over standard and custom image sets. Additional concepts introduced in this paper such as variable plane factors are transferable and can be used to improve existing published methods. It was also determined that the encoding light intensity was in fact beneficial when compared with methods that performed this later in processing. The authors feel that this would be an area worth investigating in future work.

REFERENCES

[1] J. Nakamura, *Image Sensors and Signal Processing for Digital Still Cameras*. CRC Press, Apr. 2016, google-Books-ID: aaTyk2USX1MC.

[2] B. E. Bayer, "Color Imaging Array," U.S. Patent 3 971 065, Jul., 1976.

[3] J. Li, C. Bai, Z. Lin, and J. Yu, "Automatic Design of High-Sensitivity Color Filter Arrays With Panchromatic Pixels," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 870–883, Feb. 2017.

[4] J. Wang, C. Zhang, and P. Hao, "New color filter arrays of high light sensitivity and high demosaicking performance," in *2011 18th IEEE International Conference on Image Processing*, Sep. 2011, pp. 3153–3156.

[5] Y. Li, P. Hao, Z. Lin, Y. Li, P. Hao, and Z. Lin, "Color filter arrays: representation and analysis," Queen Mary University of London, Tech. Rep., May 2008. [Online]. Available: https://pdfs.semanticscholar.org/8dce/b6232c569918ba88657acbfbd3a040be0132.pdf

[6] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demosaicking: Color Filter Array Interpolation," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 44–54, Jan. 2005.

[7] W. Lee, S. Lee, and J. Kim, "Cost-efffective color filter array demosaicing using spatial correlation," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 547–554, May 2006.

[8] H. S. Malvar, L.-w. He, and R. Cutler, "High-quality linear interpolation for demosaicing of Bayer-patterned color images," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, May 2004, pp. 485–488.

[9] K. Wachira, "Corrective term usage in the improvement of gradient-based bayer CFA demosaicking algorithms," in *EUROCON 2015 - International Conference on Computer as a Tool (EUROCON), IEEE*, Sep. 2015, pp. 1–6.

[10] I. Pekkucuksen and Y. Altunbasak, "Multiscale Gradients-Based Color Filter Array Interpolation," *IEEE*, vol. 22, no. 1, pp. 157–165, Jan. 2013.

[11] ——, "Edge Strength Filter Based Color Filter Array Interpolation," *IEEE*, vol. 21, no. 1, pp. 393–397, Jan. 2012.

[12] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Residual interpolation for color image demosaicking," in *2013 20th IEEE International Conference on Image Processing (ICIP)*, Sep. 2013, pp. 2304–2308.

[13] E. B. Gindele and A. C. Gallagher, "Sparsely sampled image sensing device with color and luminance photosites," U.S. Patent US6 476 865 B1, Nov., 2002. [Online]. Available: http://www.google.com/patents/US6476865

[14] X. Chen, L. He, J. Tang, and Y. S. Lee, "A Low-Complexity Interpolation Method for Single-Sensor Camera Imaging with White-RGB Color Filter Array," in *2015 11th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, Nov. 2015, pp. 560–565.

[15] (2017). [Online]. Available: http://www.zhenfavor.org/home/fiveTris.html

[16] S. W. Golomb, *Polyominoes: Puzzles, Patterns, Problems and Packings*, 2nd ed. Princeton, New Jersey: Princeton University Press, 1994.

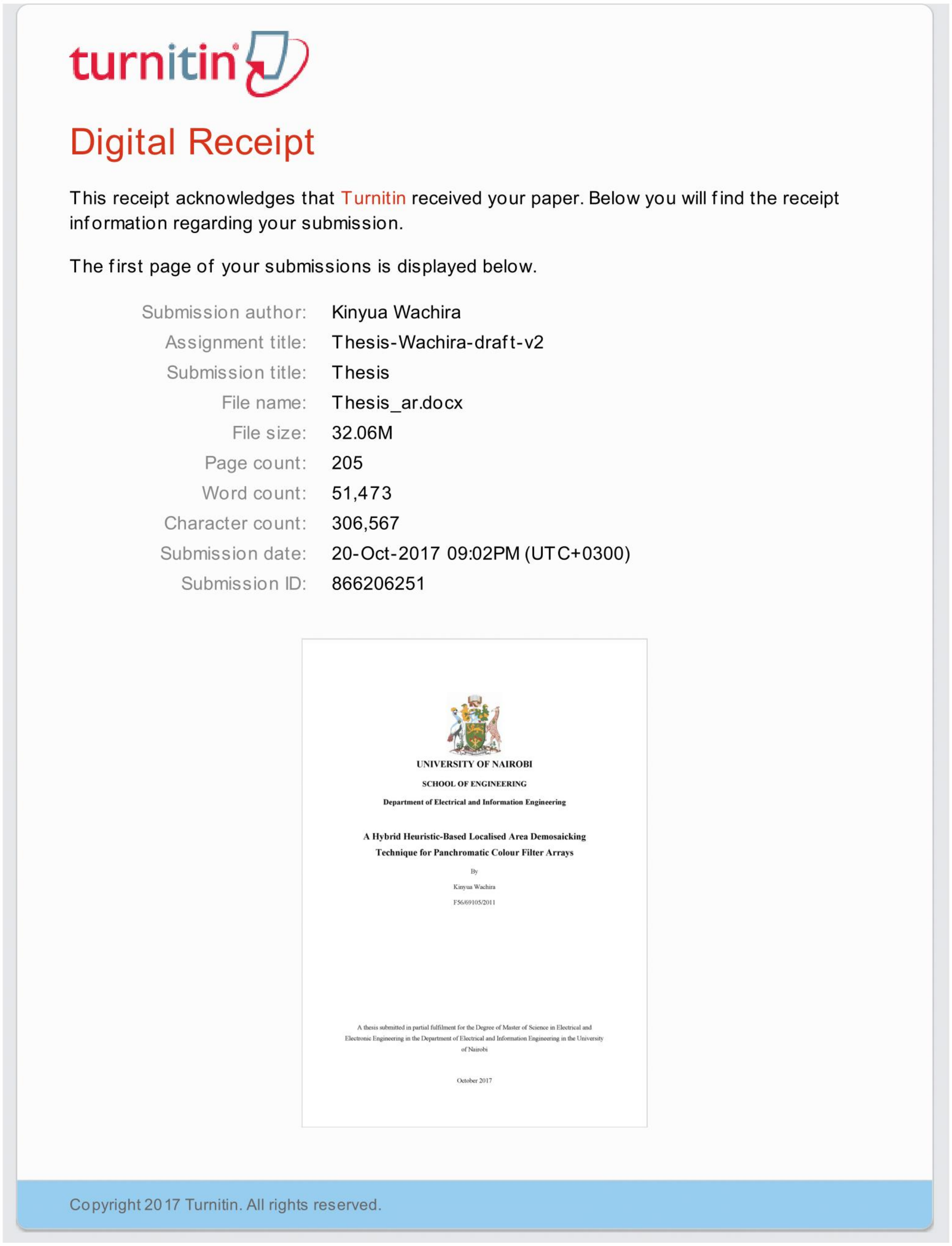
[17] "SIPI Image Database - Misc," jul 2017, [Date Accessed: 2017-07-10]. [Online]. Available: http://sipi.usc.edu/database/database.php?volume=misc

[18] "True Color Kodak Images," jul 2017, [Date Accessed: 2017-07-10]. [Online]. Available: http://r0k.us/graphics/kodak/

[19] W. Jin, "Improved Color Interpolation Method Based On Bayer Image," in *Proc. SPIE 8420, 6th International Symposium on Advanced Optical Manufacturing and Testing Technologies*, 2012.

[20] C.-s. Wang and J.-w. Chong, "An Improved White-RGB Color Filter Array Based CMOS Imaging System for Cell Phones in Low-Light Environments," *IEICE Transactions on Information and Systems*, vol. E97.D, no. 5, pp. 1386–1389, 2014. [Online]. Available: http://jlc.jst.go.jp/DN/JST.JSTAGE/transinf/E97.D.1386?lang=en&from=CrossRef&type=abstract

Appendix H: Results from Turnitin Plagiarism Checker


turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Kinyua Wachira
Assignment title:	Thesis-Wachira-draft-v2
Submission title:	Thesis
File name:	Thesis_ar.docx
File size:	32.06M
Page count:	205
Word count:	51,473
Character count:	306,567
Submission date:	20-Oct-2017 09:02PM (UTC+0300)
Submission ID:	866206251

UNIVERSITY OF NAIROBI

SCHOOL OF ENGINEERING

Department of Electrical and Information Engineering

A Hybrid Heuristic-Based Localised Area Demosaicking Technique for Panchromatic Colour Filter Arrays

By

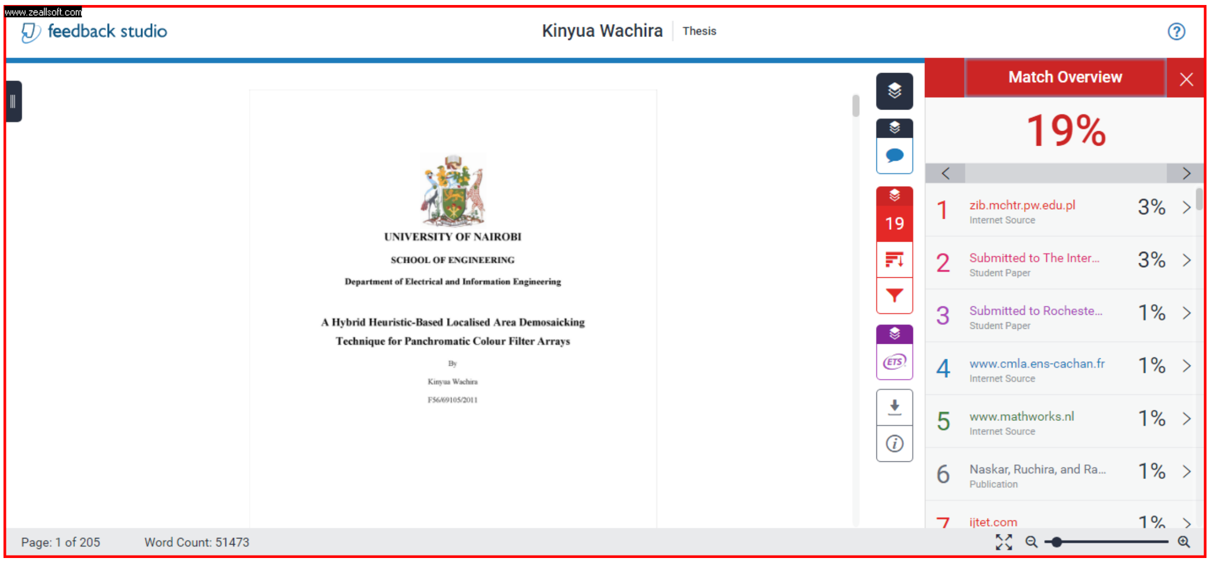
Kinyua Wachira

F56/69105/2011

A thesis submitted in partial fulfilment for the Degree of Master of Science in Electrical and Electronic Engineering in the Department of Electrical and Information Engineering in the University of Nairobi

October 2017

Figure H.1 Turnitin digital receipt


www.zeallsoft.com

feedback studio Kinyua Wachira Thesis

UNIVERSITY OF NAIROBI

SCHOOL OF ENGINEERING

Department of Electrical and Information Engineering

A Hybrid Heuristic-Based Localised Area Demosaicking Technique for Panchromatic Colour Filter Arrays

By

Kinyua Wachira

F56/69105/2011

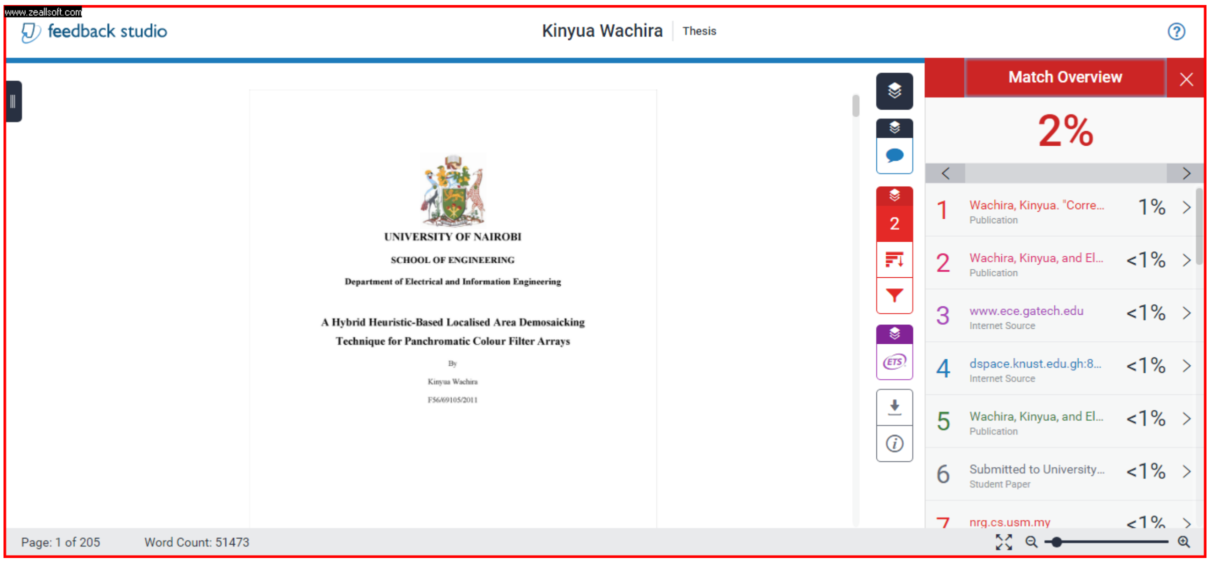
Match Overview

19%

1	zib.mchtr.pw.edu.pl Internet Source	3%
2	Submitted to The Inter... Student Paper	3%
3	Submitted to Rocheste... Student Paper	1%
4	www.cmla.ens-cachan.fr Internet Source	1%
5	www.mathworks.nl Internet Source	1%
6	Naskar, Ruchira, and Ra... Publication	1%
7	ijtet.com	1%

Page: 1 of 205 Word Count: 51473

Figure H.2 Similarity statistics from Turnitin with bibliography and appendices included


www.zeallsoft.com

feedback studio Kinyua Wachira Thesis

UNIVERSITY OF NAIROBI

SCHOOL OF ENGINEERING

Department of Electrical and Information Engineering

A Hybrid Heuristic-Based Localised Area Demosaicking Technique for Panchromatic Colour Filter Arrays

By

Kinyua Wachira

F56/69105/2011

Match Overview

2%

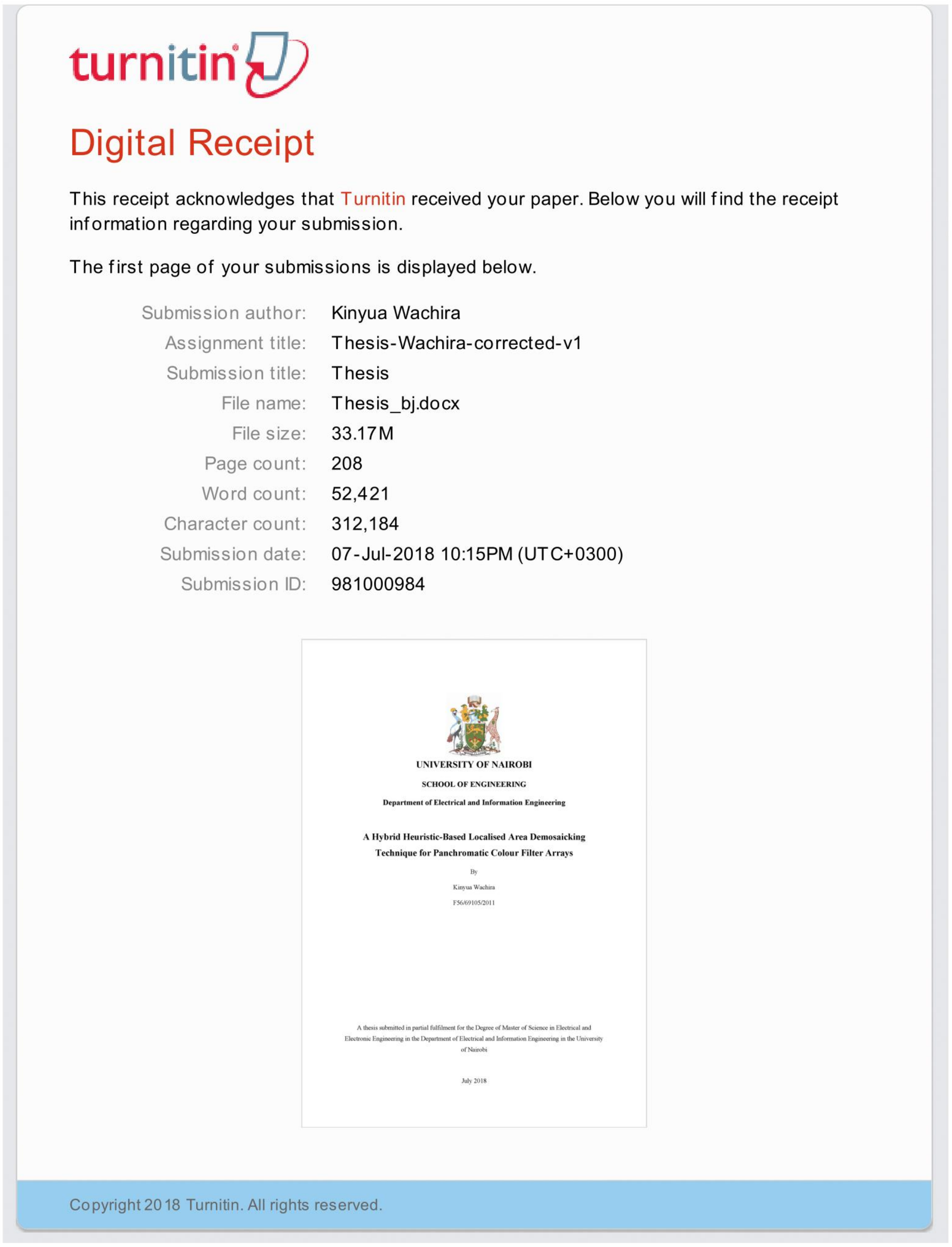
1	Wachira, Kinyua. "Corre... Publication	1%
2	Wachira, Kinyua, and El... Publication	<1%
3	www.ece.gatech.edu Internet Source	<1%
4	dspace.knust.edu.gh:8... Internet Source	<1%
5	Wachira, Kinyua, and El... Publication	<1%
6	Submitted to University... Student Paper	<1%
7	nrg.cs.usm.my	<1%

Page: 1 of 205 Word Count: 51473

Figure H.3 Similarity statistics from Turnitin with bibliography excluded

The updated Turnitin® statistics after implementing corrections are given below:

The updated digital receipt:



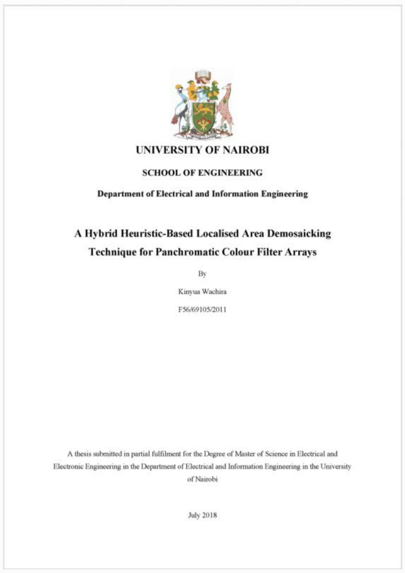
turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Kinyua Wachira
Assignment title:	Thesis-Wachira-corrected-v1
Submission title:	Thesis
File name:	Thesis_bj.docx
File size:	33.17M
Page count:	208
Word count:	52,421
Character count:	312,184
Submission date:	07-Jul-2018 10:15PM (UTC+0300)
Submission ID:	981000984



UNIVERSITY OF NAIROBI

SCHOOL OF ENGINEERING

Department of Electrical and Information Engineering

A Hybrid Heuristic-Based Localised Area Demosaicking Technique for Panchromatic Colour Filter Arrays

By

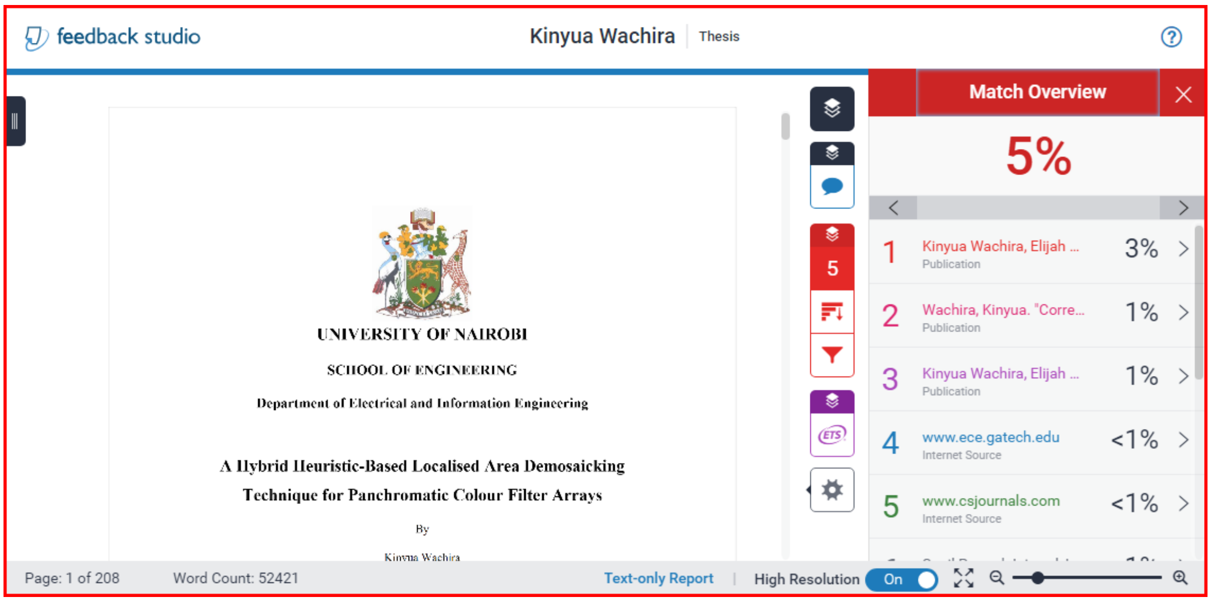
Kinyua Wachira

F56/69105/2011

A thesis submitted in partial fulfilment for the Degree of Master of Science in Electrical and Electronic Engineering in the Department of Electrical and Information Engineering in the University of Nairobi

July 2018

The Turnitin® statistics after implementing corrections (bibliography excluded):


feedback studio Kinyua Wachira | Thesis

UNIVERSITY OF NAIROBI

SCHOOL OF ENGINEERING

Department of Electrical and Information Engineering

A Hybrid Heuristic-Based Localised Area Demosaicking Technique for Panchromatic Colour Filter Arrays

By

Kinyua Wachira

Match Overview

5%

1	Kinyua Wachira, Elijah ... Publication	3%
2	Wachira, Kinyua. "Corre... Publication	1%
3	Kinyua Wachira, Elijah ... Publication	1%
4	www.ece.gatech.edu Internet Source	<1%
5	www.csjournals.com Internet Source	<1%

Page: 1 of 208   Word Count: 52421   Text-only Report | High Resolution On