TECHNICAL UNIVERSITY OF CRETE

Department of Electronics and Computer Engineering

DIPLOMA THESIS

ALGEBRAIC MODELING OF TRANSFORMATIONS FROM BAYER TO RGB IMAGES

TSENGELIDIS SAVVAS

Chania 2006

Acknowledgements

I wish to express my gratitude to my teacher, Michalis Zervakis who has guided me through the years of my studies. His advice has always been enlightening and he has been welcome to answer any questions that I had. He has showed me different ways to approach a research problem and the need to be persistent to accomplish any goal. I would also like to thank my teacher, Euripides Petrakis for his time and ideas whenever needed and for providing me with valuable comments. I am thankful to Professor Mpalas Kontsantinos for his support in my diploma thesis and for agreeing to form my committee.

Besides my advisors, I would like to thank the members of the Digital Image and Signal Processing Laboratory for the environment that they created. It is really helpful to work under these conditions, especially on hard times. To my friends who stood by me and provided me with their endless spirit. Special thanks to Aspa for her understanding, patience and support whenever needed.

Last but not least, I am greatly indebted to my family for being by my side from the beginning till now. They have always believed in me and were supportive to my decisions. Special thanks to my sister, Frosso for the endless miles that she has traveled all these years to visit me. To the newest member of my family Makis who recruited me to the lab and always supported me.

Abstract:

Digital imaging is a field of science that grows rapidly the recent years. Given the fact that in most imaging applications there is a limited amount of resources, one of the main concerns in the effort to improve these applications is the study and implementation of efficient processing procedures. This thesis handles Bayer to RGB conversion schemes, by studying and comparing them.

Previous work on image processing has revealed that applying a transformation on the RGB image is more resource demanding than applying the same transformation on the Bayer image. Related work by Tommy Olsen and Jo Steinar Strand [1] has shown that the Bayer to RGB conversion scheme, where compression is applied on the Bayer image, can be implemented in two different ways; either by using Bayer as it appears from the image sensor as input to the compression scheme or by using Bayer where the colours red green and blue are distinguished into three layers. Both implementations reveal overall better results in terms of quality compared to classical JPEG, when lossless compression is applied. Other related work by Chin Chye Koh [2], [3] has revealed that the application of transformations, such as subbband decomposition and decimation, on Bayer image before compression allow increased compression ratios or improvement in image quality.

The goal of this thesis is the application of linear transformations on Bayer to RGB Conversion schemes, in their various implementations, in order to create efficient in time and quality processing procedures. The transformations are examined with and without the application of error. The outcome of the thesis is the algebraic modeling and the comparison, in terms of image quality, of two implementations of the Bayer to RGB Conversion schemes as well as two proposals for their improvement.

Table of contents:

Acknowledgements..........2

1 INTRODUCTION..........7

2 DIGITAL IMAGE ACQUISITION AND PROCESSING TOOLS..........11

2.1 Image sensors..........11

2.1.1 The CCD image sensor..........11

2.1.2 The CMOS image sensor..........13

2.2 Bayer Color filter array..........13

2.3 Demosaicing..........14

2.3.1 Interpolation algorithms..........15

2.4 Colour correction-White balancing..........16

2.5 Compression..........17

2.5.1 Discrete Cosine Transform..........17

2.5.2 Quantization..........19

2.5.3 Zig-zag ordering..........20

2.5.4 Runlength and Entropy encoding..........20

3 CONVERSION SCHEMES..........22

3.1 Introduction..........22

3.2 BAYER to RGB conversion..........24

3.3 RGB to BAYER conversion..........27

4 MODEL APPLICATIONS..........28

4.1 Introduction..........28

4.2 Linear Transformations on Bayer to RGB conversion..........28

4.3 Linear Transformations on RGB to Bayer conversion..........30

4.4 DCT transformation on Bayer to RGB conversion..........31

4.4.1 Model case 1..........33

4.4.2 Model case 2..........34

4.4.3 Model case 3..........36

5 ERROR ANALYSIS..37

5.1 Introduction..37

5.2 Bayer to RGB conversion with error on Bayer..38

Study: distortion per Bayer image operation...42

Study: distortion per odd and even pixels' layer..43

Study: distortion of weighted error...44

Study: distortion of weighted error on full layer...47

5.3 Bayer to RGB conversion with error on DCT...49

Study: distortion on full layer..52

5.3.1 Spectral analysis...52

5.3.2 Experimental measurements..58

5.3.3 Proposed improvements..67

5.4 Conversion schemes' comparison...70

6 CONCLUSION..79

7 APPENDIX A-Mathematical proof...83

8 References:..90

Index of figures:

Figure .1-Conversion Scheme 1, Transformation on RGB image...9

Figure .2-Conversion Scheme 2, Transformation on Bayer image..9

Figure .3-Implementation 1, with two green layers...9

Figure .4-Implementation 2, with one green layer...9

Figure .5-CCD sensor...12

Figure .6-Foveon sensor...12

Figure .7-Description of acquisition for Color film, Digital sensor with CFA and Foveon sensor...13

Figure .8-CMOS sensor...13

Figure .9-Bayer CFA...14

Figure .10-Demosaicing procedure...15

Figure .11-Bayer pattern..16

Figure .12-Spectrum of an 8x8 image...18

Figure .13-Quantization matrix..20

Figure .14-Zig-zag ordering...20

Figure .15-Conversion scheme 1...22

Figure .16-Conversion scheme 2...22

Figure .17-Conversion Scheme 2...28

Figure .18-Error matrix ε...33

Figure .19-Model case 1 block diagram...34

Figure .20-Model case 2 block diagram...35

Figure .21-Model case 3 block diagram...37

Figure .22-Model case 1 with error...39

Figure .23-Model case 3 with error...40

Figure .24-Error of green layer in model case 1...41

Figure .25-Error of green layer in model case 3...41

Figure .26-Error comparison of model case 1 and 3..48

Figure .27-Model case 1with error on DCT...50

Figure .28-Model case 3 with error on DCT..50

Figure .29-Error on DCT of green layer in model case 1...51

Figure .30-Error on DCT of green layer in model case 3...51

Figure .31- Spectrum of an 8x8 subimage in model case 1 (log scale).............................54

Figure .32-Spectrum of an 8x8 subimage in model case 3 (log scale)..............................57

Figure .33-Distribution of signal energy in both model cases...58

Figure .34-Comparison of the Spectrum of images without and with edges (log scale)......62

Figure .35-Output image 8 of model case 1...65

Figure .36-Output image 8 of model case 3...65

Figure .37- Output image 9 of model case 1..66

Figure .38- Output image 9 of model case 3..........66

Figure .39-Improved error matrix..........67

Figure .40-Improved implementation of model case 3..........69

Figure .41-Bayer to RGB Conversion scheme 1..........71

Figure .42-Bayer to RGB Conversion scheme 2..........71

Figure .43-Conversion scheme 1 (transformation on RGB)..........80

Figure .44-Conversion scheme 2 (transformation on Bayer)..........80

Index of tables:

Table .1-Spectrum of an 8x8 subimage in model case 1..........53

Table .2-Spectrum of an 8x8 subimage in model case 3..........55

Table .3-Distortion comparison with error on DCT..........61

Table .4-Distortion with improved error..........68

Table .5-Distortion with improved implementation..........70

Table .6-Distortion comparison of Conversion schemes 1 and 2..........76

Table .7-Computational time comparison of Conversion schemes 1 and 2..........78

Table .8-Storage capacity comparison of Conversion schemes 1 and 2..........78

Table .9-Transmission time comparison of Conversion schemes 1 and 2..........79

1 INTRODUCTION

Digital imaging is a field of science that grows rapidly the recent years. Mobile phones equipped with digital cameras, advanced digital cameras and a huge variation of applications based on digital image demand highly sophisticated hardware and software solutions. Given the fact that in most systems there is a limited amount of resources, such as CPU processing time, memory usage, transmission speed or network bandwidth and battery, one of the main concerns in the effort to improve these systems is the study and implementation of efficient processing procedures.

The most widely used camera sensors CCD and CMOS combined with the Bayer CFA, produce the digital image, called Bayer image. The sensor captures only one third of the image information and then interpolation is performed in order to acquire the full RGB image. In most applications, data compression before storage or transmission is required, usually using the most common compression standard which is JPEG. Another more efficient scheme is to apply JPEG on the image before demosaicing, which is performed after storage or transmission. In addition to compression these schemes are used for other linear transformations, too. The transformations studied are the basic linear transformations and Discrete Cosine Transform with and without the application of error.

Previous work on image processing in this field of application has revealed that applying a transformation on the RGB image requires more processing power compared to the same procedure on the Bayer image. Furthermore the RGB image requires more storage capacity and transmission time over a network than the Bayer. Related work by Tommy Olsen and Jo Steinar Strand [1] has shown that the Bayer to RGB conversion scheme, where compression is applied on the Bayer image, can be implemented in two different ways; either by using Bayer as it appears from the image sensor as input to the compression scheme or by using Bayer where the colours red green and blue are distinguished into three layers. Both implementations reveal overall better results in terms of quality compared to classical JPEG, when lossless compression is applied. Other related work by Chin Chye Koh [2], [3] has revealed that the application of transformations, such as subband decomposition and decimation, on Bayer image before compression allow increased compression ratios or improvement in image quality.

The goal of this thesis is the application of linear transformations on Bayer to RGB Conversion schemes and the mathematical modeling of the entire process, including errors and their effects on the final image quality. The analysis of the mathematical modeling is achieved through the development of the matrix framework. The models and their analysis aim in the creation of efficient in time and quality processing procedures. The block diagrams below depict these Conversion schemes, which were mentioned:

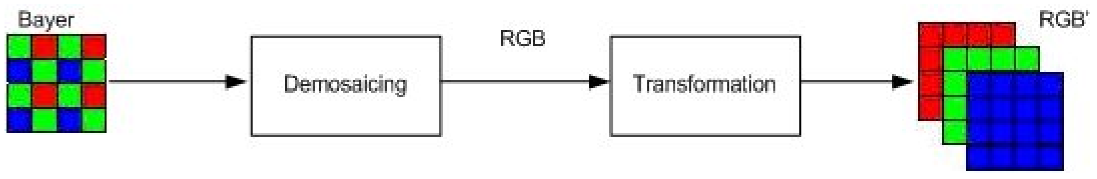


Figure .1-Conversion Scheme 1, Transformation on RGB image

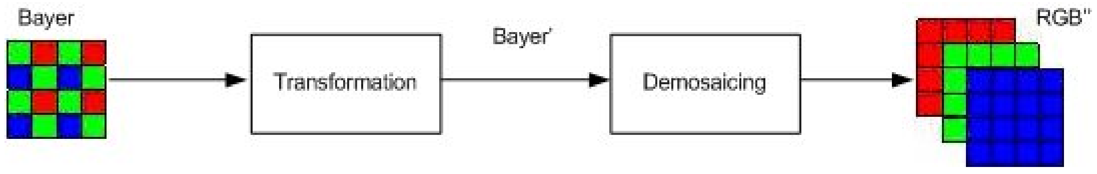


Figure .2-Conversion Scheme 2, Transformation on Bayer image

In the second scheme, where the transformation is applied directly on the Bayer image, two different implementations are examined. These implementations differ in the way the transformation is applied on the green pixels. The block diagrams below depict the different implementations:

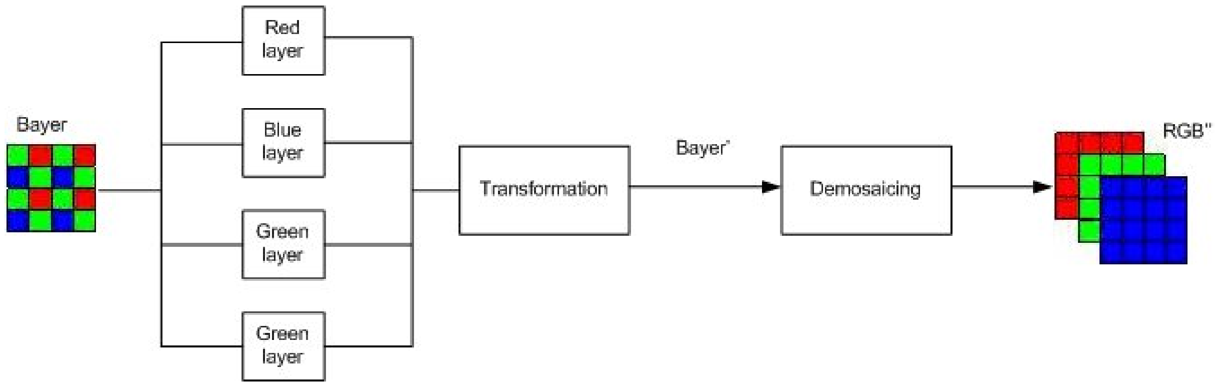


Figure .3-Implementation 1, with two green layers

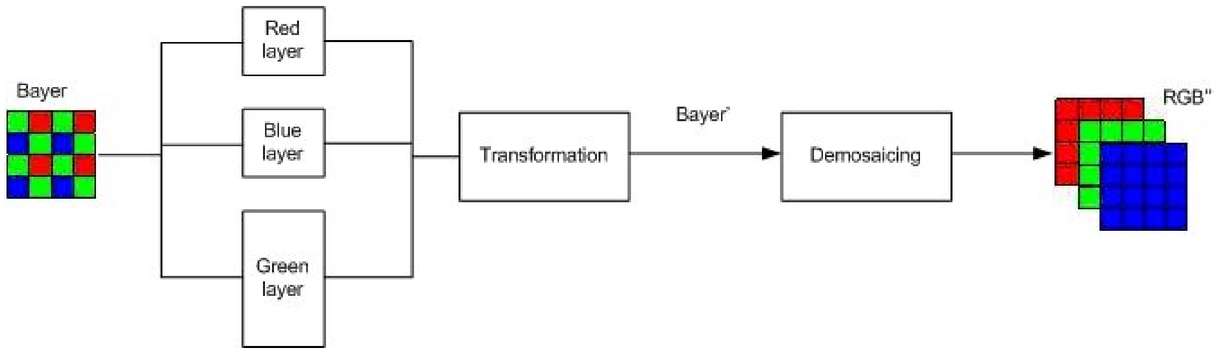


Figure .4-Implementation 2, with one green layer

In the implementation described by figure 1.3 the colour layers of the Bayer image are transformed separately, with the green layer being split to one (green$_1$) holding the pixels of the odd lines and another (green$_2$) holding the pixels of the even lines. The implementation described by figure 1.4 differs on the green layer handling. This time there is one layer which holds the whole information of the green colour. This thesis aims at the in depth studying of the above implementations, when a linear transformation [9], [13], which includes errors, is applied on them. The transformations studied are the linear ones, such as filtering and DCT, which perform an operation either on the spatial or on the frequency domain of the image. Moreover, linear transformations have been selected because they are appropriate for matrix algebraic modeling. Each transformation consists of two operations; one on the rows and one on the columns of the image. These operations are combined by the Kronecker product of the transformation matrices, which is applied on the column order vector form of the image.

The outcome of the thesis is the comparison, in terms of image quality, of the implementations depicted by figures 1.3 and 1.4, as well as two proposals for their improvement. In addition another achievement is the modeling of the Bayer to RGB Conversion schemes, when linear transformations are applied either on the Bayer or the RGB image.

In Section 2 general information about image acquisition, processing operations and compression is presented. In Section 3 the two Conversion schemes are analyzed as well as the mathematical model of Bayer to RGB conversion and the inverse. In Section 4 the applications of Conversion scheme 2 are presented. Firstly the Bayer and RGB formats are connected through the equation of the transformations applied on them. Then three different implementations of this scheme are presented. In Section 5 there is the error analysis. In this section the three implementations are analyzed when the error is applied directly on the image data and when it is applied on the DCT of the image. Apart from the theoretical analysis, experimental results are presented too. Additionally, the Conversion schemes of figures 1.1 and 1.2 are compared in terms of output image quality and experimental results are presented in this case too. Finally, there are two proposals for improvements with the presentation of their experimental results. In Section 6 the conclusions of this thesis are presented. In Section 7 the Appendix A gives the mathematical proofs. Finally in Section 8 the References are presented.

2 DIGITAL IMAGE ACQUISITION AND PROCESSING TOOLS

2.1 Image sensors

2.1.1 The CCD image sensor

CCD image sensor, which stands for Charge Coupled Device, is an electronic device capable of transforming a light pattern (image) into an electric charge pattern (an electronic image). The CCD consists of several individual elements that have the capability of collecting, storing and transporting electrical charge from one element to another. This along with the photosensitive properties of silicon is used to design the image sensor.

Figure .5-CCD sensor

A CCD image sensor can be a colour sensor or a monochrome one. In a colour image sensor there is only one layer of pixels so that a part of the colour can be captured. In order to separate each colour from the other an integral RGB colour filter array (CFA) is used which provides colour responsivity and separation. CCD sensors create high quality and low noise images although they are rather power consuming.

An attempt to improve even more the image quality of cameras using this sensor has led to the creation of the 3CCD sensor. 3CCD cameras are equipped with three separate CCDs, each one taking a separate measurement of red, green and blue light.

Finally, the most improved CCD sensor has been developed the recent years by Foveon [6], which combines the simplicity of one CCD sensor per camera with full colour capturing. This sensor has three layers of pixels embedded in silicon. The layers are positioned to take advantage of the fact that silicon absorbs different wavelengths of light to different depths. The bottom layer records red, the middle layer records green and the top layer records blue. The most important advantages of this sensor is quality (sharper pictures, truer colours and fewer artefacts) and less processing power.

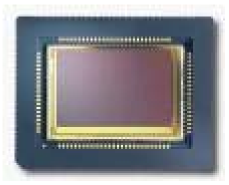


Figure .6-Foveon sensor

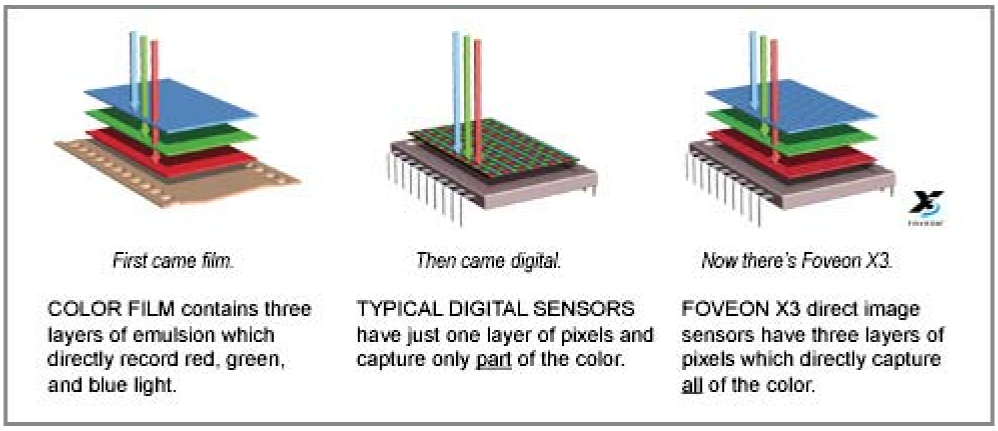

First came film.
COLOR FILM contains three layers of emulsion which directly record red, green, and blue light.
Then came digital.
TYPICAL DIGITAL SENSORS have just one layer of pixels and capture only part of the color.
Now there's Foveon X3.
FOVEON X3 direct image sensors have three layers of pixels which directly capture all of the color.

Figure .7-Description of acquisition for Color film, Digital sensor with CFA and Foveon sensor

2.1.2 The CMOS image sensor

CMOS image sensor, which stands for Complementary Metal Oxide Semiconductor, is a lower power consumption sensor than CCD with generally a much simpler design. There are two categories of CMOS sensors, analogue and digital, as defined by their manner of output. The main characteristics of this sensor is that it is quite susceptible to noise with low quality and resolution images; however it is still competitive in plenty applications because of the low cost.

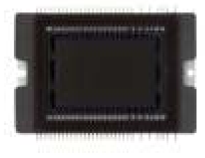


Figure .8-CMOS sensor

2.2 Bayer Color filter array

The most widespread method to give RGB colour sensitivity to image sensors is the application of a Colour Filter Array (CFA) on top of a black & white imager. In most cases a 3-color Red-Green-Blue (RGB) pattern is used, although others exist too. Bayer colour filter array [4] is the most popular format for digital acquisition of colour images. The pattern of the colour filters varies in the sequence of the colours that are filtered. The Bayer CFA that we use in this thesis is the one shown below:

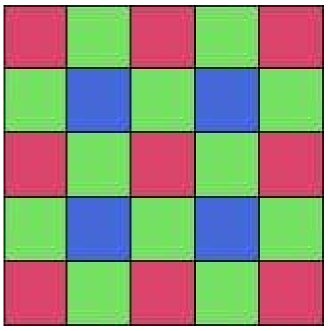


Figure .9-Bayer CFA

From the CFA it is observed that half of the total number of elements is green (G), while a quarter of the total number is assigned to both red (R) and blue (B). There are used twice as many green elements as red or blue to mimic the human eye's greater resolving power with green light. In order to construct an RGB picture, it is needed to calculate Green and Blue values for each Red pixel, Blue and Red values for each Green pixel and Red and Green values for each Blue pixel through an operation named interpolation. Many different interpolation algorithms exist and they will be discussed later.

2.3 Demosaicing

Capturing colour images with a single CCD or CMOS sensor requires the use of a CFA which covers the sensor array. The use of CFA leads to the acquisition of just one of the many colour channels for every position of the sensor array. A method of calculating values of the other colour channels at each pixel is required to recover the full-colour image. This method is commonly referred as colour interpolation or colour demosaicing algorithm and it depends on the CFA configuration. In this thesis it is not our goal to examine such algorithms, as a result we have selected a simple and computationally inexpensive demosaicing algorithm, the bilinear.
In more detail the demosaicing operation is completed in three steps. First the Bayer image is downsampled to obtain the red colour plane, the green one and the blue one. The next step is the interpolation of each one of them using an appropriate algorithm to create the red layer, the green one and the blue one. Finally, the layers are combined in a single file to form the RGB image. This procedure is shown in the following figure:

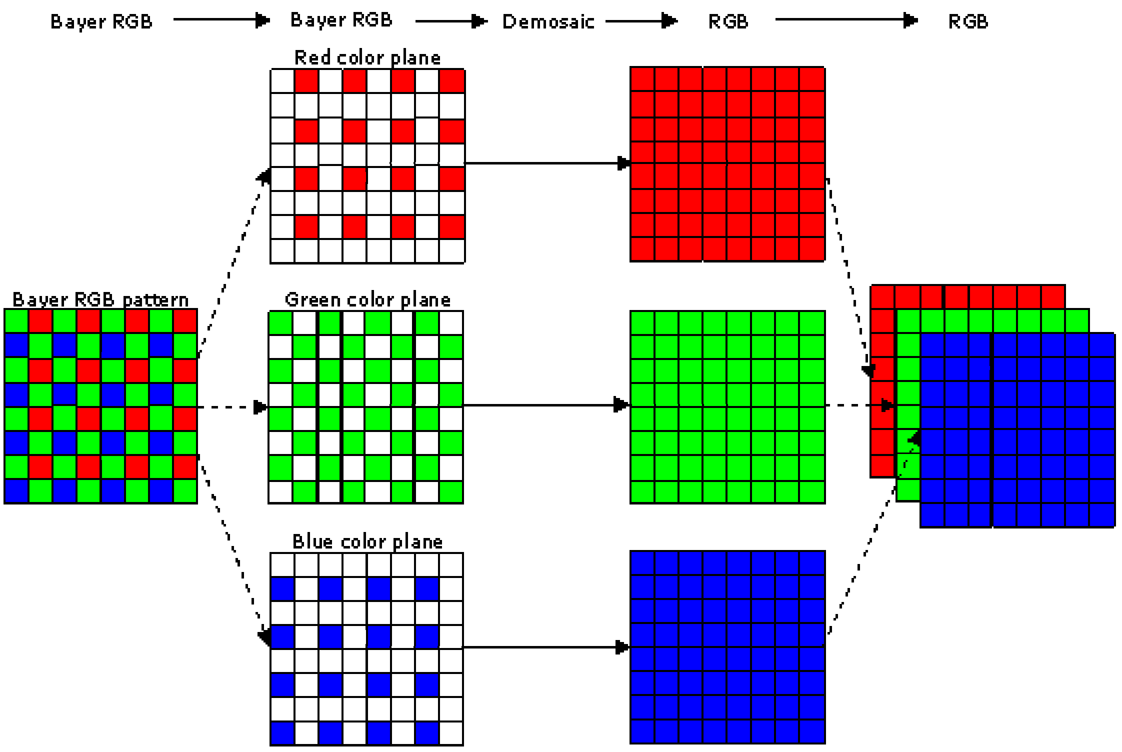


Figure .10-Demosaicing procedure

2.3.1 Interpolation algorithms

The goal of these algorithms, apart from the Bayer to RGB conversion, is also colour fidelity, no false colours, no jagged edges, and computational practicality [5]. There are plenty algorithms proposed ranging from simple linear to sophisticated adaptive ones. As expected, the advantages and disadvantages of these algorithms are weighting between image quality and computational cost. Interpolation algorithms are classified into two groups, the non-adaptive such as nearest neighbour replication, bilinear and cubic convolution and the adaptive ones such as edge sensing and interpolation with colour correction. Non-adaptive perform interpolation in a fixed pattern for every pixel, while adaptive algorithms detect local spatial features of the pixel neighbourhood and make effective choices depending on the algorithm. The one used in this thesis is the bilinear and it has been preferred because of its simplicity and computational cost effectiveness. The pattern that follows depicts a part of a typical Bayer image:

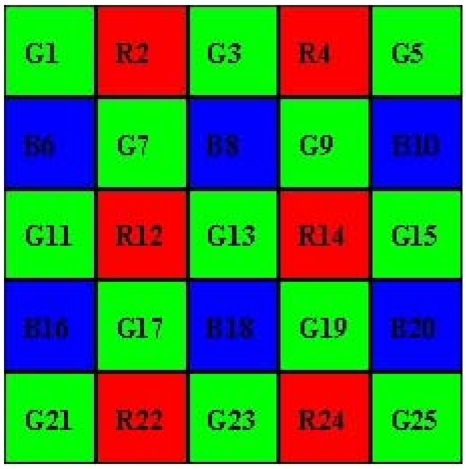


Figure .11-Bayer pattern

The algorithm of bilinear interpolation for:

- the green pixels at the red and blue positions is: the average of the upper, lower, left and right pixel values. For example: G8=(G3+G7+G9+G13)/4
- the red and blue pixels at the green positions is: the average of two adjacent pixel values in corresponding colour. For example: B7=(B6+B8)/2 and R7=(R2+R12)/2
- the red and blue pixels at the blue and red positions is: the average of four adjacent diagonal pixel values. For example: R8=(R2+R4+R12+R14)/4 and B12=(B6+B8+B16+B18)/4

2.4 Colour correction-White balancing

Colour correction and white balancing are processing operations performed to ensure proper colour fidelity in a captured digital camera image. In digital cameras an array of light detectors with colour filters over them is used to detect and capture the image. This sensor does not detect light exactly as the human eye does, and so some processing or correction of the detected image is necessary to ensure that the final image realistically represents the colours of the original scene.

Colour correction is an operation which compensates the differing colour temperatures in order to reconstruct correctly the desired colours or real life colours of an image.

White balancing is an operation which gives the camera a reference to white. The purpose of this technique is to aid in overcoming colour problems created by adverse lighting conditions.

2.5 Compression

The operations discussed in the previous sections describe the procedure from capturing an image to preparing the image data for compression. One of the most commonly used image compression algorithms is the JPEG [7].

JPEG is a compression standard for a wide range of applications and for images of varying resolution and size. JPEG is designed for compressing either full-colour or greyscale images, of natural and real-world scenes. JPEG can be lossy, meaning that the decompressed image appears some distortion in comparison to the original one or lossless where the produced image is exactly as the source one, which is used in specific applications. The compression rates that lossy JPEG can achieve are rather high, but they are limited by the demand of high image quality and low computational cost. The basic compression method that JPEG uses is the Discrete Cosine Transform (DCT), which is mainly used in lossy compression schemes. The processing steps of the DCT based compression scheme are the following: DCT, quantization, zigzag ordering, runlength encoding and entropy encoding.

2.5.1 Discrete Cosine Transform

The Discrete Cosine Transform (DCT) is a Fourier-related transform similar to the Discrete Fourier Transform. The DCT is a technique for converting a signal from the spatial domain to the frequency domain. It is often used in image and signal processing, especially for lossy data compression, because it has a strong energy compaction property which is to concentrate most of the signal information in a few low frequency components of the DCT. There are both one dimension and two dimensions algorithms for 1-d signals and 2-d signals respectively, such as images. The 2-d DCT is a 1-d DCT applied twice, once in the rows direction and once in the columns.

In order to apply the DCT the source image is separated to 8x8 blocks. The DCT is applied on each one of them and the transformation produced is an 8x8 block. The source image turns up by applying the inverse DCT to the transformed block. The energy compaction property results in gathering most of the signal energy at low frequencies. These frequencies appear in the upper left corner of the DCT and the high frequencies appear in the lower right values. The values of the elements with high horizontal and vertical index values tend to be small enough to be neglected with little visible distortion. Here lies the compaction property of the DCT weighting between image quality and compression rate. The figure below presents the energy compaction property of the DCT, showing that elements near (1, 1) hold most of the information of the signal and elements near (8, 8) hold the least:

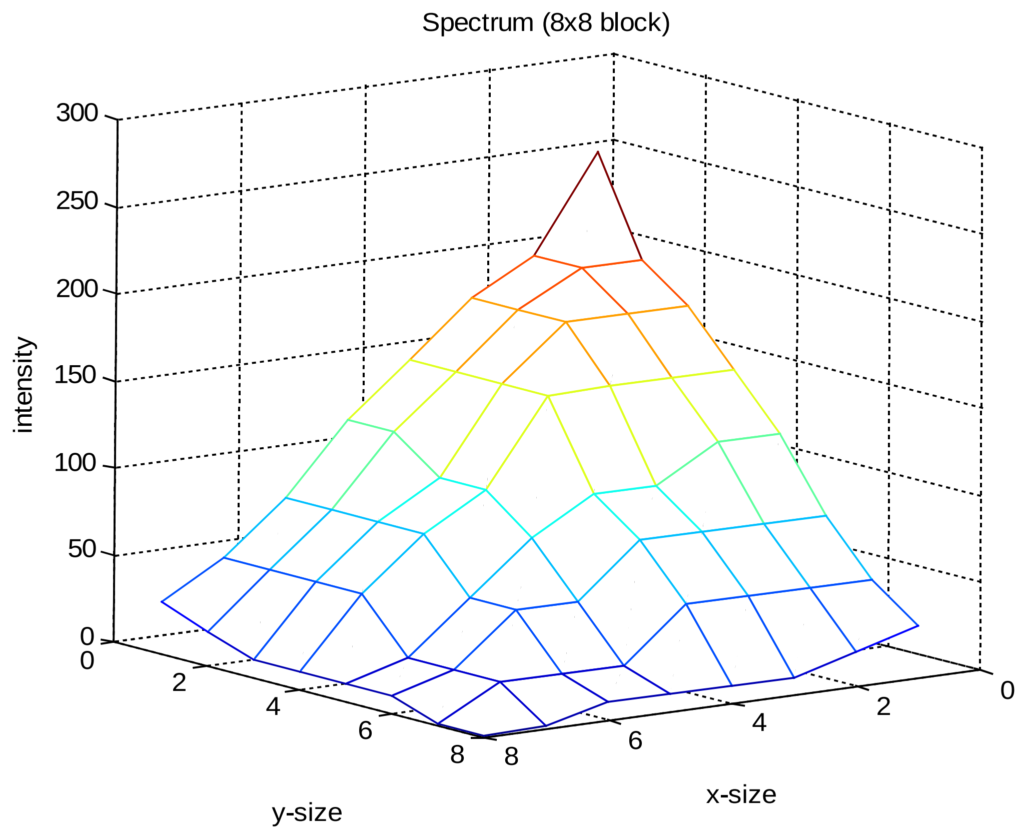


Figure .12-Spectrum of an 8x8 image

The 2-d DCT algorithm that is used in this thesis is the one shown below:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{array}{l} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{array}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

Equation .1-DCT algorithm

The 2-d inverse DCT algorithm that is used in this thesis is the one shown below:

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{array}{l} 0 \le m \le M-1 \\ 0 \le n \le N-1 \end{array}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

Equation .2-Inverse DCT algorithm

From the above it is concluded that the DCT is a linear transformation, which consists of the sum of cosines for every image value. It is represented by an 8x8 matrix applied on the rows and an 8x8 matrix applied on the columns of the image.

2.5.2 Quantization

The human eye is good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation. This fact allows one to reduce the amount of information in the high frequency components of the DCT of an 8x8 block. A quantization matrix of size 8x8 is defined by the application or the user and contains a constant for every DCT coefficient. Each coefficient is divided by the corresponding constant and the result is rounded to the nearest integer. As a result of this process, many of the higher frequency components of the DCT are rounded to zero. The goal of this process is to discard information which is not visually significant. A typical quantization matrix is the following:

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Figure .13-Quantization matrix

2.5.3 Zig-zag ordering

The next step after quantization of the DCT coefficients is zig-zag ordering, described by figure 2.10. In this step all coefficients are ordered in a zig-zag sequence by placing the low frequency before the high frequency ones. Usually, after quantization the majority of high frequency elements are equal to zero, so this procedure serves in the data preparation for the next step, runlength encoding.

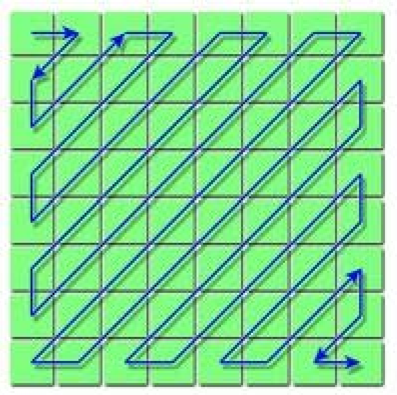


Figure .14-Zig-zag ordering

2.5.4 Runlength and Entropy encoding

Zig-zag ordering is expected to increase the run length of zeros at the end of the block. This property is used by runlength encoding, which maps the zig-zaged data into a sequence of data pairs, the first one indicating the length of sequential values and the second one their value itself. For example the following sequence:
0 0 0 0 0 is encoded as 5,0.

The final step of JPEG compression algorithm is entropy encoding. There are two different schemes used, Huffman and arithmetic encoding. Although arithmetic encoding often results in better compression ratio, it is not commonly used due to computational complexity. Huffman encoding is a lossless compression scheme based on the statistical characteristics of the runlength encoded DCT coefficients. It takes as an input the encoded coefficients, as described before, and the Huffman table which is specific to the application.

The analysis of the processing steps of the JPEG compression scheme reveals the importance of the DCT and its energy compaction property. As a linear transformation, DCT is used in this thesis to examine the result of the proposed conversion schemes in terms of quality. The application of error on the high frequency coefficients creates a minimum distortion on the output image, because of the energy compaction property of the DCT. The erroneous coefficients are then quantized, where depending on the application additional error is inserted. Then zig-zag transforms the 2-d image to 1-d vector where similar frequencies are grouped together, with many zeros being at the end. The zeros are eliminated and the vector is entropy encoded to finish the compression algorithm. In this thesis, though, only the DCT step is examined.

3 CONVERSION SCHEMES

3.1 Introduction

The sections above give a description of the most important parts that constitute the processing sequence of a digital image, from the acquisition to compression and display. The processing chain that is most commonly used is the one shown at the block diagram that follows:

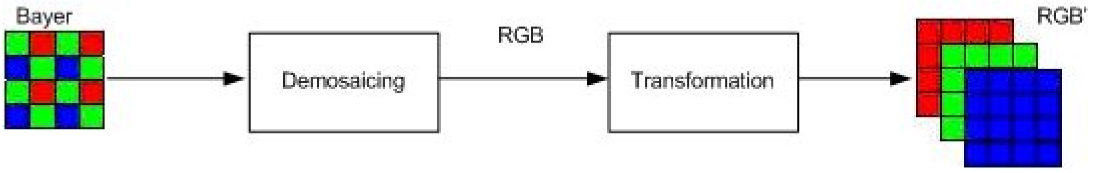


Figure .15-Conversion scheme 1

The sensor in combination with the CFA gives the Bayer image, which is demosaiced to produce the RGB image. This one is either compressed or a transformation is performed on it. After compression the data may be handled in many ways, such as being stored or transmitted via a network. Then the inverse procedure of the transformation follows in order to get the image in RGB format. This implementation has a drawback. There is a waste of computational resources as the compression algorithm is applied on an upsampled version of the RGB image, resulting in having more data to compress. Furthermore for the JPEG compression scheme there is even more waste of computational resources, as the image has to be converted to the Y'C_bC_r colour space in order to be compressed.

An improved processing chain, which is not so costly in computational resources, is to apply the transformation directly on the Bayer image, by omitting the demosaicing and the RGB to Y'C_bC_r conversion. The improved conversion scheme is presented below in the block diagram:

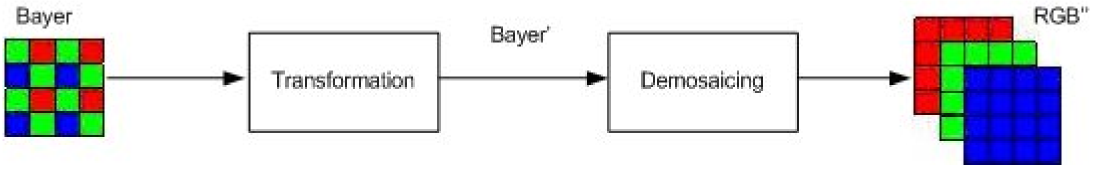


Figure .16-Conversion scheme 2

The conversion models analysed in this thesis are implemented according to the conversion scheme 2. These models are presented in the following sections. Before proceeding with the mathematical representation of the conversion schemes, some clarifications about the Bayer and RGB images have to be presented.

Bayer: The Bayer format varies from one camera sensor to another. It depends on the CFA filter that is applied. In this thesis we assume that the filter is of the following form: $\begin{bmatrix} R & G \\ G & B \end{bmatrix}$. From the Bayer quadruplet it is shown that the Bayer image is a composition of red, blue and green pixels. The green pixels are double in number in comparison to the red and blue ones. The transformations on the Bayer image that follow have better results if they are applied on each separate colour layer than on the whole image. As a result it is needed to split each colour layer from the others, which is achieved by downsampling the Bayer image. Furthermore for reasons of uniformity and ease of handling the Bayer image is separated to four distinct layers: the one with the blue pixels, the red, the green of the odd lines and the green of the even lines. All of them are a result of downsampling the Bayer image in both lines and columns and their size is m/2xm/2 (taken that the Bayer image is of size mxm). From now on the red layer will be symbolized as r, the blue as b, the green of the odd lines as g_1 and the green of the even lines as g_2.

RGB: The RGB image consists of three colour layers: the green, red and blue one. All of them have the same size mxm. From now on the red layer will be symbolized as R, the blue as B and the green as G.

Vector: Both Bayer and RGB images used in this thesis and their layers are converted to vector form [8], [9], [10]. The vector is formed by putting the columns one after the other, the inverse procedure gives back the original image. A linear transformation on the lines and columns of an image can be transferred to its vector form using the Kronecker product. Assuming an image f of size mxm and the transformations A and B applied on f as shown in the following equation: $F = A * f * B$. The same transformation applied on the vector form of this image is given by the equation: $F' = (B^T \Delta A) * f'$, where F′ is the vector form of the transformed image and f′ is the vector form of the source image. From the above equations it is clear that A matrix is nxm and B matrix is mxn, f′ is m^2x1 and the size

of $(B \Delta A)$ is computed by the multiplication of the lines sizes and the columns sizes of A and B matrices: (n*n)x(m*m).
After the above clarifications on the Bayer and RGB image and their vector form we can move on with the conversion models.

3.2 BAYER to RGB conversion

In this section the modeling of the Bayer to RGB image conversion is presented. We assume an image "Bayer" in Bayer format. We want to convert it to the RGB format. The equation that represents Bayer, in correspondence to its red, green and blue layers is the following:

$$Bayer = [(U_1 \Delta U_1)(D_1 \Delta D_1) + (U_2 \Delta U_2)(D_2 \Delta D_2) + (U_2 \Delta U_1)(D_2 \Delta D_1) + (U_1 \Delta U_2)(D_1 \Delta D_2)]Bayer =$$
$$= (U_1 \Delta U_1)r + (U_2 \Delta U_2)b + (U_2 \Delta U_1)g_1 + (U_1 \Delta U_2)g_2$$

where r, b, g_1 and g_2 are the four layers of the Bayer image.
Our aim is to create the R, B and G layers. In order to achieve this we have to upsample each of the r, b, g_1 and g_2 and then perform interpolation on each one of them. This is the case for red and blue layers but green is slightly different. After upsampling g_1 and g_2 there are two images of the size of Bayer image, both having information about green color. This information can be combined by addition to form the green layer of RGB image. This one will be interpolated to create the final green layer.
The final equation of the RGB image f is:

$$f = \begin{pmatrix}1\\0\\0\end{pmatrix} \Delta R + \begin{pmatrix}0\\1\\0\end{pmatrix} \Delta B + \begin{pmatrix}0\\0\\1\end{pmatrix} \Delta G = e_1 \Delta R + e_2 \Delta B + e_3 \Delta G =$$
$$= e_1 \Delta (IN \Delta IN)(U_1 \Delta U_1)r + e_2 \Delta (IN \Delta IN)(U_2 \Delta U_2)b + e_3 \Delta (IN \Delta IN)[(U_2 \Delta U_1)g_1 + (U_1 \Delta U_2)g_2]$$

In equations 3.1 and 3.2 there are some matrices used for different operations, such as downsampling, upsampling and interpolation. These are the main procedures that have to be done in order to convert an image from one format to the other. The matrices used are described in more details below:

Downsampling: it is an irreversible linear operation that selects some elements from a source matrix and discards the rest. There are two matrices for downsampling. Both of them reduce the size of the image they are applied to by half, though they differ on the pixels they downsample. D_1 matrix keeps the odd lines or columns and discards the even ones. On the contrary the second matrix, D_2, keeps the even lines or columns and discards the odd ones. The combination of D_1 and D_2 enables to downsample every image both in lines and columns and keep only either red pixels or green or blue. Their size depends on the image they are applied to. For a Bayer image of size mxn and for the red pixels, D_1 of size m/2xm would be used to downsample the lines and another D_1 of size nxn/2 would be used to downsample the columns. In the same way, the two downsample matrices can be combined in order to get the blue or green layers. As mentioned before in this model all images and layers are used in their vector forms. The downsampling matrices can be combined by Kronecker product, as shown earlier in the previous section, so as to perform the same transformation on the vector form of the source image. For example, assuming that we want to downsample the Bayer image to get the red layer, then this operation is shown in the following equation: $r = D_1 * Bayer * D_1^T$. The same operation on the vector form of Bayer is given by: $r' = (D_1 \Delta D_1) * Bayer'$, where r′ and Bayer′ are the vector forms of r and Bayer respectively. Throughout this thesis both D_1 and D_2 will be expressed in terms of the identity matrix I, whose size depends on the source image, and the unity vector ε_κ, whose size is 1x2 with a non zero element in the k-th position. Additionally, matrices I_0 and I_1 will replace D_1 and D_2 respectively, for computational and modeling convenience as shown below. Assuming that the image to be downsampled is mxm then I will be m/2xm/2. The following equation shows the expression:

$$D_1 = I \Delta \begin{bmatrix} 1 & 0 \end{bmatrix} = I \Delta \varepsilon_0 = I_0$$
$$D_2 = I \Delta \begin{bmatrix} 0 & 1 \end{bmatrix} = I \Delta \varepsilon_1 = I_1$$

Upsampling: it is a linear operation that increases the size of the source matrix by inserting zeros in the new positions. In the conversion model it is used to upsample the layers of the Bayer image by putting zeros in the new pixels, which are going to be replaced by the process of interpolation. Upsampling is handled in the same way described for downsampling. There are two upsampling matrices which double the size of the image they are applied to, though they differ on the pixels they upsample. U_1 keeps the pixels of the source matrix in the odd lines or columns and inserts zeros in the even lines or columns by expanding the matrix. On the contrary the second matrix, U_2 keeps the pixels of the source matrix in the even lines or columns and inserts zeros in the odd lines or columns by expanding the matrix. The combination of U_1 and U_2 enables us to upsample every image both in lines and columns for the red layer, green or blue one. Their size depends on the image they are applied to. For the red layer of size m/2xn/2, U_1 of size mxm/2 would be used to upsample the lines and another U_1 of size n/2xn would be used to upsample the columns. In the same way, the two upsample matrices can be combined in order to transform the blue or green layers. As mentioned before in this model all images and layers are used in their vector forms. The upsampling matrices can be combined by Kronecker product, as shown earlier in the previous section, so as to perform the same transformation on the vector form of the source image. For example, assuming that we want to upsample the red layer of the Bayer image to get the red layer in full size, then this operation is shown in the following equation: $R = U_1 * r * U_1^T$. The same operation on the vector form of the red layer is given by: $R' = (U_1 \Delta U_1) * r'$, where R′ and r′ are the vector forms of red layer in full size and red layer respectively. Throughout this thesis both U_1 and U_2 will be expressed in terms of the identity matrix I, whose size depends on the source image, and the unity vector ε_k, whose size is 1x2 with a non-zero element in the k-th position. Additionally, matrices I_0 and I_1 will replace U_1 and U_2 respectively, for computational and modeling convenience as shown below. Assuming that the image to be upsampled is m/2xm/2 then I will be m/2xm/2. The following equation shows the expression:

$$U_1 = I \,\Delta \begin{bmatrix} 1 \\ 0 \end{bmatrix} = I \,\Delta \varepsilon_0^T = I_0^T$$

$$U_2 = I \,\Delta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = I \,\Delta \varepsilon_1^T = I_1^T$$

Interpolation: The last matrix used in the model is IN, the one that performs the interpolation. This matrix can take several forms depending on the type of interpolation performed. The process of interpolation is a linear process performed in lines and columns and its complication depends on the interpolation algorithm. If the image is of size mxn then IN has size mxm for line interpolation and nxn for column interpolation.

3.3 RGB to BAYER conversion

In this section the modeling of the RGB image to Bayer conversion is presented. In this case the RGB image is given which means that Red, Green and Blue layers are known. The Bayer image is expressed using them as input. RGB image can be expressed by the following equation:

$$f = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \Delta R + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \Delta B + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \Delta G = e_1 \Delta R + e_2 \Delta B + e_3 \Delta G$$

Where: R, B and G correspond to the Red, Blue and Green layer respectively. So the image can be represented as an addition of the Kronecker product of the three layers (R, B and G) with the unity vectors e_1, e_2 and e_3.

Our effort here is to use the only known data, which is f, in order to construct the Bayer image that corresponds to f. The Bayer image is represented as an addition of four factors, which are the red layer, the blue one and two layers to represent the green one (green$_1$, green$_2$). The conversion procedure is done by downsampling R in both lines and columns to give r, which holds all the red pixels of the Bayer image. B is also downsampled to create b which holds the blue pixels. G is downsampled twice, the first time to give g_1, which holds the green pixels in the odd lines and the second time to give g_2, which holds the green pixels in the even lines.

The equation of the Bayer image is:

$$Bayer = (U_1 \Delta U_1)(D_1 \Delta D_1)R + (U_2 \Delta U_2)(D_2 \Delta D_2)B + (U_2 \Delta U_1)(D_2 \Delta D_1)G + (U_1 \Delta U_2)(D_1 \Delta D_2)G$$

Where R, B and G are the Red, Blue and Green layers of the RGB image respectively and D and U are the downsample and upsample matrices which were analyzed in section 3.1.

4 MODEL APPLICATIONS

4.1 Introduction

In this section the second Conversion scheme is studied in order to get useful conclusions about the relation of an RGB and a Bayer image. The conversion scheme is shown in the following figure:

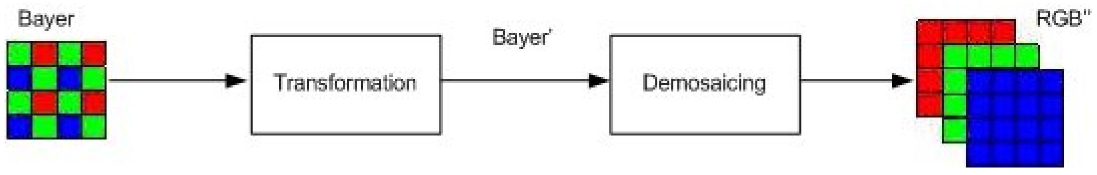


Figure .17-Conversion Scheme 2

This study is being held by applying basic linear transformations on the Bayer image, such as downsampling, filtering and the first stage of JPEG transformation (DCT). The transformations are applied in both ways of the conversion schemes (from BAYER to RGB and the reverse). The goal of this section is to find an algebraic expression that connects a transformation on the Bayer image with the transformation on the corresponding RGB image and the inverse. After that the study is focused on the application of the DCT directly on the Bayer image and three implementations are proposed and analyzed.

4.2 Linear Transformations on Bayer to RGB conversion

In this section we want to apply a matrix B, which performs a linear transformation, such as downsampling or filtering, on the Bayer image. The corresponding RGB image can be found by the conversion equation 3.2 and it is symbolized as f′. If a new RGB image is computed by the Bayer, then there is a matrix A, which performs a transformation on the new RGB, so as to result in the same f′. The transformed

Bayer image is symbolized as Bayer′ and the same stands for f′. From equation 3.1 Bayer′ can be expressed as follows:

$$Bayer' = (B \Delta B)Bayer =$$
$$= (B \Delta B)(U_1 \Delta U_1)r + (B \Delta B)(U_2 \Delta U_2)b + (B \Delta B)(U_2 \Delta U_1)g_1 + (B \Delta B)(U_1 \Delta U_2)g_2$$

The transformation matrix B is applied on the upsampled layers of Bayer, which is not desired. From matrix B and the upsampling matrices, a matrix B′ can be found which performs the same operation on the Bayer layers. This matrix is given by the equation: $B' = U^T * B * U$, where the operation performed on B is downsampling on both lines and columns.

Appendix A.1 presents the proof of this equation.

So, the previous equation becomes:

$$Bayer' = (B \Delta B)Bayer =$$
$$= (U_1 \Delta U_1)(B' \Delta B')r + (U_2 \Delta U_2)(B' \Delta B')b + (U_2 \Delta U_1)(B' \Delta B')g_1 + (U_1 \Delta U_2)(B' \Delta B')g_2$$

According to the conversion equation 3.2, f′ can be expressed as follows:

$$f = e_1 \Delta (IN \Delta IN)(U_1 \Delta U_1)(B' \Delta B')r + e_2 \Delta (IN \Delta IN)(U_2 \Delta U_2)(B' \Delta B')b +$$
$$+ e_3 \Delta (IN \Delta IN)[(U_2 \Delta U_1)(B' \Delta B')g_1 + (U_1 \Delta U_2)(B' \Delta B')g_2]$$

In this part it is assumed that there is an equivalent transform on RGB image. This transform is matrix A which results in the same f′, when applied on the RGB image. According to the conversion equation 3.2 the expression of f′ using the transformation A is:

$$f' = (A \Delta A) f =$$
$$= e_1 \Delta (A \Delta A)(IN \Delta IN)(U_1 \Delta U_1)r + e_2 \Delta (A \Delta A)(IN \Delta IN)(U_2 \Delta U_2)b +$$
$$+ e_3 \Delta (A \Delta A)(IN \Delta IN)[(U_2 \Delta U_1)g_1 + (U_1 \Delta U_2)g_2]$$

Equations 4.2 and 4.3 are two different expressions of the RGB images, which we want to be the same. If B performs a simple linear operation, such as filtering, in the first expression of f′, then the upsampled layers of Bayer are filtered and interpolation is performed. However in the second expression of f′, interpolation precedes the application of A. We have already defined B to perform a linear transformation, so the two expressions of f′ could be equal only if A performs a similar linear operation. In fact the relation of the two matrices is given by the following equation:

$$IN * U * B' = A * IN * U$$

Matrices A and B′ have such dimensions that all multiplications hold.

Appendix A.1 presents the proof of equation 4.4.

4.3 Linear Transformations on RGB to Bayer conversion

In this section we want to apply a matrix A, which performs a linear transformation, such as downsampling or filtering, on the RGB image. The corresponding Bayer image can be found by the conversion equation 3.4 and it is symbolized as Bayer′. If a new Bayer image is computed by the RGB, then there is a matrix B, which performs a transformation on the new Bayer, so as to result in the same Bayer′. The transformed RGB image is symbolized as f′ and the same stands for Bayer′. From equation 3.3 f′ can be expressed as follows:

$$f' = (A \Delta A) f = e_1 \Delta (A \Delta A) R + e_2 \Delta (A \Delta A) B + e_3 \Delta (A \Delta A) G$$

According to conversion equation 3.4 Bayer′ can be expressed as follows:

$$\begin{aligned} Bayer' &= (U_1 \Delta U_1)(D_1 \Delta D_1)(A \Delta A) R + (U_2 \Delta U_2)(D_2 \Delta D_2)(A \Delta A) B + \\ &+ [(U_2 \Delta U_1)(D_2 \Delta D_1) + (U_1 \Delta U_2)(D_1 \Delta D_2)](A \Delta A) G \end{aligned}$$

In this part it is assumed that there is an equivalent transform on Bayer image. This transform is matrix B which results in the same Bayer′, when applied on the Bayer image. According to the conversion equation 3.4 the expression of Bayer′ using the transformation B is:

$$\begin{aligned} Bayer' &= (B \Delta B) Bayer = \\ &= (B \Delta B)(U_1 \Delta U_1)(D_1 \Delta D_1) R + (B \Delta B)(U_2 \Delta U_2)(D_2 \Delta D_2) B + \\ &+ (B \Delta B)(U_2 \Delta U_1)(D_2 \Delta D_1) G + (B \Delta B)(U_1 \Delta U_2)(D_1 \Delta D_2) G \end{aligned}$$

Equations 4.4 and 4.5 are two different expressions of Bayer′. If A performs a simple linear operation, such as filtering, in the first expression of Bayer′, then the layers of the RGB image are filtered, the result is downsampled and then upsampled to obtain the size of the Bayer image. However in the second expression of Bayer′ downsampling and upsampling precede the application of B. Before we go on with conclusions we have to notice that the downsampling matrix D might not be the same in both expressions. This depends on matrix A. Though even if D differs, it is only its size but not its format. From now on it is considered that D is in fact different, in order to get more general conclusions, so D refers to the first case and D′ to the second one. A is already defined to perform a linear transformation, so the two expressions of Bayer′ could be equal only if B performs a similar linear operation. In fact the relation of the two matrices is given by the following equation:

$$D * A * D^T = U^T * B * U$$

Matrices A and B have such dimensions that all multiplications hold. Equation 4.7 reveals that matrices A and B are equal when the operation of downsampling on both lines and columns is applied on them. In the second part of the equation it is not clear that the operation performed is downsampling. So it has to be noticed that the relation between the downsampling and upsampling matrices is the following: $D=U^T$, so equation 4.7 can be written as:

$$D * A * D^T = D * B * D^T$$

Appendix A.2 presents the proof of equation 4.7.

4.4 DCT transformation on Bayer to RGB conversion

The DCT itself is a linear transformation that can be applied on the vector form of either Bayer or RGB image [8], [9] and [13]. In this section the DCT transformation is applied on the Bayer image before demosaicing, as shown in conversion scheme of figure 4.1. The transformation can be applied using three different implementations, which are analyzed in detail below. The question that this section is trying to answer is when there is an error at the DCT level of Bayer, what is the effect on RGB? This question refers to all implementations, which as we will see generate different distortion for the same error.

Before proceeding with the analysis of the implementations some comments have to be made on the DCT and the error matrix used in this thesis. When applying a DCT transformation on an image, the image is split to 8x8 subimages and the transformation is applied on each one of them. The DCT operation on each subimage consists of an 8x8 matrix application on the lines (DCT_{lines}) and another 8x8 matrix application on the columns ($DCT_{columns}$). If the image is in vector form then every subimage is a 64x1 vector. The DCT matrix which is applied on the vector form of the subimage is 64x64 and it is a result of the Kronecker product of DCT_{lines} and $DCT_{columns}$, as shown in the following equation:

$$DCT = DCT_{columns} \Delta DCT_{lines}$$

This equation holds for every subimage. So the DCT matrix for the whole image can be represented as a repetition of the 64x64 DCT as: $I \Delta DCT$, where DCT is the 64x64 matrix and I is the identity matrix whose size depends on the image it is applied on. For a m^2x1 image I is m^2/64x m^2/64. The expression of the DCT combined with the identity matrix represents the application of a transformation on the whole image. Though, this expression hides two distinct operations. The first one is the operation of the DCT matrix on the blocks of the column ordering of the image, which is denoted by the Kronecker product of the identity and DCT matrices. The second operation is the DCT application on the rows and columns of each block, as shown in the equation above. The same happens with the inverse DCT.

It is assumed that the error is applied on the DCT transformation of an image. Because of the fact that images are spilt to 8x8 subimages, which are converted to their vector form, the error matrix has to be applied on each one of them separately. So the error is a matrix of size 64x64 which is applied on each subvector and is repeated as many times as the number of the subvectors. Two error matrices are used at the error analysis: one for the general behavior of the model, where the error matrix ε has k errors in even lines and l errors in odd lines, and another one where the error matrix ε has one error in even lines and one error in odd lines, both of which are in the last even and odd line. This error is represented by the following expression:

$I_{e1} \Delta (I_e + \varepsilon)$

$$with: \varepsilon = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Figure .18-Error matrix ε

Supposing that the image it is applied on has size m^2x1 then I_{e1} is a m^2/64xm^2/64 identity matrix, I_e is a 64x64 identity matrix and ε is 64x64. If there is no error applied then ε=0 and no distortion is generated.

4.4.1 Model case 1

This model is described by the following procedure: Bayer image is separated in red, blue, $green_1$ and $green_2$ layers by downsampling and then DCT transformation is applied on each one of them. It is assumed that this transformation is described by the "DCT" matrix. The next step is to apply the Inverse DCT transformation which results back in the red, blue, $green_1$ and $green_2$ layers. It is assumed that this transformation is described by the "IDCT" matrix. Afterwards, demosaicing follows with upsampling and interpolation on each layer in order to get the Red, Blue and Green layers of the RGB image. If there is an error on the DCT level of the Bayer image then the inverse transformation will result in the color layers with the addition of distortion.

The procedure described can be expressed in mathematical terms:

$$r_{idct} = (I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_1 \Delta D_1) Bayer$$
$$b_{idct} = (I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_2 \Delta D_2) Bayer$$
$$g_{1idct} = (I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_2 \Delta D_1) Bayer$$
$$g_{2idct} = (I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_1 \Delta D_2) Bayer$$

The equation giving the RGB image is:

$$f = e_1 \Delta (IN \Delta IN)(U_1 \Delta U_1) r_{idct} + e_2 \Delta (IN \Delta IN)(U_2 \Delta U_2) b_{idct} +$$
$$+ e_3 \Delta (IN \Delta IN)[(U_2 \Delta U_1) g_{1idct} + (U_1 \Delta U_2) g_{2idct}]$$

Where, supposing that Bayer is m^2x1, I_{e1} is the identity matrix of size $m^2/(4*64)$x$m^2/(4*64)$, I_e is the identity matrix of size 64x64 and ε is 64x64 too.
The main characteristic of this implementation is the fact that the green pixels of odd lines are separated from the ones of the even lines. In this way the data are more compact, there are no zeros and the edges are avoided. All this is analyzed and commented in detail in the following sections. Red and blue layers are the same for all model cases, so they will not be examined. The block diagram that follows is the one of model case 1 and it refers to the green layer:

Case1

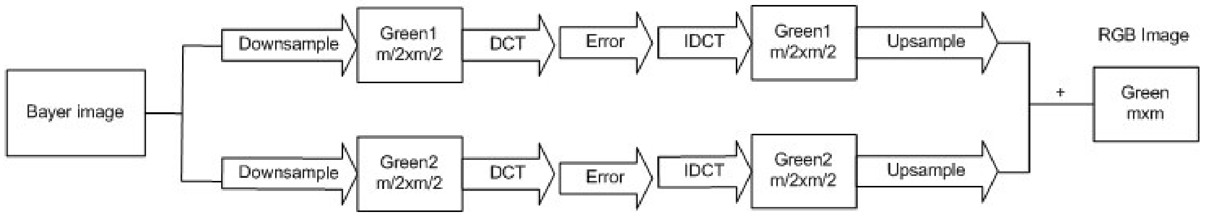


Figure .19-Model case 1 block diagram

4.4.2 Model case 2

The second model is differentiated in the green layer, in the way that the DCT transformation is applied. In this case the green pixels of the odd lines are not separated from the ones of the even lines but finally there is only one green layer on which the DCT is applied. The green layer is derived after downsampling the Bayer image into $green_1$ and $green_2$ and upsampling them in both directions by adding zeros in the empty positions. After that, the green layers are added. This procedure described can be expressed in mathematical terms:

$$G_1 = (U_2 \Delta U_1)(D_2 \Delta D_1) Bayer$$
$$G_2 = (U_1 \Delta U_2)(D_1 \Delta D_2) Bayer$$
$$G = G_1 + G_2$$

Moreover, a view of the green layer G is presented below in order to distinguish this model from the previous one:

As described before, the transformations are applied on the layers of the Bayer image as follows:

$$r_{idct} = (I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_1 \Delta D_1)Bayer$$
$$b_{idct} = (I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_2 \Delta D_2)Bayer$$
$$G_{idct} = (I_{e2} \Delta IDCT)[I_{e2} \Delta (I_e + \varepsilon)](I_{e2} \Delta DCT)G$$

So the equation giving the RGB image is:

$$f = e_1 \Delta (IN \Delta IN)(U_1 \Delta U_1)r_{idct} + e_2 \Delta (IN \Delta IN)(U_2 \Delta U_2)b_{idct} + e_3 \Delta (IN \Delta IN)G_{idct}$$

Where, supposing that Bayer is m^2x1, I_{e1} is the identity matrix of size $m^2/(4*64)$x$m^2/(4*64)$, I_{e2} is the identity matrix of size $m^2/64$x$m^2/64$, I_e is the identity matrix of size 64x64 and ε is 64x64 too.

The size of the green layer remains the same with the Bayer image, though the green pixels are actually occupying half of the space. The rest of it is filled with zeros. This is the fact that makes this image unsuitable for compression. The useless information that has been added by the zeros inserts high pass information, which is gathered in the high frequency elements of the DCT matrix. Because of the fact that error ε affects the high frequencies of the DCT the distortion generated will be maximum. Furthermore applying quantization will result in an additional loss of the energy held in the high frequencies of the signal. The block diagram that follows is the one of model case 2:

Case2

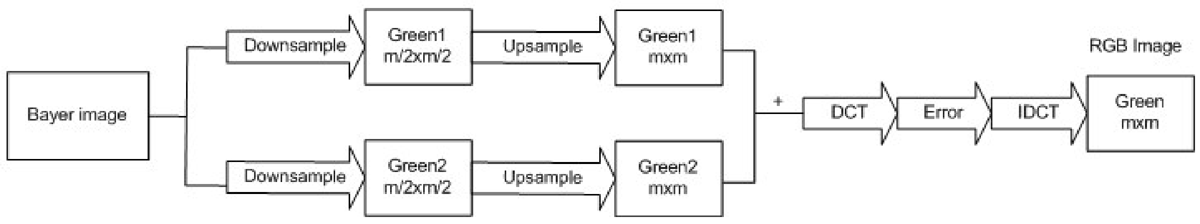


Figure .20-Model case 2 block diagram

This case does not get satisfactory results, so we will not work on it any more.

4.4.3 Model case 3

The model we presented in the previous section is an alternative implementation of our conversion scheme which however has important disadvantages. In this section an approach to the previous case is described which is more effective when compressing an image. In this approach the green layers are combined but without generating zero pixels, which means that we try to avoid inserting high pass information. This is done by creating a mxm/2 matrix which carries all green pixels. In more detail, for a Bayer image with dimensions mxm green_1 and green_2 layers with dimensions m/2xm/2 come up by downsampling in both dimensions. Instead of upsampling in both lines and columns, only the lines are upsampled by adding zeros in the even lines for green_1 and by adding zeros in the odd lines for green_2. The results are added to get the final green layer. The matrices created are as follows:

The procedure described can be expressed in mathematical terms:

The green layer can be expressed as:

$$G = [(I \Delta U_1)(D_2 \Delta D_1) + (I \Delta U_2)(D_1 \Delta D_2)]Bayer$$

The DCT transformations are applied in the same way:

$$r_{idct} = (I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_1 \Delta D_1)Bayer$$
$$b_{idct} = (I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_2 \Delta D_2)Bayer$$
$$G_{idct} = (I_{e3} \Delta IDCT)[I_{e3} \Delta (I_e + \varepsilon)](I_{e3} \Delta DCT)G$$

So the equation giving the RGB image is:

$$f = e_1 \Delta (IN \Delta IN)(U_1 \Delta U_1) r_{idct} + e_2 \Delta (IN \Delta IN)(U_2 \Delta U_2) b_{idct} +$$
$$+ e_3 \Delta (IN \Delta IN)[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)] G_{idct}$$

Where I_{e1}, I_{e3} and I are identity matrices of size $m^2/(4*64)$x$m^2/(4*64)$, $m^2/(2*64)$x$m^2/(2*64)$ and m/2xm/2 respectively. I_e and ε remain the same as before, 64x64.

The main characteristic of this implementation is the fact that there is one green layer on which the DCT is applied and there is no highpass information inserted because of zeros. Although at first sight this scheme seems to be compact with little possibilities of error, its study reveals that the shifting of the green pixels in the even columns to the left inserts some highpass information which generates a distortion to the output image. All this is analyzed and commented in detail in the following sections. The block diagram that follows is the one of case 3:

Case 3

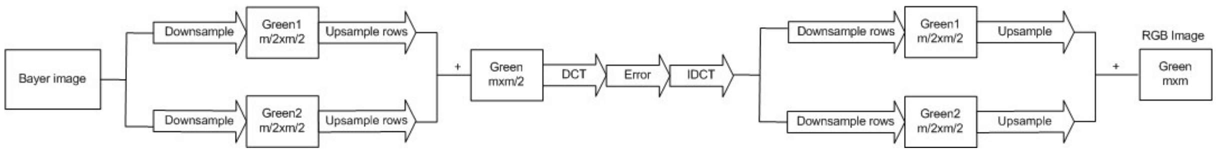


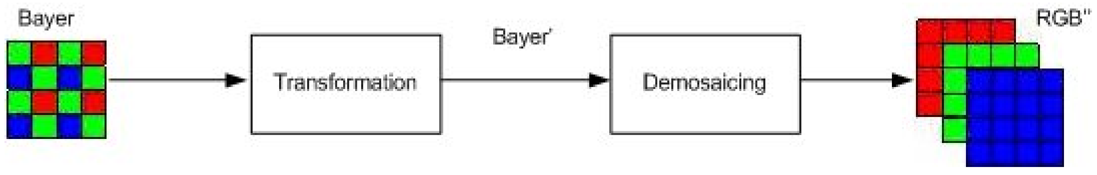
Figure .21-Model case 3 block diagram

5 ERROR ANALYSIS

5.1 Introduction

In this section the mathematical analysis of the conversion models in their different implementations is presented. The conversion scheme used is again the one of figure

4.1, where the transformation is applied on Bayer image before demosaicing. For convenience it is presented below:



The goal of this analysis is to examine thoroughly each implementation, when the erroneous DCT is applied on the input image. By this analysis we aim to reach conclusions about the different compression schemes in combination with loss of data. The further goal is the application of this knowledge on the JPEG compression scheme and the optimization of its implementations.

The situations of interest to us, in this section, include firstly the Bayer to RGB conversion with the application of the error ε directly on the Bayer image. In this situation the implementations of model case 1 and 3, presented in the previous section, are going to be compared in terms of quality performance. Model case 2 will not be considered, because as already mentioned it performs poorly on JPEG compression schemes. In the next situation we examine the application of the error ε on the DCT conversion of the Bayer layers. The spectral analysis will facilitate the comparison of the implementations of model case 1 and 3 with detailed explanation. Finally there is a comparison between the Conversion schemes described by figures 3.1 and 3.2, in which the transformation is considered to be the DCT with error ε. From this comparison it is concluded that the distortion of the RGB image is higher either when the transformation is applied on the Bayer or on the RGB image.

5.2 Bayer to RGB conversion with error on Bayer

In this case we examine the conversion from Bayer to RGB supposing that it involves an error. Our final goal is to study the model when the error is applied on DCT matrix, so the error will be imposed as if DCT existed. Our goal in this section is to examine the difference of the performance in quality between the RGB format images that arise from model case 1 and case 3.

In more detail, the procedure that we are going to follow, for model case 1, is first to downsample every layer of Bayer, which gives us red, green$_1$, green$_2$ and blue layers and before demosaicing we apply the error. The image f$_1$ that we get is affected by the error. If we apply the same error on model case 3, there will be an f$_2$ image with major differences compared to f$_1$. These differences result from the non-identical structure of the green layer in the two cases. In the first case we apply the error on green$_1$, which consists of the green pixels in the odd lines of the Bayer image and on green$_2$, which consists of the green pixels in the even lines. However, in case 3 the error is applied on the whole green layer, which consists of both odd and even lines. The reasons why the error differs in each case will be discussed later, when we add the DCT in the model. Thus, the next step is to find how does the error affect both cases and study the results for further conclusions. The following equation is the full representation of Bayer to f$_1$ conversion and figure 5.1 is the corresponding block diagram:

$$
\begin{aligned}
f_1 = {} & e_1 \Delta(IN\,\Delta IN)(U_1 \Delta U_1)[I_{e1}\Delta(I_e+\varepsilon)](D_1 \Delta D_1)Bayer + \\
& + e_2 \Delta(IN\,\Delta IN)(U_2 \Delta U_2)[I_{e1}\Delta(I_e+\varepsilon)](D_2 \Delta D_2)Bayer + \\
& + e_3 \Delta(IN\,\Delta IN)[(U_2 \Delta U_1)[I_{e1}\Delta(I_e+\varepsilon)](D_2 \Delta D_1) + (U_1 \Delta U_2)[I_{e1}\Delta(I_e+\varepsilon)](D_1 \Delta D_2)]Bayer
\end{aligned}
$$

Where, supposing that Bayer is m^{2}x1, I$_{e1}$ is the identity matrix of size m^2/(4*64)xm^2/(4*64), I$_e$ is the identity matrix of size 64x64 and ε is 64x64 too.

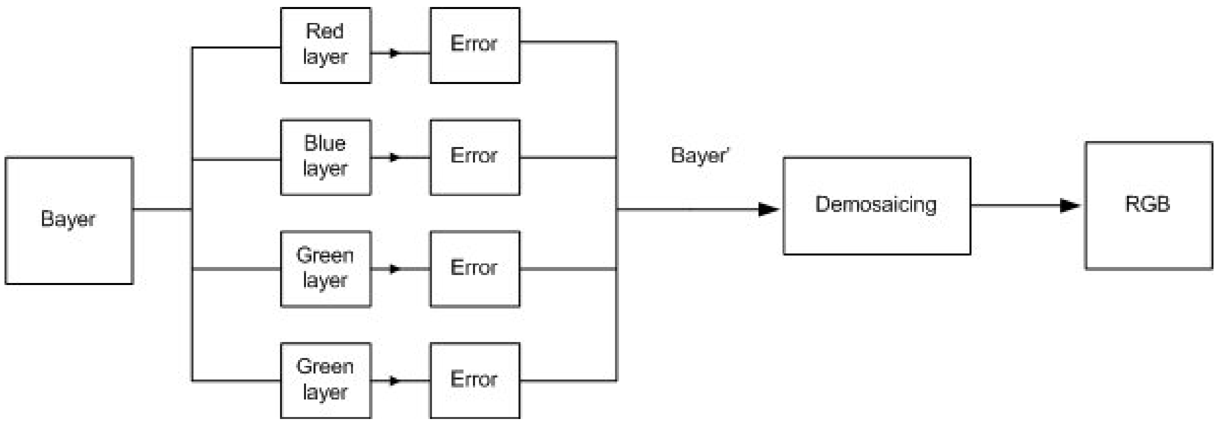


Figure .22-Model case 1 with error

The following equation is the full representation of Bayer to f$_2$ conversion and figure 5.2 is the corresponding block diagram:

$$f_2 = e_1 \Delta(IN \Delta IN)(U_1 \Delta U_1)[I_{e1} \Delta(I_e + \varepsilon)](D_1 \Delta D_1)Bayer +$$
$$+e_2 \Delta(IN \Delta IN)(U_2 \Delta U_2)[I_{e1} \Delta(I_e + \varepsilon)](D_2 \Delta D_2)Bayer +$$
$$+e_3 \Delta(IN \Delta IN)[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)]*$$
$$*[I_{e3} \Delta(I_e + \varepsilon)][(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)]Bayer$$

Where * is the matrix multiplication operation. I_{e1}, I_{e3} and I are identity matrices of size $m^2/(4*64)xm^2/(4*64)$, $m^2/(2*64)xm^2/(2*64)$ and m/2xm/2 respectively. I_e and ε remain the same as before, 64x64.

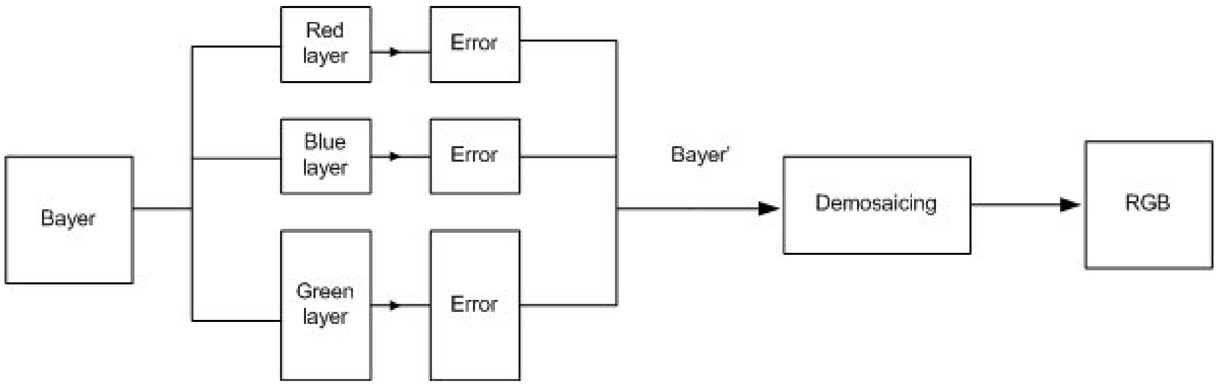


Figure .23-Model case 3 with error

Using the above equations a further analysis can be done on the green layer. This analysis will give useful conclusions about the error and its effect on every case. First of all, we focus on the model case 1 and 3 without taking into consideration the error. The expressions that follow include only the downsampling and upsampling matrices, which have already been described. The left and the right parts of the equation below correspond to the conversion of the Bayer image to the green layer of the RGB using model case 1 and 3, respectively. The first conclusion of the analysis is that the outcome of both equations is the same, which proves that the expressions of the green layer are equal:

$$(U_2 \Delta U_1)(D_2 \Delta D_1) + (U_1 \Delta U_2)(D_1 \Delta D_2) =$$
$$= [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)][(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)]$$

Another conclusion is that both equations are not equal to the identity matrix I, which is expected because with downsampling we lose information that cannot be recovered:

$(U_2 \Delta U_1)(D_2 \Delta D_1) + (U_1 \Delta U_2)(D_1 \Delta D_2)\,\text{Ή}I$

$[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)][(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)]\,\text{Ή}I$

Appendix A.3 provides the proof.

The analysis will go on further by including the error. This time the models give different results. The equation that follows is the expression of the distortion of the green layer for model case 1:

$$distortion = [(U_2 \Delta U_1)(I_{e1} \Delta \varepsilon)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I_{e1} \Delta \varepsilon)(D_1 \Delta D_2)]Bayer$$

The block diagram below presents the green layer in model case 1 when the error ε is applied directly on Bayer:

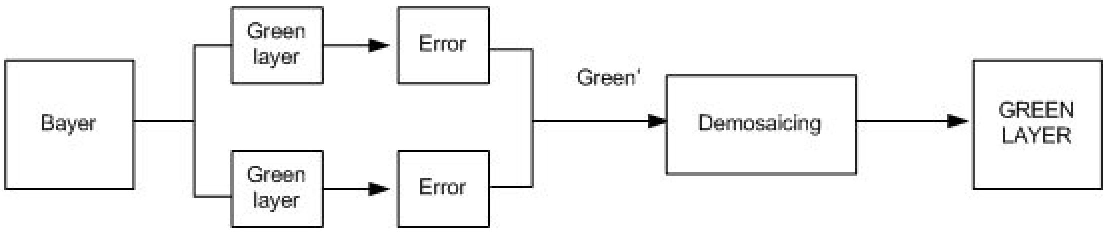


Figure .24-Error of green layer in model case 1

Similarly the equation that follows is the expression of the distortion of the green layer for model case 3:

$$distortion = [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta \varepsilon)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)]Bayer$$

The block diagram below presents the green layer in model case 3 when the error ε is applied directly on Bayer:

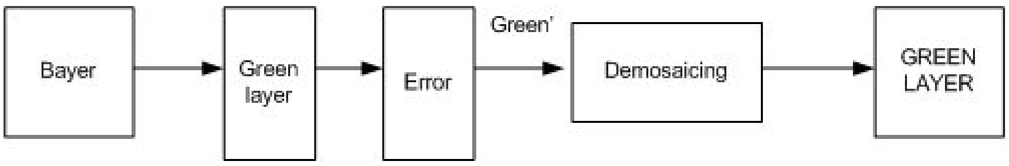


Figure .25-Error of green layer in model case 3

Appendix A.4 presents the detailed procedure that equations 5.3 and 5.4 are created.

Equations 5.3 and 5.4 which represent the distortion of the green layer for model case 1 and 3, respectively, are analyzed further in order to find which model performs best. The model is considered to be expressed as a simple linear system [11] of the type: *Ax=b*, where A stands for the cluster of the demosaicing matrices (downsampling, upsampling), x stands for the Bayer image and b is the outcome, which is an image in RGB format. This is the ideal case when no error exists. However, in this analysis according to our model, we consider that the error affects matrix A and has an impact on the outcome b. The erroneous linear system can be expressed as: *(A+ΔA)x=b+Δb*, where ΔA and Δb express the overall error applied and the outcome due to this error respectively. The error imposed can be isolated with its result in the following equation: *ΔAx=Δb*, which is the one that is studied from now on. The study uses the Frobenius norm [12] in order to find an upper limit of the error in the resulting image. The Frobenius norm of a mxn matrix is expressed as follows:

$$\|A\|_F = \left(\sum_{j=1}^{m} \sum_{i=1}^{n} |a_{ij}|^2 \right)^{1/2} = \|A^T\|_F$$

Study: distortion per Bayer image operation

The goal in the first study is to reveal the general behavior of model case 1 and 3 when an error is applied directly on the green layer of Bayer. Initially a general error matrix ε is considered, which has k errors in the even lines and l errors in the odd lines, and all errors are on the diagonal. With the term image operation we mean that the Frobenius norm of the error is applied on the each operation such as downsampling or upsampling separately. The first expression below is about the model in case 1 and the second is about the model in case 3:

$$\|\Delta b_1\|_F = \|\Delta A_1 x\|_F \le \|\Delta A_1\|_F \|x\|_F \Rightarrow \frac{\|\Delta b_1\|_F}{\|x\|_F} \le \|\Delta A_1\|_F \le$$
$$\le \|U_2 \Delta U_1\|_F \|I_{e1} \Delta \varepsilon\|_F \|D_2 \Delta D_1\|_F + \|U_1 \Delta U_2\|_F \|I_{e1} \Delta \varepsilon\|_F \|D_1 \Delta D_2\|_F = \sqrt{k+l} * m^3/32$$

$$\|\Delta b_2\|_F = \|\Delta A_2 x\|_F \le \|\Delta A_2\|_F \|x\|_F \Rightarrow \frac{\|\Delta b_2\|_F}{\|x\|_F} \le \|\Delta A_2\|_F \le$$
$$\le [\|U_2 \Delta U_1\|_F \|I \Delta D_1\|_F + \|U_1 \Delta U_2\|_F \|I \Delta D_2\|_F] \|I_{e3} \Delta \varepsilon\|_F [\|I \Delta U_2\|_F \|D_1 \Delta D_2\|_F + \|I \Delta U_1\|_F \|D_2 \Delta D_1\|_F] =$$
$$= \sqrt{k+l} * m^5/45$$

Appendix A.5 provides the proof.

From the comparison of the above expressions it is understood that the error in case 3 has always a higher upper bound than that of case 1. This is true when the error is the one stated in this study. However, in this thesis we have already mentioned that the error ε used is the one shown in figure 4.2. So, with the application of this error and according to the previous analysis the upper bounds of both cases are the following:

$$\frac{\|\Delta b_1\|_F}{\|x\|_F} \leq \frac{m^3}{22}$$

$$\frac{\|\Delta b_2\|_F}{\|x\|_F} \leq \frac{m^5}{32}$$

The results in this study are a first hint of the behavior of both cases when the error is applied. However, this comparison cannot yet be taken into consideration because it is not as punctual as expected. This holds due to the fact that the Frobenius norm measures each operation matrix separately but not their combination, which really gives more precise results. Furthermore, it is important to study the behavior of the model in a way that is parallel with the model when we add the DCT.

Study: distortion per odd and even pixels' layer

The goal in the second study is to reveal the general behavior when an error is applied on the green layers of model case 1 and 3. The same error matrix ε, with k errors in the even lines and l errors in the odd lines is considered. This study differs from the previous one because the Frobenius norm is applied on the layer of the green pixels in the odd lines and those in the even lines. The first expression below is about the model in case 1 and the second is about the model in case 3:

$$\frac{\|\Delta b_1\|_F}{\|x\|_F} \leq \|\Delta A_1\|_F \leq$$
$$\leq \|(U_2\, \Delta U_1)(I_{e1}\, \Delta\varepsilon)(D_2\, \Delta D_1)\|_F + \|(U_1\, \Delta U_2)(I_{e1}\, \Delta\varepsilon)(D_1\, \Delta D_2)\|_F =$$
$$= \sqrt{k+l} * m/8$$

$$\frac{\|\Delta b_2\|_F}{\|x\|_F} \le \|\Delta A_2\|_F \le$$
$$\le \|(U_2\ \Delta U_1)(I\ \Delta D_1)(I_{e3}\ \Delta\varepsilon)(I\ \Delta U_1)(D_2\ \Delta D_1)\|_F + \|(U_1\ \Delta U_2)(I\ \Delta D_2)(I_{e3}\ \Delta\varepsilon)(I\ \Delta U_2)(D_1\ \Delta D_2)\|_F =$$
$$= (\sqrt{k} + \sqrt{l}) * m/11.2$$

Appendix A.6 provides the proof.

From the comparison of the above expressions it is understood that the error in case 3 has always a lower upper bound than that of case 1. This is true when the error is the one stated in this study. However, in this thesis we have already mentioned that a specified error is used, the one shown in figure 4.2. So applying error ε and according to the previous analysis the upper bounds of both cases are the following:

$$\frac{\|\Delta b_1\|_F}{\|x\|_F} \le \frac{m}{5.7}$$
$$\frac{\|\Delta b_2\|_F}{\|x\|_F} \le \frac{m}{5.6}$$

In this study the results of the comparison contradict to the ones of the previous. This was something expected, although the results in this study are still away from the real ones, because it is considered that if the DCT existed every element would not be of the same importance. This is a mistake analyzed and corrected in the next study.

Study: distortion of weighted error

In the previous studies when measuring the upper bound of the error of each case, it was supposed that the importance of every element of the 64x1 subimage, on which the error was applied, was equal at all location. However, if one intends to adopt the behavior of the DCT, this is not exactly representative. As already mentioned, the DCT has a strong energy compaction property as most of the signal information tends to be concentrated in a few low frequency components of the DCT. The lowest frequency component is the (0,0) element, which is the most important, and entries with increasing vertical and horizontal index values represent higher frequencies, which hold little information so they are less important. As a result, it is not correct to treat an error in a specific element of the 64x1 subimage the same way with another one that has a higher index value. To solve this problem a matrix of weights from 1 to 0 is introduced, with the first element of the subimage being the most important and the last one being the least important. The matrix of weights replaces the role of the DCT as far as it concerns the importance of the error imposed. Although this method does not represent exactly each element's significance we find it to be satisfactory as it still is a method of ranking. Using the matrix of weights it is aimed to make a better approach to the error generated in each case. In this study the error of figure 4.2 is used.

The matrix of weights is applied on the Frobenius norm; both of them are shown below:

$$w = \begin{bmatrix} 1 & \frac{63}{64} & \frac{62}{64} & \frac{61}{64} & \cdots & \frac{1}{64} \end{bmatrix}$$

$$\|A\|_F = \left(\sum_{j=1}^{m} \sum_{i=1}^{n} w_i |a_{ij}|^2 \right)^{1/2} = \|A^T\|_F$$

The first expression that follows represents the error in model case 1 and the second expression the error in model case 3:

$$\frac{\|\Delta b_1\|_F}{\|x\|_F} \le \|\Delta A_1\|_F \le$$
$$\le \|(U_2 \; \Delta U_1)(I_{e1} \; \Delta\varepsilon)(D_2 \; \Delta D_1)\|_F + \|(U_1 \; \Delta U_2)(I_{e1} \; \Delta\varepsilon)(D_1 \; \Delta D_2)\|_F =$$
$$= \frac{\sqrt{3} * m}{64} \simeq 0.02m$$

$$\frac{\|\Delta b_2\|_F}{\|x\|_F} \le \|\Delta A_2\|_F \le$$
$$\le \|(U_2 \; \Delta U_1)(I \; \Delta D_1)(I_{e3} \; \Delta\varepsilon)(I \; \Delta U_1)(D_2 \; \Delta D_1)\|_F + \|(U_1 \; \Delta U_2)(I \; \Delta D_2)(I_{e3} \; \Delta\varepsilon)(I \; \Delta U_2)(D_1 \; \Delta D_2)\|_F =$$
$$= \frac{(\sqrt{33} + \sqrt{35}) * m}{128} \simeq 0.09m$$

Appendix A.7 provides the proof.

Obviously, the error in case 3 has a higher upper bound than that of case 1. There is also a profound difference at the results of this study because of the usage of weights.

The difference can be justified in more detail if we analyze visually the impact of the weighted error on the model. By comparing the expressions of the models it is observed that in case 1 there is the product of the upsampling, the error and the downsampling matrices. In case 3 there are two additional matrices before and after the error matrix. The interesting part is the one where they differ:

$$case1: (I_{e1} \Delta \varepsilon) + (I_{e1} \Delta \varepsilon)$$
$$case3: (I \Delta I_0)(I_{e3} \Delta \varepsilon)(I \Delta I_0)^T + (I \Delta I_1)(I_{e3} \Delta \varepsilon)(I \Delta I_1)^T$$

The visual representation of the error matrix in case 1 is:

$$I_{e1} \Delta \varepsilon = \begin{bmatrix} 0 & \cdots & 0 & 0 & & & & & \\ \vdots & \ddots & & \vdots & & & & & \\ 0 & & -1 & & & & & & \\ 0 & \cdots & & -1 & & & & & \\ & & & & 0 & \cdots & 0 & 0 & \\ & & & & \vdots & \ddots & & \vdots & \\ & & & & 0 & & -1 & & \\ & & & & 0 & \cdots & & -1 & \cdots \\ & & & & & & & \vdots & \end{bmatrix}$$

This matrix is a direct result of the Kronecker product. Each block has size 64x64 and the whole matrix has size $m^2/4 \text{x} m^2/4$.

The visual representation of the error matrix in case 3 is:

$$(I\,\Delta I_1)(I_{e3}\,\Delta\varepsilon)(I\,\Delta I_1)^T = \begin{bmatrix} 0 & \cdots & 0 & 0 & & & & & \\ \vdots & \ddots & & \vdots & & & & & \\ 0 & & 0 & & & & & & \\ 0 & \cdots & & -1 & & & & & \\ & & & & 0 & \dots & 0 & 0 & \\ & & & & \vdots & \ddots & & \vdots & \\ & & & & 0 & & 0 & & \\ & & & & 0 & \dots & & -1 & \cdots \\ & & & & & & & \vdots & \end{bmatrix}$$

$$(I\,\Delta I_0)(I_{e3}\,\Delta\varepsilon)(I\,\Delta I_0)^T = \begin{bmatrix} 0 & \cdots & 0 & 0 & & & & & \\ \vdots & \ddots & & \vdots & & & & & \\ 0 & & -1 & & & & & & \\ 0 & \cdots & & 0 & & & & & \\ & & & & 0 & \dots & 0 & 0 & \\ & & & & \vdots & \ddots & & \vdots & \\ & & & & 0 & & -1 & & \\ & & & & 0 & \dots & & 0 & \cdots \\ & & & & & & & \vdots & \end{bmatrix}$$

The matrix $(I_{e3}\,\Delta\varepsilon)$ has size m^2/2xm^2/2, but the other two matrices perform downsampling in both rows and columns, so the matrices $(I\,\Delta I_0)(I_{e3}\,\Delta\varepsilon)(I\,\Delta I_0)^T$ and $(I\,\Delta I_1)(I_{e3}\,\Delta\varepsilon)(I\,\Delta I_1)^T$ have size m^2/4xm^2/4. In contrast with case 1 the blocks shown have size 32x32. To conclude, after this analysis it is clear that in case 1 the errors are enforced in elements 63 and 64, though in case 3 the errors are enforced in elements 32 and 64 or 31 and 63. Because of the downsampling on the error matrix, the number of errors in both cases is equal, though the error in case 3 has a higher upper bound because half of its elements are more important.

Study: distortion of weighted error on full layer

This is the last study of this section which is similar to the previous one. The only thing changed is the measurement of the upper bound of the error in both cases. This time the error is calculated on the full colour layer. The matrix of weights is used again in order to distinguish the importance of each element. The first expression that follows represents the error in model case 1 and the second one the error in model case 3:

$$\frac{\|\Delta b_1\|_F}{\|x\|_F} \le \|\Delta A_1\|_F \le \|(U_2\ \Delta U_1)(I_{e1}\ \Delta\varepsilon)(D_2\ \Delta D_1) + (U_1\ \Delta U_2)(I_{e1}\ \Delta\varepsilon)(D_1\ \Delta D_2)\|_F =$$

$$= \frac{\sqrt{6} * m}{128} \simeq 0.0191m$$

$$\frac{\|\Delta b_2\|_F}{\|x\|_F} \le \|\Delta A_2\|_F =$$

$$\|[(U_2\ \Delta U_1)(I\ \Delta D_1) + (U_1\ \Delta U_2)(I\ \Delta D_2)](I_{e3}\ \Delta\varepsilon)[(I\ \Delta U_2)(D_1\ \Delta D_2) + (I\ \Delta U_1)(D_2\ \Delta D_1)]\|_F =$$

$$= \frac{\sqrt{68} * m}{128} \simeq 0.0884m$$

Appendix A.8 provides the proof.

Obviously, the error in case 3 has a higher upper bound than that of case 1. There is also a profound difference at the results of this study because of the usage of weights. The difference existing can be justified by the visual analysis of the previous study. The approach of this study is the most punctual because the error is calculated on the whole green layer. The figure below shows the error in relation with the size of the image (mxm):

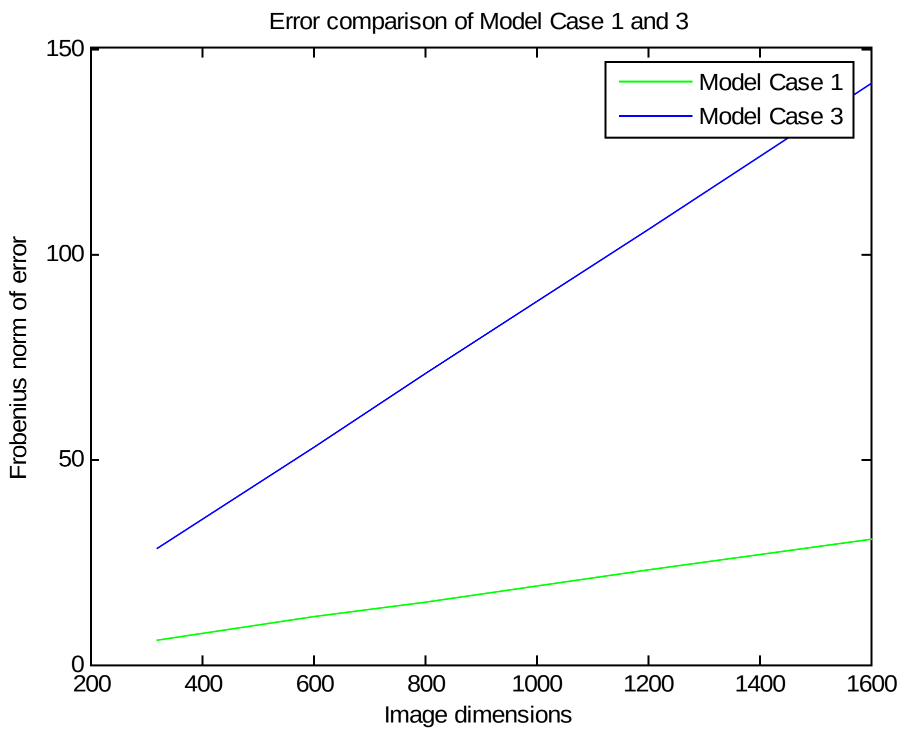


Figure .26-Error comparison of model case 1 and 3

This figure reveals that the distortion in the green layer of model case 3 is higher than the one of model case 1 and their difference grows bigger as the image gets larger.

This sequence of studies has lead us to focus on the real error, insert the weight matrix in the Frobenius norm in order to simulate the behavior of the DCT and make a visual representation of the error matrices in both cases. The most important is that we have gathered useful information about the model and have prepared to study it with the DCT.

5.3 Bayer to RGB conversion with error on DCT

In the second part of Error analysis section, the Bayer to RGB conversion models are examined supposing that the transformation, which is the DCT, is applied on the Bayer image, as shown in figure 4.1. The transformation involves the error ε, which is the one described by figure 4.2. The goal in this section is to examine the difference and a possible connection between the RGB images that result from the implementations of model case 1 and 3. The existence of DCT makes this analysis more complex than the previous one, due to the fact that in each case it is applied on different data. The procedure followed for model case 1 is first to downsample Bayer, which gives the red, $green_1$, $green_2$ and blue layers, apply DCT on them and before applying IDCT the error is enforced. The RGB image that is created by model case 1 is f_1 and is affected by the error. The procedure followed for model case 3 is first to downsample Bayer, which gives the red, $green_1$, $green_2$ and blue layers, upsample green layers in lines so as to combine them to one. Then apply DCT on the green layer and before applying IDCT the error is enforced. If the same error is applied on model case 3, there will be an f_2 image with major differences compared to f_1. These differences are a result of the nonidentical structure of the green layer in the two cases. As mentioned earlier in the first case the error is applied on the DCT conversion of $green_1$, which consists of the green pixels in the odd lines of the Bayer image, and on the DCT conversion of $green_2$, which consists of the green pixels in the even lines. However, in case 3 the error is applied on the DCT conversion of the whole green layer which consists of both odd and even lines. We realize that in the

previous study, error had an impact only on some pixels of the image. On the contrary, in this study because of the DCT and mainly the IDCT conversion, the error diffuses in the whole image. So the next step is to find how does the error affect both cases and to study the results for further conclusions.
The equations of the full representations of the conversion of Bayer to f_1 and f_2 respectively and the corresponding block diagrams are given below:

$$\begin{aligned} f_1 = {} & e_1 \Delta (IN \Delta IN)(U_1 \Delta U_1)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_1 \Delta D_1) Bayer + \\ & + e_2 \Delta (IN \Delta IN)(U_2 \Delta U_2)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_2 \Delta D_2) Bayer + \\ & + e_3 \Delta (IN \Delta IN)[(U_2 \Delta U_1)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_2 \Delta D_1) + \\ & + (U_1 \Delta U_2)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_1 \Delta D_2)] Bayer \end{aligned}$$

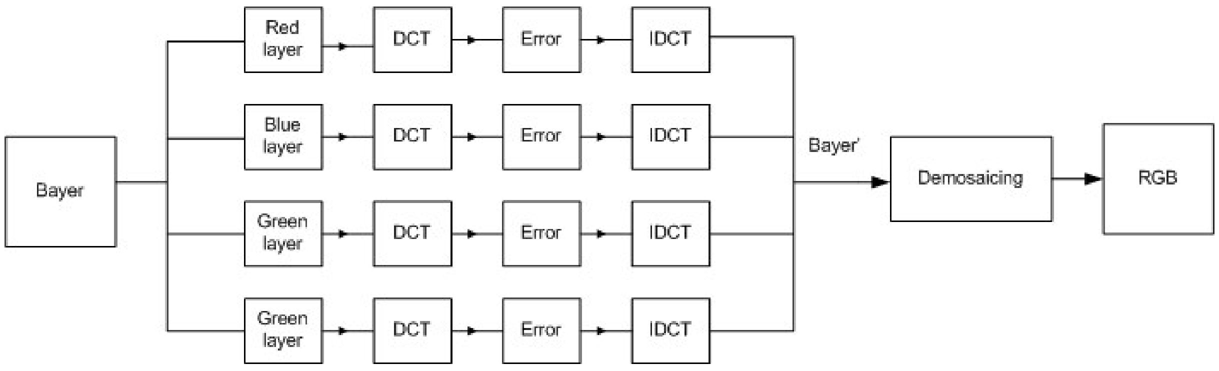


Figure .27-Model case 1with error on DCT

$$\begin{aligned} f_2 = {} & e_1 \Delta (IN \Delta IN)(U_1 \Delta U_1)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_1 \Delta D_1) Bayer + \\ & + e_2 \Delta (IN \Delta IN)(U_2 \Delta U_2)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT)(D_2 \Delta D_2) Bayer + \\ & + e_3 \Delta (IN \Delta IN)[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta IDCT)[I_{e3} \Delta (I_e + \varepsilon)] * \\ & * (I_{e3} \Delta DCT)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)] Bayer \end{aligned}$$

Where * is the matrix multiplication operation

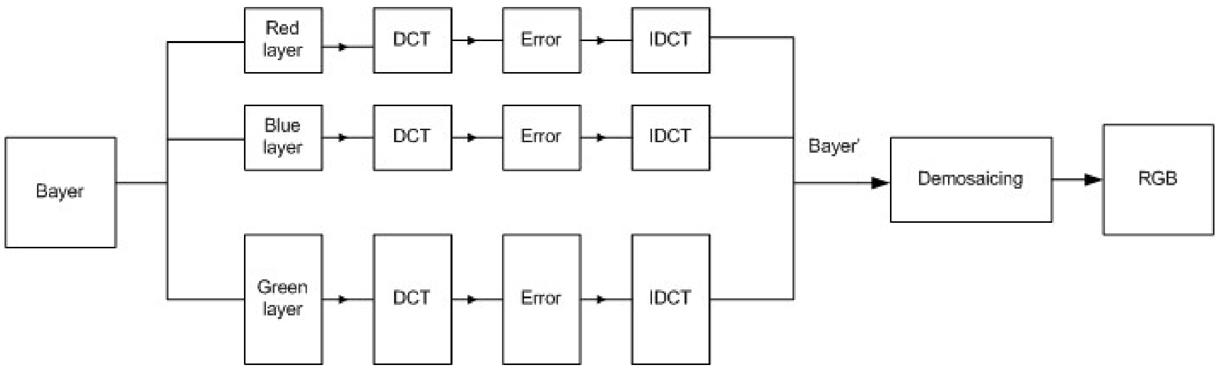


Figure .28-Model case 3 with error on DCT

The matrices used in equations 5.5 and 5.6 remain as described in the section 4.3. Using the above equations a further analysis can be done on the green layer. This analysis gives useful conclusions about the error and its effect on every Bayer to RGB conversion implementation. The equation that follows is the error expression of the green layer of model case 1:

$$distortion = [(U_2 \Delta U_1)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(D_2 \Delta D_1) +$$
$$+(U_1 \Delta U_2)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(D_1 \Delta D_2)]Bayer$$

The block diagram of the above equation is:

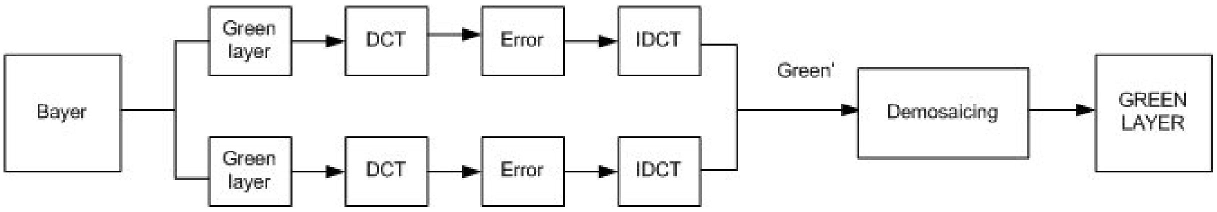


Figure .29-Error on DCT of green layer in model case 1

The equation that follows is the error expression of the green layer of model case 3:

$$distortion = [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta IDCT)(I_{e3} \Delta \varepsilon) *$$
$$*(I_{e3} \Delta DCT)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)]Bayer$$

The block diagram of the above equation is:

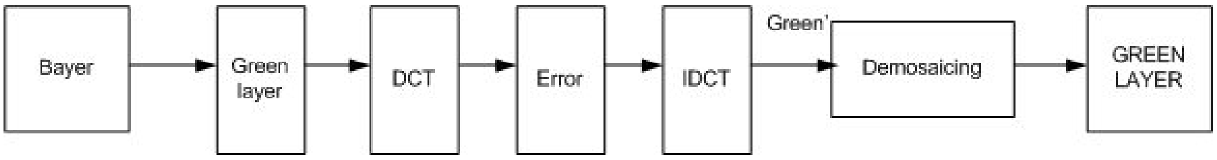


Figure .30-Error on DCT of green layer in model case 3

Appendix A.9 provides the detailed representation of the error in both cases.

Equations 5.7 and 5.8 will be further examined in order to compare the distortion generated on the green layer of both implementations. Once again the model is expressed as the simple linear system of type: *Ax=b*, which was described in the previous section. The equation of the error and its result *ΔAx=Δb* are studied in this section. The study initially uses the Frobenius norm in order to find an upper limit of the error in the resulting image. The matrix of weights is not used any more as its role is replaced by the DCT transformation.

Study: distortion on full layer

The goal of this study is to reveal the behavior of the model in both cases when the error ε is applied on the DCT. The Frobenius norm is applied on the whole green layer. The first expression is about the model in case 1 and the second is about the model in case 3:

$$\frac{\|\Delta b_1\|_F}{\|x\|_F} \leq \|\Delta A_1\|_F = \left\| \begin{matrix} (U_2 \Delta U_1)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(D_2 \Delta D_1) + \\ +(U_1 \Delta U_2)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(D_1 \Delta D_2) \end{matrix} \right\|_F$$

$$\frac{\|\Delta b_2\|_F}{\|x\|_F} \leq \|\Delta A_2\|_F = \left\| \begin{matrix} [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta IDCT)(I_{e3} \Delta \varepsilon) * \\ *(I_{e3} \Delta DCT)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)] \end{matrix} \right\|_F$$

Appendix A.10 provides a detailed analysis of the distortion measurement.

The Frobenius norms above cannot be computed as easily as before (as a factor of the size of the image), because with the introduction of the DCT conversion and its inverse, the matrices are no longer diagonal and the error is spread in the whole image by the coefficients of IDCT.
Another approach needs to be made in order to get more useful conclusions about the error of the two model cases.

5.3.1 Spectral analysis

As it is already mentioned the DCT transformation has a strong energy compaction property. This means that it gathers the energy of the signal in the low frequency components, which are the first elements of the DCT matrix. Elements with higher index values represent the high frequencies of the signal and tend to hold less signal information. Taken as granted this property of the DCT, the spectral analysis is expected to give a more solid explanation of the effect of the error on both model cases.

The behavior of the model case 1 is examined for a single 8x8 image through matrix analysis for one of the two summands of equation 5.7:

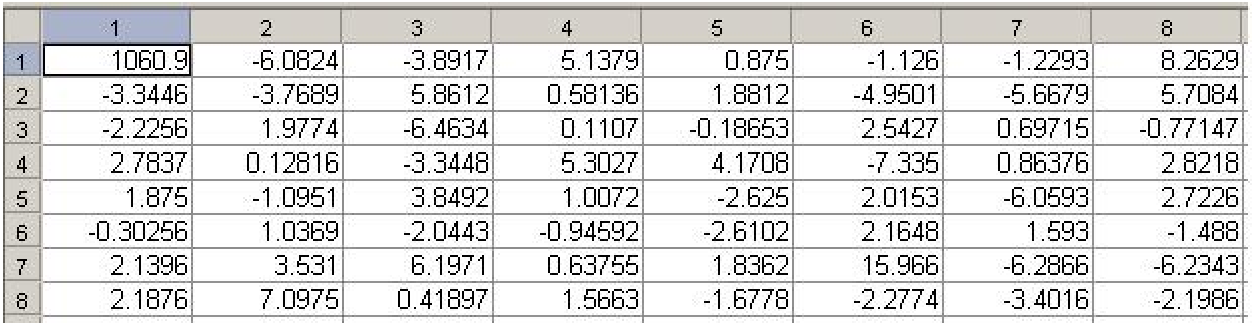
$$(U_1 \Delta U_2)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(D_1 \Delta D_2)Bayer =$$

$$= (U_1 \Delta U_2)(I_{e1} \Delta IDCT)\begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & -1 & \\ 0 & \cdots & & -1 \end{bmatrix}\begin{bmatrix} x & \cdots & x & x \\ \vdots & \ddots & & \vdots \\ x & & x & \\ x & \cdots & & x \end{bmatrix}\begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_{31} \\ g_{32} \\ g_{33} \\ \vdots \\ g_{64} \end{bmatrix} =$$

$$= (U_1 \Delta U_2)(I_{e1} \Delta IDCT)\begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & -1 & \\ 0 & \cdots & & -1 \end{bmatrix}\begin{bmatrix} {}_1 \\ {}_2 \\ \vdots \\ {}_{31} \\ {}_{32} \\ {}_{33} \\ \vdots \\ {}_{64} \end{bmatrix}$$

Where the Bayer image is an 16x16 matrix, the Downsampling matrices are 8x16, the Upsampling are 16x8 and I_{e1} is 1x1.

The above scheme represents the downsampled green$_1$ layer in vector form (64x1), which is then converted to its DCT transformation of size 64x1. The error, which is of size 64x64, is applied on the DCT and affects its last two elements. This analysis is the same for every 8x8 subimage and for the other summand of equation 5.7. The spectrum for a subimage in 2-D form is (the numbers may differ depending on the image data but the change is relative for all elements):



	1	2	3	4	5	6	7	8
1	1060.9	-6.0824	-3.8917	5.1379	0.875	-1.126	-1.2293	8.2629
2	-3.3446	-3.7689	5.8612	0.58136	1.8812	-4.9501	-5.6679	5.7084
3	-2.2256	1.9774	-6.4634	0.1107	-0.18653	2.5427	0.69715	-0.77147
4	2.7837	0.12816	-3.3448	5.3027	4.1708	-7.335	0.86376	2.8218
5	1.875	-1.0951	3.8492	1.0072	-2.625	2.0153	-6.0593	2.7226
6	-0.30256	1.0369	-2.0443	-0.94592	-2.6102	2.1648	1.593	-1.488
7	2.1396	3.531	6.1971	0.63755	1.8362	15.966	-6.2866	-6.2343
8	2.1876	7.0975	0.41897	1.5663	-1.6778	-2.2774	-3.4016	-2.1986

Table .1-Spectrum of an 8x8 subimage in model case 1

From the above matrix it is observed that the energy of the signal is indeed gathered in the first elements and mainly the very first one. The two last elements affected by the error hold very little information, so the error generated is minimal. A similar behavior is expected by all other subimages and by the second summand. The spectrum of the signal in 3-d representation is the following:

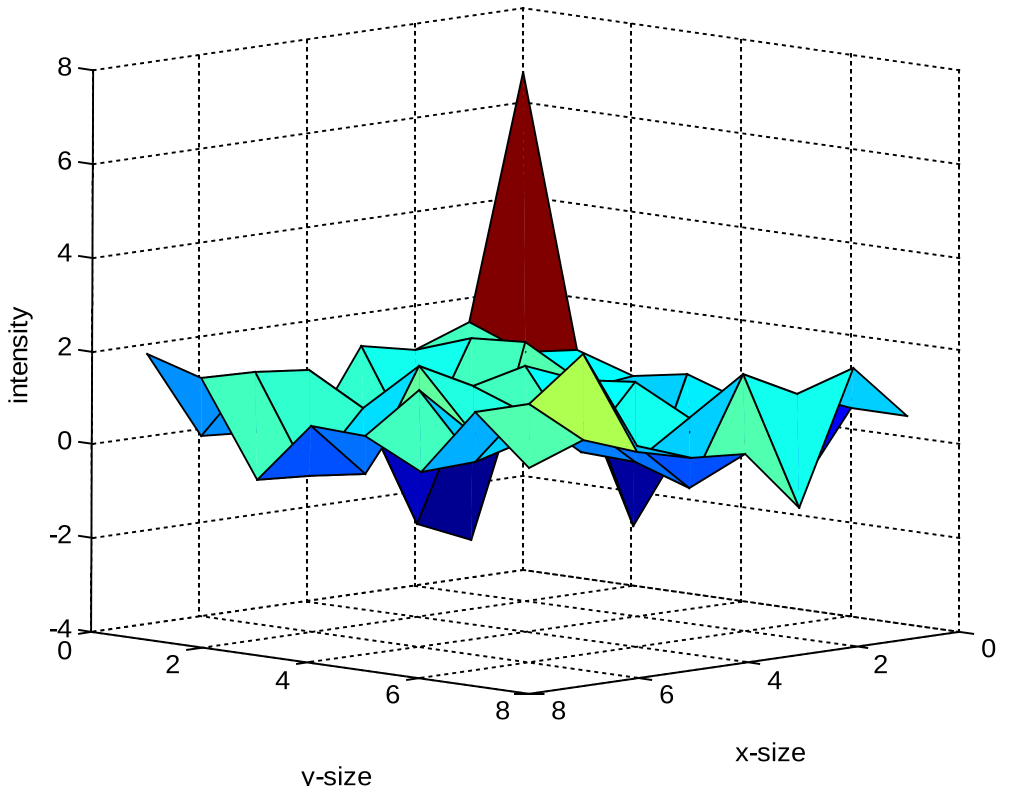


Figure .31- Spectrum of an 8x8 subimage in model case 1 (log scale)

This representation shows that the intensity of the spectrum is high in the first element of the DCT matrix and very low (close to zero) elsewhere.

The behavior of the model case 3 is examined for a single 8x8 image through matrix analysis:

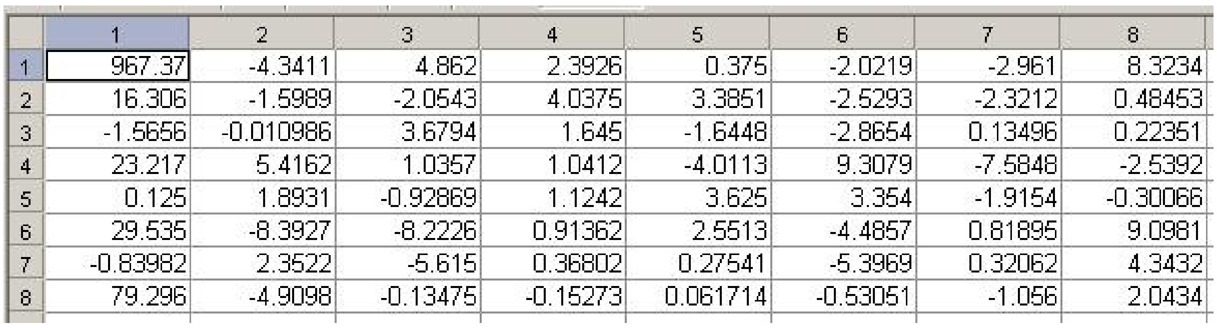
$$[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta IDCT)(I_{e3} \Delta \varepsilon)(I_{e3} \Delta DCT)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)]Bayer =$$

$$= [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta IDCT)\begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & -1 & \\ 0 & \cdots & & -1 \end{bmatrix}\begin{bmatrix} x & \cdots & x & x \\ \vdots & \ddots & & \vdots \\ x & & x & \\ x & \cdots & & x \end{bmatrix}\begin{bmatrix} g_{1-1} \\ g_{2-1} \\ g_{1-2} \\ g_{2-2} \\ g_{1-3} \\ g_{2-3} \\ \vdots \\ g_{1-32} \\ g_{2-32} \end{bmatrix} =$$

$$= [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta IDCT)\begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & -1 & \\ 0 & \cdots & & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \\ \vdots \\ 31 \\ 32 \\ 33 \\ \vdots \\ 64 \end{bmatrix}$$

Where the Bayer image is an 16x16 matrix, the Downsampling matrices are 8x16, the Upsampling are 16x8, I is 8x8 and I_{e3} is 2x2.

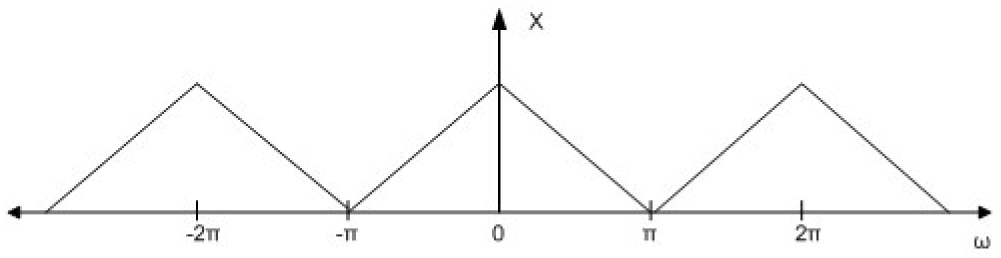
The above scheme shows an 8x8 image in its vector form (64x1). The input data this time are a combination of 32 green pixels of the odd lines with 32 green pixels of the even lines of the Bayer image. In the above equation the green pixels of the odd lines are represented by g_{1-k} and those of the even lines are represented by g_{2-k}. The DCT is applied to the input data and creates a 64x1 matrix. The error is afterwards applied on the last two elements of the DCT transformation. The error affects the same elements of the DCT (63 and 64) as in model case 1 and the total number of errors imposed in the whole image is the same too. In an attempt to compare the error of the two model cases the importance of the erroneous elements is analyzed through spectral analysis. The following table presents the spectrum of an 8x8 subimage in the 2-D form:

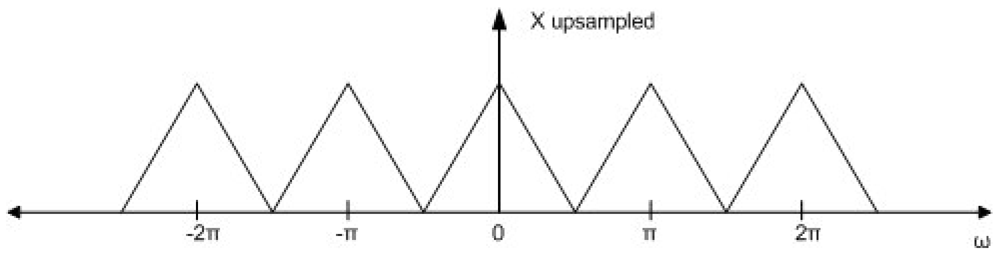


	1	2	3	4	5	6	7	8
1	967.37	-4.3411	4.862	2.3926	0.375	-2.0219	-2.961	8.3234
2	16.306	-1.5989	-2.0543	4.0375	3.3851	-2.5293	-2.3212	0.48453
3	-1.5656	-0.010986	3.6794	1.645	-1.6448	-2.8654	0.13496	0.22351
4	23.217	5.4162	1.0357	1.0412	-4.0113	9.3079	-7.5848	-2.5392
5	0.125	1.8931	-0.92869	1.1242	3.625	3.354	-1.9154	-0.30066
6	29.535	-8.3927	-8.2226	0.91362	2.5513	-4.4857	0.81895	9.0981
7	-0.83982	2.3522	-5.615	0.36802	0.27541	-5.3969	0.32062	4.3432
8	79.296	-4.9098	-0.13475	-0.15273	0.061714	-0.53051	-1.056	2.0434

Table .2-Spectrum of an 8x8 subimage in model case 3

The above matrix reveals that there is a profound difference between the spectrum of case 1 and case3. The energy of the signal in case 3 is not gathered completely at the first elements of the DCT, though there are other high energy elements, the ones in positions (2, 1), (4, 1), (6, 1) and (8, 1). This is a result of the upsampling operation on the green layer [14]. In more detail, let's assume that the signal X in the frequency domain is given by the first of the following figures:





It is observed that signal X is repeated every 2π. The second figure shows that if the signal is upsampled by a factor of two, then its periodicity becomes every π instead of 2π. This happens because when upsampling, the objective is to obtain samples every T'=T/L, where T and T' are the periodicities of the original signal and the upsampled, respectively and L is the increase in sampling rate. This is exactly what happens in the green layer situation too. Although the green layer is a 2-d signal, the rational stays the same. It is upsampled in one direction, thus the periodicity of its spectrum in this direction increases, however in the other direction the periodicity stays the same. This explanation justifies the existence of high energy elements in positions (2, 1), (4, 1), (6, 1) and (8, 1). The 3-d representation of the spectrum of the signal of every 8x8 subimage in model case 3 is the following:

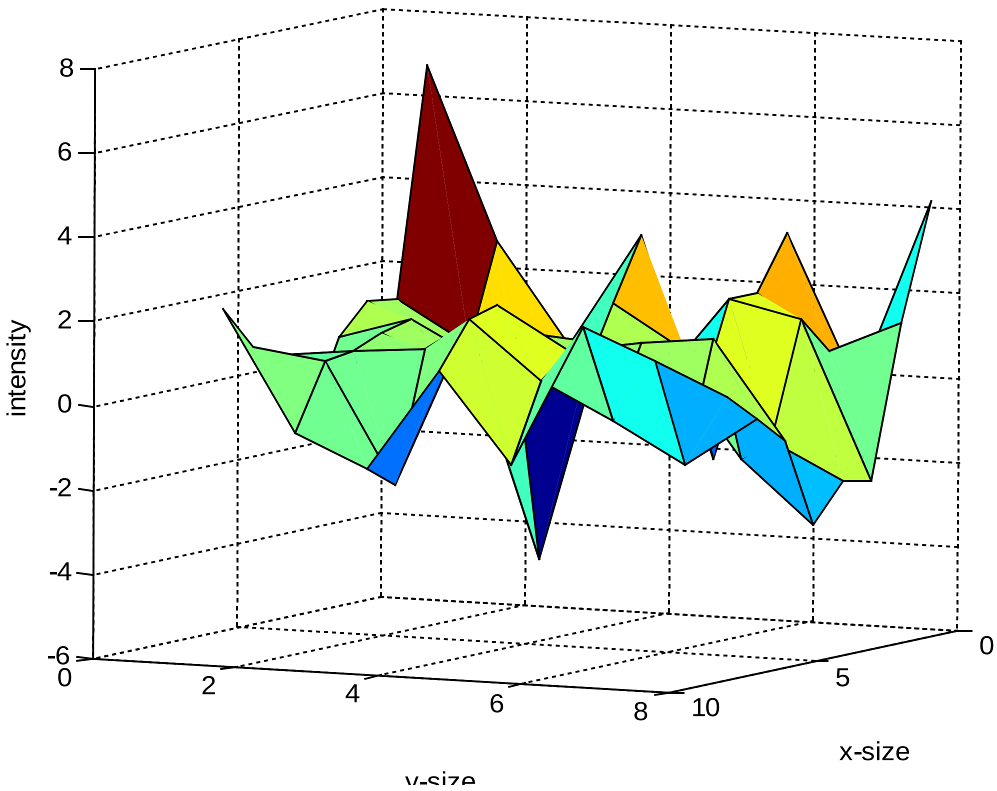


Figure .32-Spectrum of an 8x8 subimage in model case 3 (log scale)

Figure .32 shows graphically the insertion of highpass information in the signal as some high frequency elements hold more energy than in model case 1.

After having made the first observations the explanation of this behavior is needed. The energy concentrated in the high frequency elements is a result of the combination of the green pixels of the odd lines with the ones of the even lines. This happens because the green pixels of the even lines have been shifted one position to the left. It is emphasized that this fact actually inserts high pass information in the image. So when computing the DCT conversion of the whole green layer it is expected to get some energy of it in elements that represent the high frequencies of the signal. As already mentioned, the error in both cases affects the same DCT coefficients; from the above analysis, though, it is clear that the signal information that these coefficients hold is not the same in both cases. In case 3 these coefficients are additionally affected by the high frequency elements in positions (2, 1), (4, 1), (6, 1) and (8, 1). The following figure shows the way that the energy of the signal is distributed in every 8x8 DCT transformation for both implementations:

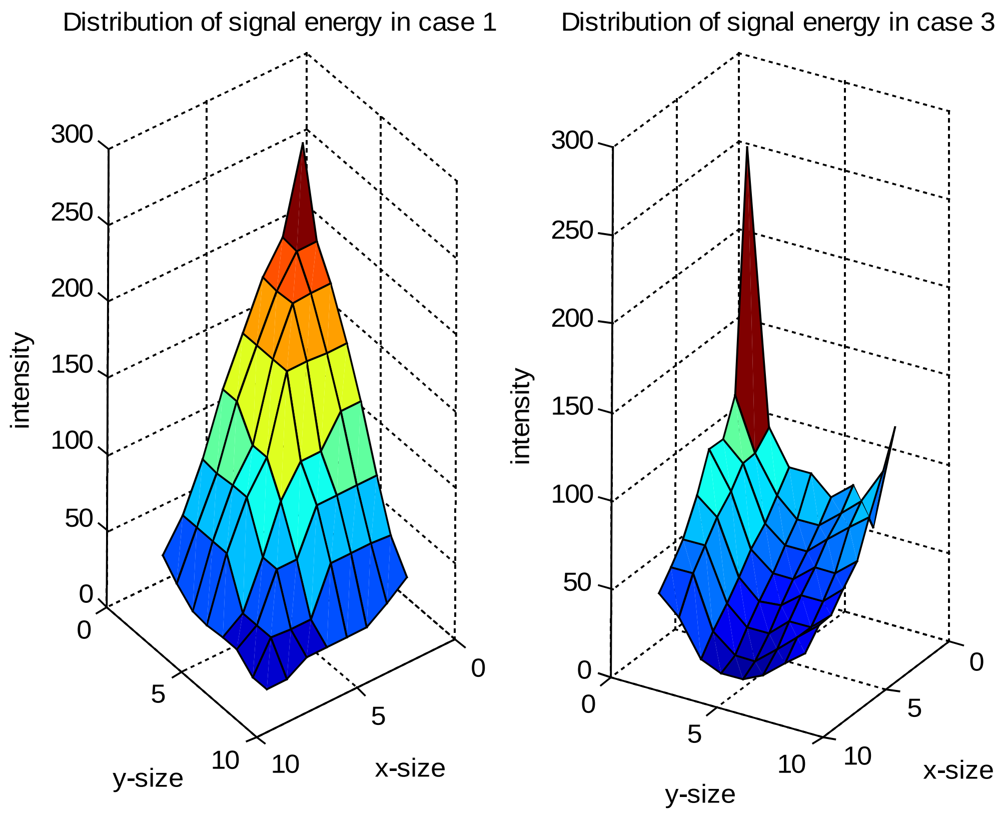


Figure .33-Distribution of signal energy in both model cases

In model case 1 there is one lobe with high energy in position (1, 1) which affects the other elements less and less as their index values get higher. The element with index value (8, 8) and the ones around it are the least affected and hold little energy. This is the exact explanation why model case 1 performs best when applying the error ε of figure 4.2. In model case 3 there are two lobes with high energy, one in position (1, 1) and one in (8, 1). Additionally elements between them hold much signal information too. These elements affect the other coefficients less and less as their index values get higher. However this time the graphic representation reveals that the least affected elements are the ones with index values (4, 8) and (5, 8) and this is the explanation why model case 3 does not perform so well when the specified error is applied.

5.3.2 Experimental measurements

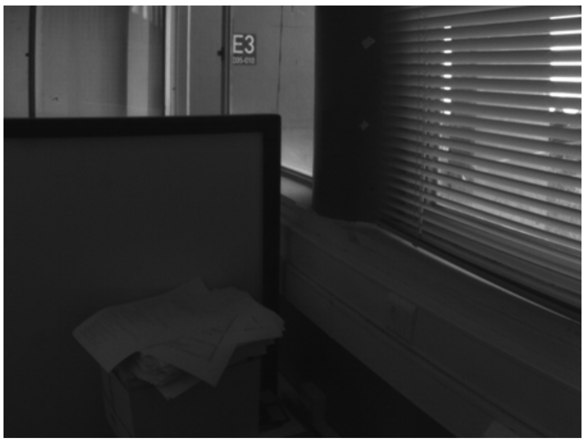
In this section we present the experimental results and the conclusions that originate from them. All of the images tested have resolution 1600x1200 with 8 bits per pixel and some of them are taken in the DISPLAY laboratory. All input images are stored as BMP images and they are grayscale because they are in Bayer format. They have been selected because they reflect different signal activity, high-frequency content and colour intensities. Each image has been compared with the ideal one, which is considered to be the image converted to DCT and its inverse, without applying any error. The measurement of the error is done by subtracting the erroneous image with the ideal, pixel by pixel. L_2 norm is then used to compute the total error of that image. L_2 norm is given by:

$$L_2 = \left(\sum_{j=1}^{m} \sum_{i=1}^{n} \left| x_{ij} \right|^2 \right)^{1/2}$$

The images used for measurements are the following:

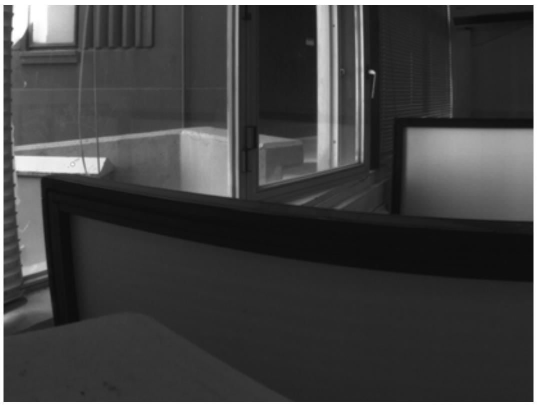
Source Image 0

Source Image 1

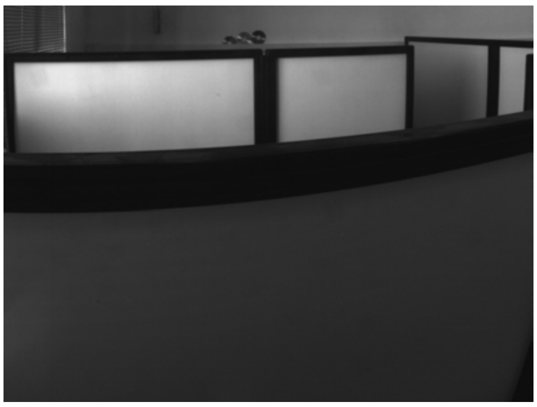


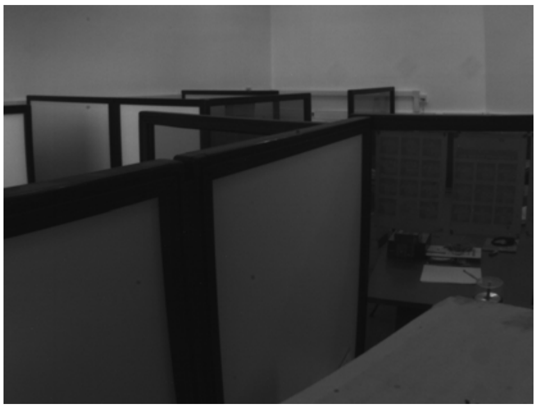
Source Image 2

Source Image 3

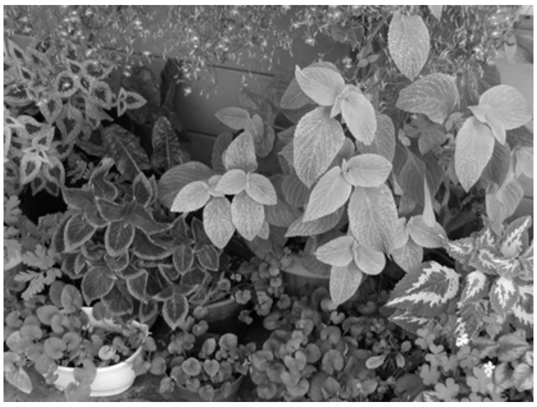


Source Image 4 **Source Image 5**





Source Image 6 **Source Image 7**



Source Image 8 **Source Image 9**

WORMS3D
http://www.worms3d.com

Source Image 10

The measurements shown in the table below are results of the eleven different input images. Each image has been tested twice, once for each model case. On the left hand side of the table is the image tested and on the right hand side is the total error of the respective image. The error applied in this comparison is the one of figure 4.2.

Image No	L_2 Norm	
	CASE 1	**CASE 3**
Image0	858	922
Image1	960	1020
Image2	819	864
Image3	964	1051
Image4	1018	1104
Image5	834	963
Image6	858	920
Image7	842	887
Image8	22683	22793
Image9	35663	38328
Image10	16542	16706

Table .3-Distortion comparison with error on DCT

Table 5.3 confirms the results of the spectrum analysis of the previous section as all images have bigger error in case 3 than in case 1. At the same time, it is observed that among images, which are taken in the lab, 3 and 4 have a higher error value than the others. This happens due to the fact that apart from the type of error, a key role holds the image itself; for instance its frequency content. In case there are many edges the image holds highpass information, which is gathered in the high frequency elements of its DCT transformation. This results in high image distortion, when applied an error in the high frequency elements of the DCT. It is exactly what happens with images 3 and 4, where it is observed that the error is approximately 5-10% higher than that of the rest images. This issue is also explained by the following figures of model case 1:

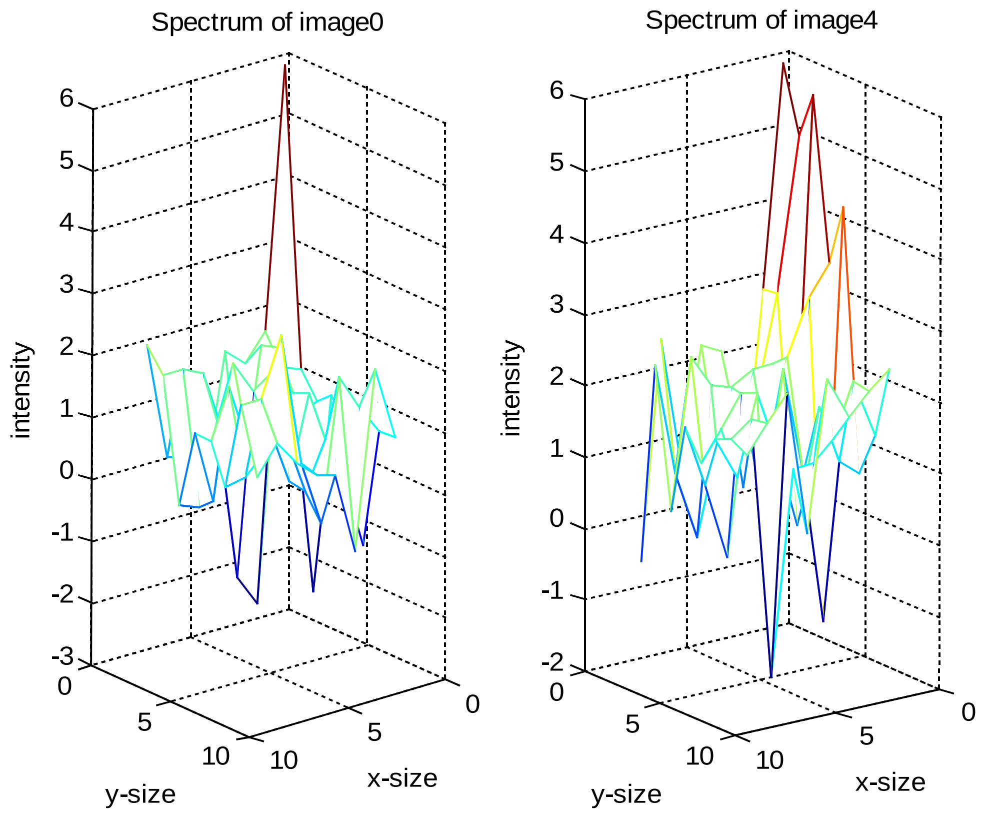


Figure .34-Comparison of the Spectrum of images without and with edges (log scale)

The above figure shows that for model case 1, the main part of the image information is gathered in the first element as far as it concerns image 0, though in image 4 the image information not only expands in the y dimension of its spectrum, but at all location too. This happens because image 4 has edges in the y dimension. The expansion of the image energy results in higher distortion when the error is applied on high frequency elements. The same holds for image 3 and for model case 3. Another noticeable fact in table 5.3 is the increased distortion of the last three images. This happens because of two reasons. Firstly, there are much more edges in these images than in the ones taken in the laboratory and secondly, the difference in the intensity of the colours is extremely high. These two facts result in the generation of considerably higher distortion than the other images.

The output images of both implementations are in RGB format, thus they are coloured. Their resolution remains the same 1600x1200 however each pixel is represented by 24 bits of information (8 for every colour). Some of the output images of both models are shown below:

Output image 0-case 1

Output image 0-case 3

Output image 1-case 1

Output image 1-case 3

Output image 2-case 1

Output image 2-case 3

Output image 8-case 1

Output image 8-case 3

Output image 9-case 1

Output image 9-case 3

WORMS 3D
http://www.worms3d.com
Output image 10-case 1

WORMS 3D
http://www.worms3d.com
Output image 10-case 3

The next two images present a magnified part of output image 8 of model case 1 and case 3, respectively:

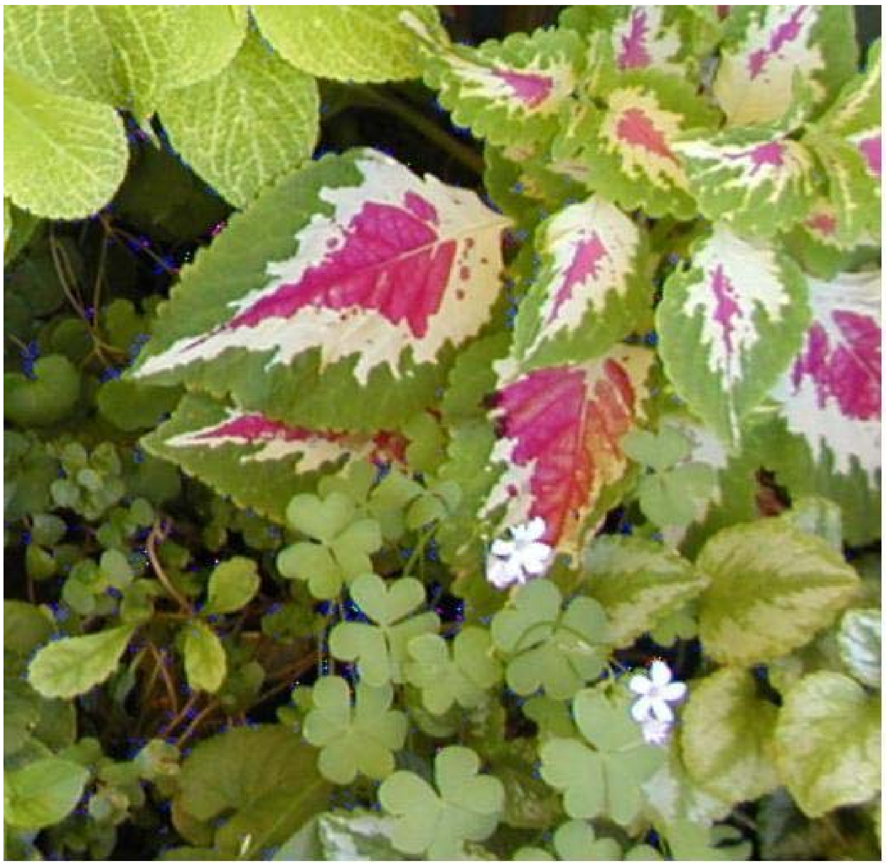


Figure .35-Output image 8 of model case 1

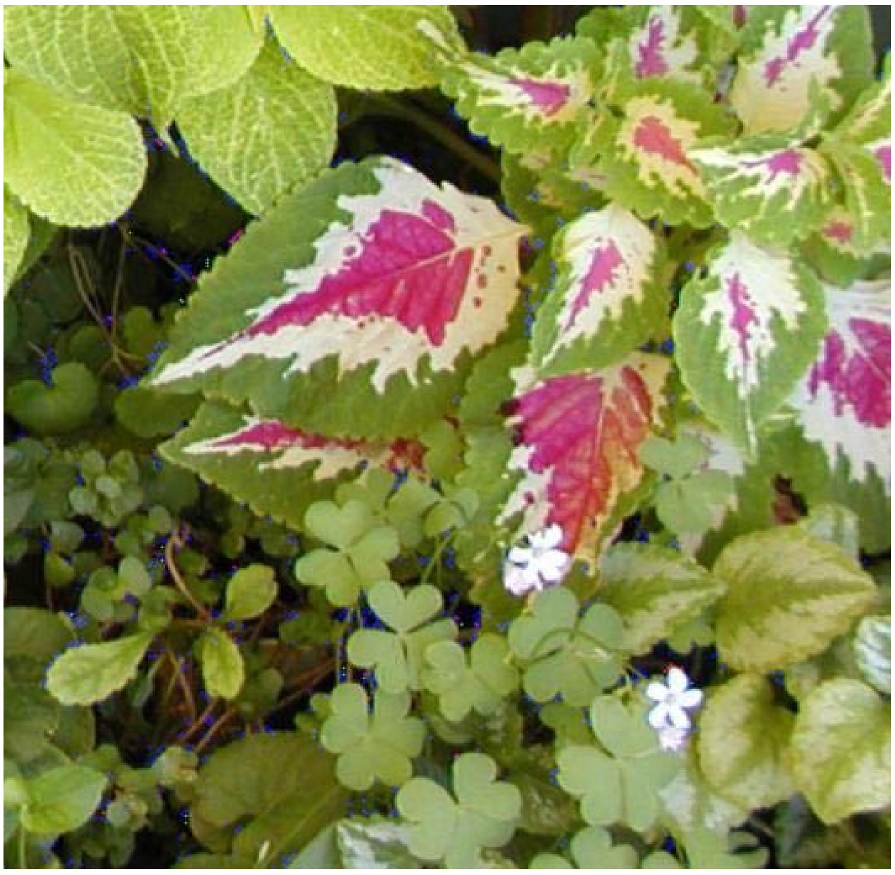


Figure .36-Output image 8 of model case 3

The next two images present a magnified part of output image 9 of model case 1 and case 3, respectively:

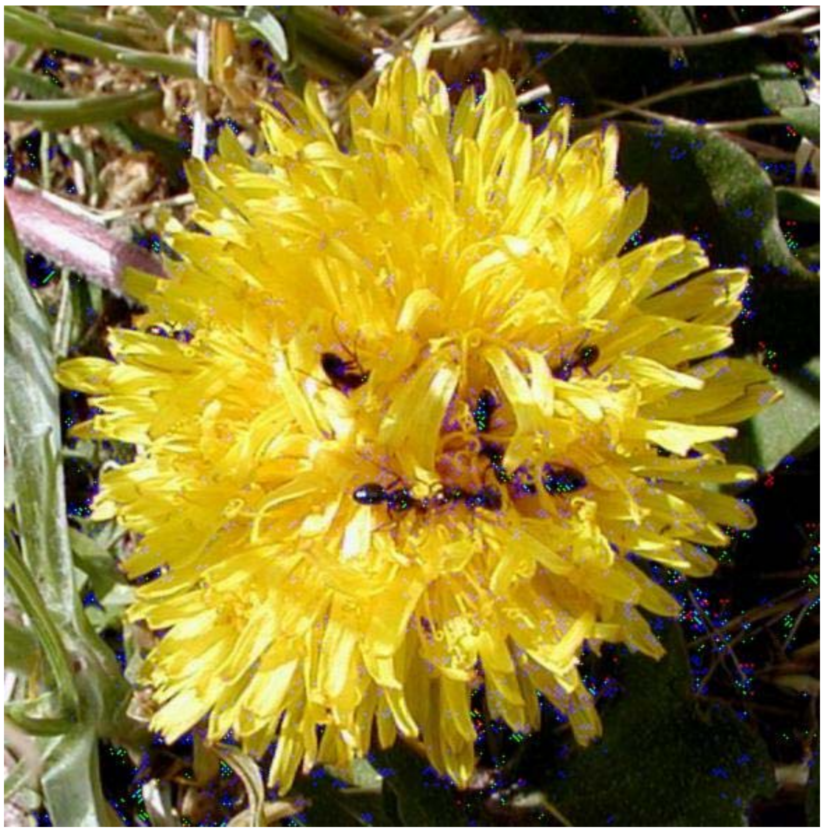


Figure .37- Output image 9 of model case 1

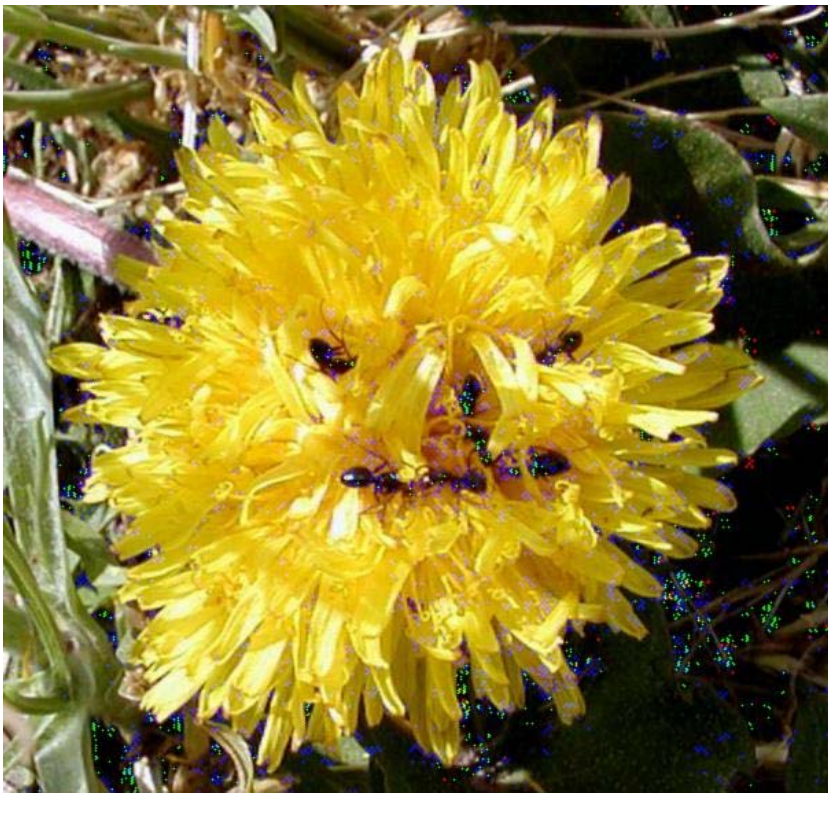


Figure .38- Output image 9 of model case 3

From a visual inspection of the images it is obvious that both model cases give quite satisfactory results. The distortion generated by the error applied is very low in the images taken in the laboratory but in the rest images it is high in model case 1 and even higher in model case 3. For example, this is obvious in output image 9 in the center of the flower and on the right lower corner that the distortion is higher in model case 3 that case 1. This distortion appears as pixels of false colour and it is higher in the last three images because they have more edges and intense colours.

5.3.3 Proposed improvements

In this section an approach towards improving the model case 3 is proposed. The previous sections have revealed that model case 1 works better with this specific error than model case 3. A possible improvement is to find another error matrix of the same size (64x64), which produces a reduced distortion for case 3 in comparison to the distortion of the error given by figure 4.2. The analysis of the previous section is particularly helpful as it is the appropriate guide in order to find this error. Coefficients 63 and 64 of the DCT transformation do not generate minimal error because they are affected by some high frequency elements. According to Figure .33 the coefficients with the least energy are the ones in positions (4, 8) and (5, 8). It is expected that applying the error on these coefficients results in the generation of the minimum distortion in the image. The new error is a 64x64 diagonal matrix with two error coefficients in it, one in line 60 and one in line 61. The error matrix is shown below:

$$I_{e1} \Delta (I_e + \varepsilon)$$

$$with : \varepsilon = \begin{pmatrix} 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & & & & & \\ 0 & & -1 & & & & \\ 0 & & & -1 & & & \\ 0 & & & & 0 & & \\ 0 & & & & & 0 & \\ 0 & & & & & & 0 \end{pmatrix}$$

Figure .39-Improved error matrix

Where I_{e1} is an identity matrix whose size depends on the image it is applied on, for instance if the image is m^2x1 then I_{e1} will be $m^2/64$x$m^2/64$. I_e is the identity matrix of size 64x64.

The error of figure 5.18 is expected to perform better when using model case 3 but not with case 1. The reason that this happens is explained by Figure .33 and is the same reason for which this error performs better in model case 3.

To verify the performance improvement of model case 3, all eight input images are tested again but this time with the error matrix of figure 5.18. This error improvement is only about model case 3 and the new error matrix is enforced only on the green layer, the red and blue layers keep the error matrix of figure 4.2. The results are compared with the ones of table 5.3. The table below presents the new results:

Image No	DCT Type	L_2 Norm
Image0	Case 3 Green	868
Image1	Case 3 Green	970
Image2	Case 3 Green	828
Image3	Case 3 Green	974
Image4	Case 3 Green	1031
Image5	Case 3 Green	851
Image6	Case 3 Green	870
Image7	Case 3 Green	850
Image8	Case 3 Green	22737
Image9	Case 3 Green	36350
Image10	Case 3 Green	16517

Table .4-Distortion with improved error

The above table confirms the quality improvement when using model case 3 with the error of figure 5.18 in comparison to the error of figure 4.2 for all images, with a quality improvement approximately **6.3%**, for the images taken in the laboratory. For the rest the improvement depends on the image itself; the edges, the colours. The reason why this happens lies on Figure .33. By the comparison of table 5.3 and table 5.4 it is observed that the model case 3 with the improved error performs slightly worse than model case 1.

To sum up the improvement achieved in model case 3 by changing the error matrix is limited by the highpass information existing in the image. No matter how much this model is optimized it cannot reach the performance of model case 1.

Another way to improve model case 3 is to change the model itself. By now it is clear that the major difference between the two models is the handling of the green layer. An improvement of case 3 would be to keep the green layer as it is (one layer) but when it comes to applying the DCT, use the implementation of case 1 (two distinct layers). This modeling keeps the main characteristic of case 3 intact and combines it with the main advantage of case 1. The model in block diagram is presented below:

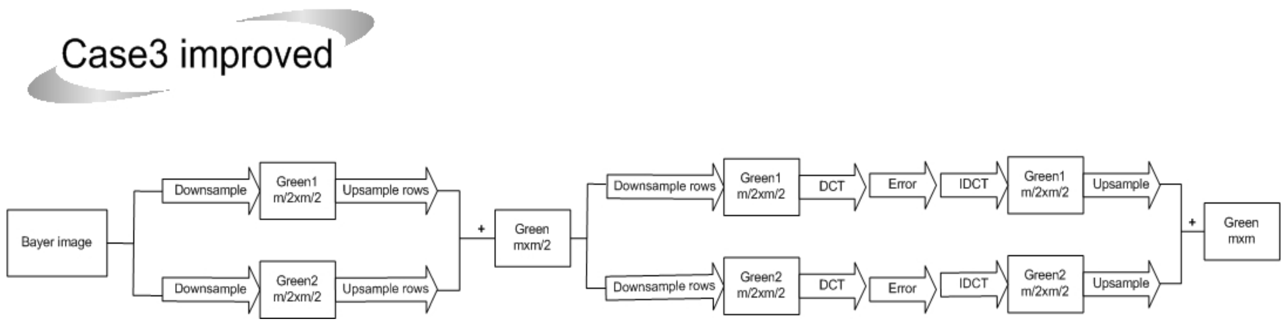


Figure .40-Improved implementation of model case 3

Theoretically this model is expected to perform as well as model case 1, because the part in which the DCT is applied is identical with the one of case 1. On the other hand, there are some more computations to be made by the CPU.

To verify the performance improvement all eleven images are tested again using the new implementation. The results of this improvement are shown in the table below:

Image No	**DCT Type**	**L_2 Norm**
Image0	Case 3	858
Image1	Case 3	960
Image2	Case 3	819
Image3	Case 3	964
Image4	Case 3	1018
Image5	Case 3	834

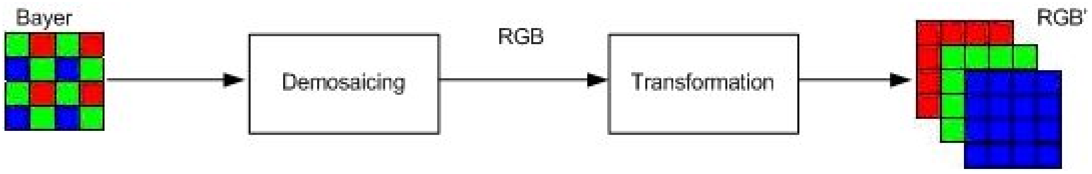
Image6	Case 3	858
Image7	Case 3	842
Image8	Case 3	22683
Image9	Case 3	35663
Image10	Case 3	16542

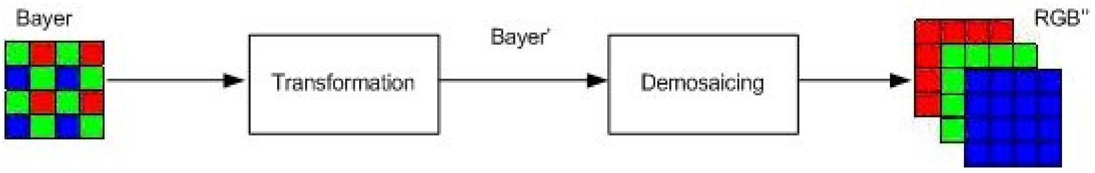
Table .5-Distortion with improved implementation

The table above reveals that the new implementation of model case 3 performs similar to model case 1, which was expected by the theoretical analysis. The improvement in image quality that has been achieved in comparison with the implementation of model case 3 described by equation 4.10 is approximately **7.5%**, for the images taken in the laboratory. For the rest the improvement depends on the image itself; the edges, the colours.

5.4 Conversion schemes' comparison

After having analyzed the implementations of Conversion scheme described by figure 3.2, it is time to compare this scheme with the one of figure 3.1. For convenience, the two Conversion schemes of figure 3.1 and 3.2 are shown again:





It is helpful to compare them in combination with the knowledge of the previous sections. The implementation used in this comparison is the one of model case 1.
In Section 4 that analyses the Conversion schemes, an expression that relates a transformation on the Bayer image with a transformation on the RGB has been found and it is equation 4.4:

$IN * U * B' = A * IN * U$

The attempt of this section is to replace the operations A and B′ of equations 4.2 and 4.3 with the erroneous DCT. This means that in equation 4.2 the DCT transformation is applied on each layer of Bayer image, the error is applied on the result, then the inverse DCT and finally demosaicing is performed. On the other hand, in equation 4.3 the DCT is applied on each layer of the RGB image, after that, the error is applied on the result and finally the inverse DCT. The new equations are presented below with their corresponding block diagrams:

$$\begin{aligned} f &= e_1 \Delta (IN \Delta IN)(U_1 \Delta U_1)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT) r + \\ &+ e_2 \Delta (IN \Delta IN)(U_2 \Delta U_2)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT) b + \\ &+ e_3 \Delta (IN \Delta IN)[(U_2 \Delta U_1)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT) g_1 + \\ &+ (U_1 \Delta U_2)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_e + \varepsilon)](I_{e1} \Delta DCT) g_2] \end{aligned}$$

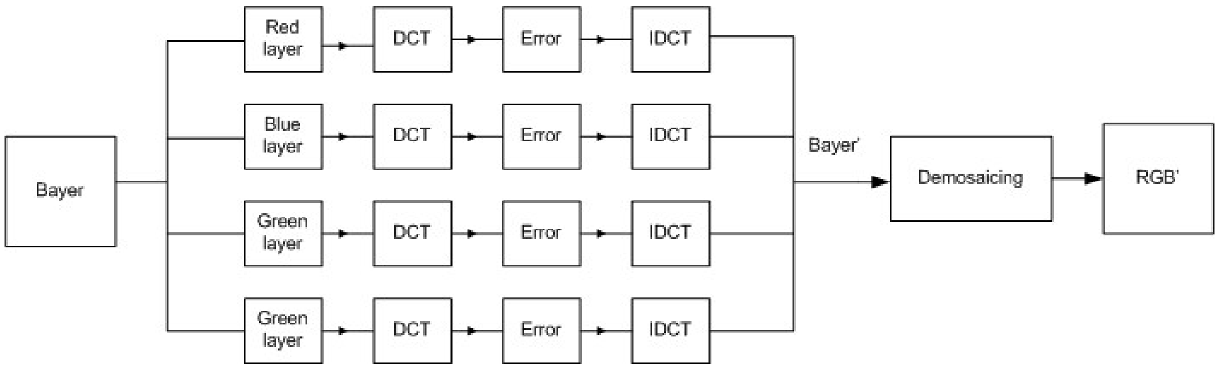


Figure .41-Bayer to RGB Conversion scheme 1

$$\begin{aligned} f' &= (A \Delta A) f = \\ &= e_1 \Delta (I_{e1}' \Delta IDCT)[I_{e1}' \Delta (I_e + \varepsilon)](I_{e1}' \Delta DCT)(IN \Delta IN)(U_1 \Delta U_1) r + \\ &+ e_2 \Delta (I_{e1}' \Delta IDCT)[I_{e1}' \Delta (I_e + \varepsilon)](I_{e1}' \Delta DCT)(IN \Delta IN)(U_2 \Delta U_2) b + \\ &+ e_3 \Delta (I_{e1}' \Delta IDCT)[I_{e1}' \Delta (I_e + \varepsilon)](I_{e1}' \Delta DCT)(IN \Delta IN)[(U_2 \Delta U_1) g_1 + (U_1 \Delta U_2) g_2] \end{aligned}$$

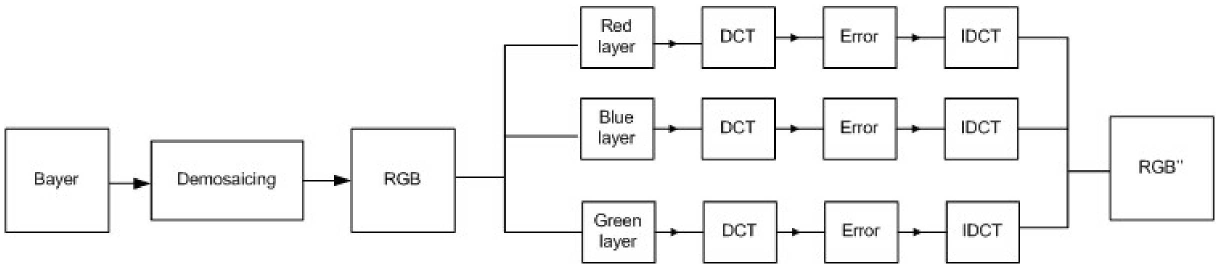


Figure .42-Bayer to RGB Conversion scheme 2

The matrices used in equations 5.9 and 5.10 are the ones that have been described in the previous sections. I_{e1} and I_{e1}' are the identity matrices of size $m^2/(4*64)xm^2/(4*64)$ and $m^2/64xm^2/64$, respectively.

The goal of this analysis is to find how the error of a Bayer image inserts in the corresponding RGB image. Additionally, we compare the distortion generated by this error with the distortion if the error was applied directly on the RGB image. The relation described by equation 4.4 does not hold for the whole image, when B is replaced by DCT, because matrix multiplications cannot be computed due to the different concatenation; block by block rather than the original image concatenation, which is column by column. However, it is correct for an 8x8 subimage, where the interpolation matrix is of size 16x16, the upsampling matrix is 16x8 and the DCT matrix is 8x8, for equation 5.9 and 16x16 for equation 5.10. The difference in the size of the DCT matrices comes from the fact that they are applied on images of different sizes. The equation below is the equation 4.4 for an 8x8 subimage:

$$IN * U * DCT_{(8x8)} = DCT_{(16x16)} * IN * U$$

Where interpolation and upsampling matrices are exactly the same for both parts of the equation, although the DCT is either applied on the 8x8 subimage or on the upsampled and interpolated 16x16 subimage.

In the study that follows there is an attempt to find which scheme performs better when erroneous DCT is applied.

Study: Distortion on Conversion schemes

In this study the goal is to compare the error produced on the conversion scheme, when the erroneous DCT is applied on the Bayer layers and RGB layers. The error matrix used is the error ε described by figure 4.2. The comparison of the error cannot be done by the Frobenius norm, due to the spreading effect of IDCT, as presented in section 5.3. So, matrix analysis is used to provide a visual representation of how data is handled in both situations. This analysis refers to the distortion of the red layer of equation 5.9:

$$(I_N \Delta I_N)(U_1 \Delta U_1)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)r =$$

$$= (I_N \Delta I_N)(U_1 \Delta U_1)(I_{e1} \Delta IDCT)\begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ & & & \\ 0 & & -1 & \\ 0 & \cdots & & -1 \end{bmatrix}\begin{bmatrix} x & \cdots & x & x \\ \vdots & \ddots & & \vdots \\ & & & \\ x & & x & \\ x & \cdots & & x \end{bmatrix}\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{63} \\ r_{64} \\ r_1 \\ \vdots \\ r_{63} \\ r_{64} \end{bmatrix} =$$

$$= (I_N \Delta I_N)(U_1 \Delta U_1)(I_{e1} \Delta IDCT)\begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ & & & \\ 0 & & -1 & \\ 0 & \cdots & & -1 \end{bmatrix}\begin{bmatrix} {}_1 \\ {}_2 \\ \vdots \\ {}_{64} \\ {}_1 \\ {}_2 \\ \vdots \\ {}_{63} \\ {}_{64} \end{bmatrix} =$$

$$= (I_N \Delta I_N)(U_1 \Delta U_1)\begin{bmatrix} r_1' \\ r_2' \\ \vdots \\ r_{63}' \\ r_{64}' \\ r_1' \\ \vdots \\ r_{63}' \\ r_{64}' \end{bmatrix}$$

Where r refers to the pixels of the red layer of the Bayer image, and r′ refers to their distortion due to the application of error ε.

In this case the DCT is applied to the red layer of the Bayer image. The error enforced affects only the last two elements of the DCT, which hold little information of the signal. After applying the inverse DCT the image is upsampled in both lines and columns. As a result the quantity of the error in the upsampled image remains the same but the interpolation expands it in the whole layer. The same happens with the rest summands of equation 5.9.

The following analysis refers to the red layer of equation 5.10:

$$(I_{e1}\Delta IDCT)[I_{e1}\Delta(I_e+\varepsilon)](I_{e1}\Delta DCT)(IN\,\Delta IN)(U_1\Delta U_1)r =$$
$$= (I_{e1}\Delta IDCT)[I_{e1}\Delta(I_e+\varepsilon)](I_{e1}\Delta DCT)R =$$

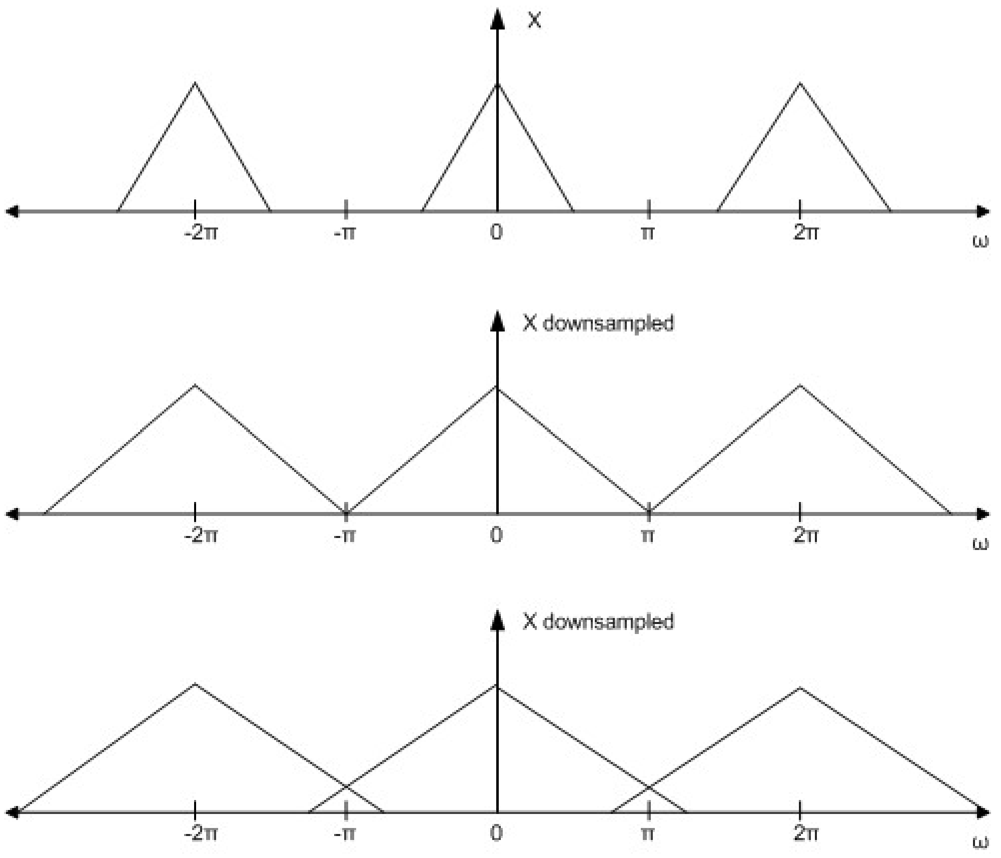
$$= (I_{e1}\Delta IDCT)\begin{bmatrix}0&\cdots&0&0\\ \vdots&\ddots&&\vdots\\ 0&&-1&\\ 0&\cdots&&-1\end{bmatrix}\begin{bmatrix}x&\cdots&x&x\\ \vdots&\ddots&&\vdots\\ x&&x&\\ x&\cdots&&x\end{bmatrix}\begin{bmatrix}R_1\\R_2\\\vdots\\R_{63}\\R_{64}\\R_1\\\vdots\\R_{63}\\R_{64}\end{bmatrix}=$$

$$= (I_{e1}\Delta IDCT)\begin{bmatrix}0&\cdots&0&0\\ \vdots&\ddots&&\vdots\\ 0&&-1&\\ 0&\cdots&&-1\end{bmatrix}\begin{bmatrix}{}_{1}\\{}_{2}\\\vdots\\{}_{63}\\{}_{64}\\{}_{1}\\\vdots\\{}_{63}\\{}_{64}\end{bmatrix}=\begin{bmatrix}R_1'\\R_2'\\\vdots\\R_{63}'\\R_{64}'\\R_1'\\\vdots\\R_{63}'\\R_{64}'\end{bmatrix}$$

Where R refers to the pixels of the Red layer of the RGB image, and R’ refers to their distortion due to the application of error ε.

From the above analysis it is shown that the red layer of the Bayer image is first upsampled in both lines and columns and interpolated in order to create the Red layer of RGB. DCT is applied on the Red layer and the error on the result. The elements affected by the error are the last two of the DCT, as before. Finally, the inverse DCT gives the total distortion of this layer, which in quantity is double to the one of the previous analysis prior to demosaicing.

The difference of these two situations lies on the data on which the DCT is applied. In the conversion scheme described by equation 5.9, the DCT is applied on a downsampled RGB layer. The procedure of downsampling inserts highpass information in the image, which appears at the high frequency coefficients after the conversion to the frequency domain. As already mentioned these coefficients are in the lower and right elements of the DCT, exactly where the error is applied. The distorted image is finally upsampled and interpolated. These two operations do not induce distortion on the image; they just expand it to the corresponding RGB image. On the other hand, the data of an RGB layer do not hold highpass information, resulting in lower distortion of the images. The difference of the distortion can be justified by the following spectral analysis too:



In the above figures it is assumed that there is a signal X, whose frequency domain representation is shown in the first figure. The second figure shows the downsampled version of X. It is obvious that the operation of downsampling spreads the spectrum on [-π, π] region [14]. In other words, the signal is spread to the high frequency regions. Another situation when downsampling is shown in the third figure, where there are aliasing effects and the signal cannot be completely restored. The signal X corresponds to the RGB layers and its downsampled version to the Bayer layers. This analysis confirms that the application of the error on the Bayer image results in higher distortion than on the RGB image.

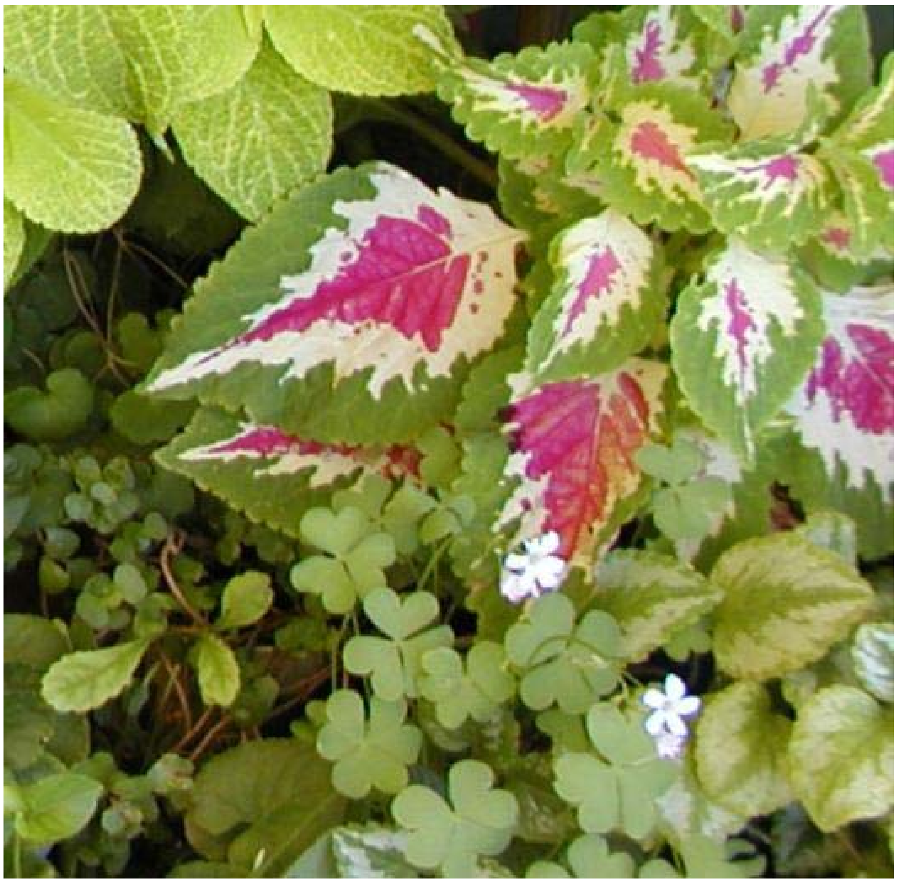
To conclude, the distortion generated in the scheme of figure 3.2 is higher than that of figure 3.1. Apart from the output image quality, the computational cost and the demand of bandwidth is an issue of major importance in this comparison. The operation of demosaicing is the same in both situations, although the transformation (DCT) is performed on different data. In equation 5.10 the computational cost is expected to be higher than the one of equation 5.9, because the data processed in 5.10 are triple in number than the data of 5.9. Additionally, the transformed image requires more storage capacity in the Conversion scheme of figure 5.21 than that of figure 5.20. If the transformed image is transmitted through the network, the bandwidth or transmission time required will be higher for the Conversion scheme of figure 5.21 too.

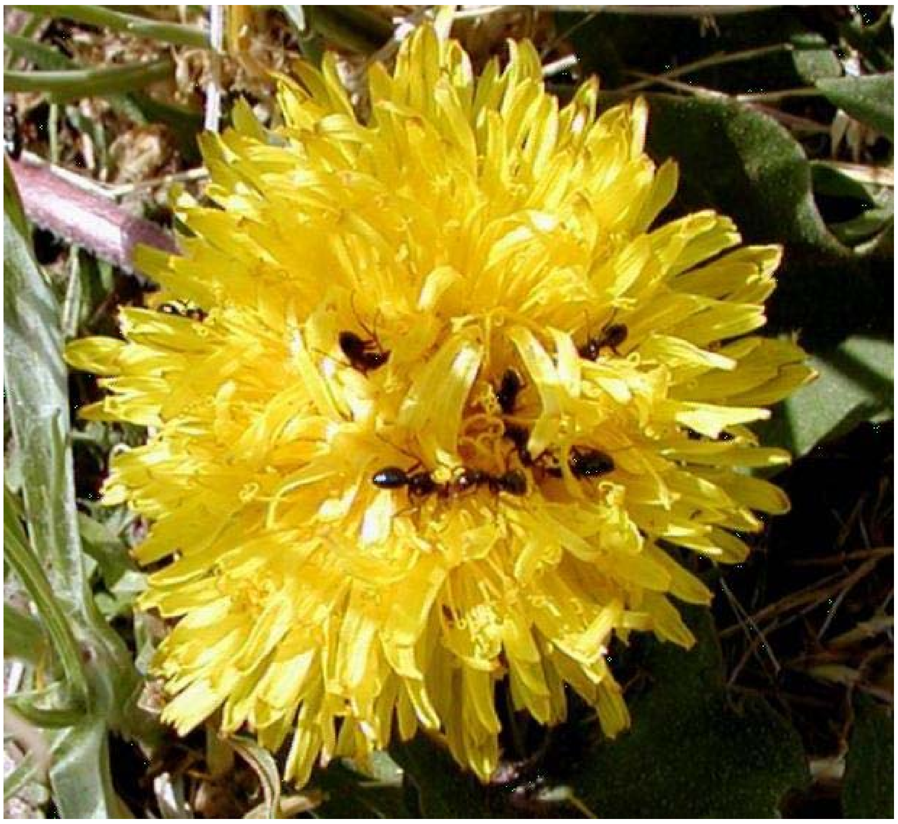
The above theoretical analysis has to be supported by experimental results. An image quality comparison of the Conversion schemes from a Bayer image to RGB is performed in the following. The first Conversion scheme is described by equation 5.9, where the erroneous DCT is applied on the Bayer image. The second one is described by equation 5.10, where the erroneous DCT is applied on the RGB image. The comparison is performed for all input images and the error is computed using the L_2 norm. The results are shown in the table below:

Image No	**L_2 Norm**	
	SCHEME 1	**SCHEME 2**
Image0	858	186
Image1	960	181
Image2	819	266
Image3	964	476
Image4	1018	462
Image5	834	252
Image6	858	174
Image7	842	160
Image8	22683	910
Image9	35663	7727
Image10	16542	6663

Table .6-Distortion comparison of Conversion schemes 1 and 2

Table .6 confirms the results of the mathematical analysis, as the distortion of images produced by the Conversion scheme of equation 5.9 is larger than that of images produced by the Conversion scheme of equation 5.10. Two sample images of Conversion scheme 2 are shown below:





In the above images the improvement in image quality is obvious in comparison with images of Conversion scheme 1, as there are few false coloured pixels.

Finally, a computational cost and bandwidth or transmission time comparison on the Conversion schemes from a Bayer image to RGB is performed. A Pentium4 2.54MHz with 512MB of RAM and Windows XP has been used to make the measurements. The first Conversion scheme is described by equation 5.9, where the erroneous DCT is applied on the Bayer image and the second one is described by equation 5.10, where the erroneous DCT is applied on the RGB image. The comparison is performed for one input image, as their sizes are the same, so does the computational time. The results are shown in the table below:

Computational Time (seconds)	
SCHEME 1	**SCHEME 2**
31.2	94.8

Table .7-Computational time comparison of Conversion schemes 1 and 2

The above table reveals that the computational time for the application of the DCT on a 1600x1200 image is three times lower, when the transformation is applied directly on the Bayer image. The storage capacity required by a transformed image for both Conversion schemes is shown in the table below:

Storage Capacity (MB)	
SCHEME 1	**SCHEME 2**
1.92	5.76

Table .8-Storage capacity comparison of Conversion schemes 1 and 2

The above table shows that the storage capacity required by a transformed image is three times lower, when the transformation is applied directly on the Bayer image. As a result the transmission time through a network is three times lower for the Conversion scheme 1 too. Specifically, in the following table the transmission times of images in both Conversion schemes are presented, for networks of various bandwidths. Supposing that the full bandwidth is available the transmission times are:

Bandwidth of Network	**Transmission Time (seconds)**	
	SCHEME 1	**SCHEME 2**
10Mbps	1.536	4.608
100Mbps	0.1536	0.4608

1Gbps	0.01536	0.04608

Table .9-Transmission time comparison of Conversion schemes 1 and 2

The above table confirms the theoretical approach, that the transmission time of a transformed image is three times lower, when the transformation is applied directly on the Bayer image.

To sum up, the first Conversion scheme performs better in terms of computational complexity and transmission time over a network, although the second Conversion scheme performs better in terms of image quality.

6 CONCLUSION

This thesis implements and examines two Conversion schemes from the Bayer to the RGB image, when linear transformations are applied on them. The difference of the Conversion schemes is the application of the transformation either on the Bayer or on the RGB image. The major transformation of concern is the DCT with the application of a specific error on it. The DCT has been chosen to be studied, because it is a very important component of the compression schemes. These schemes are used in applications which involve transmission of images through network and our intension is to optimize such applications. The Conversion schemes are shown in the figures below:

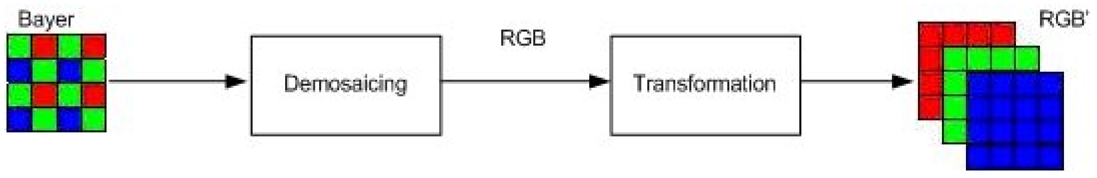


Figure .43-Conversion scheme 1 (transformation on RGB)

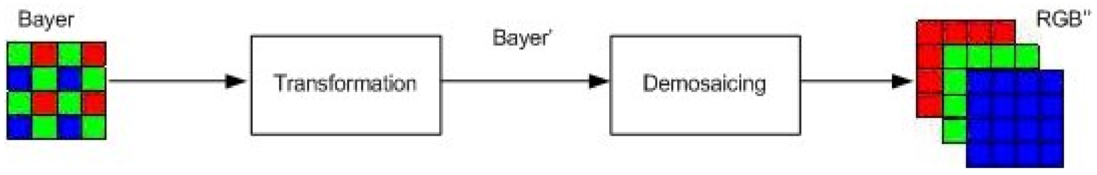


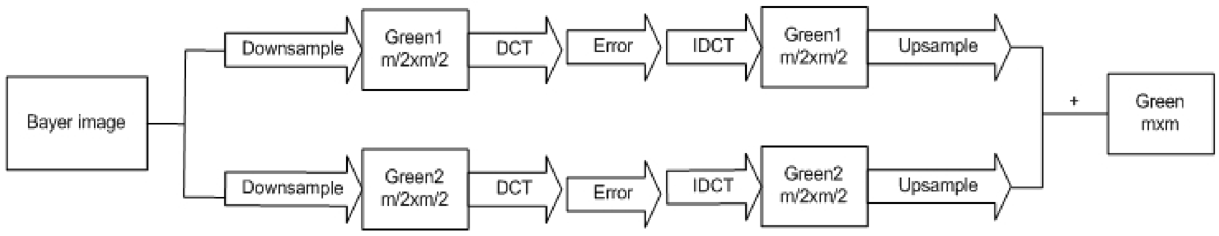
Figure .44-Conversion scheme 2 (transformation on Bayer)

The two Conversion schemes have been compared on the quality performance of the output image. The outcome of the theoretical and experimental analysis is that the first Conversion scheme, described by figure 6.1, performs better than the second Conversion scheme, described by figure 6.2. The difference in performance is a result of the data on which the transformation is applied. In more detail, the data of the colour layers of the Bayer image form actually a downsampled version of the colour layers of the RGB image. The operation of downsampling inserts highpass information in the image, which is reflected in the high frequency elements of its DCT. Because of the fact that the specific error affects these high frequency elements, the distortion generated in the output image is high. In contrary, the highpass information of the RGB image is rather lower and, as a result, the distortion generated in the output image is also low.

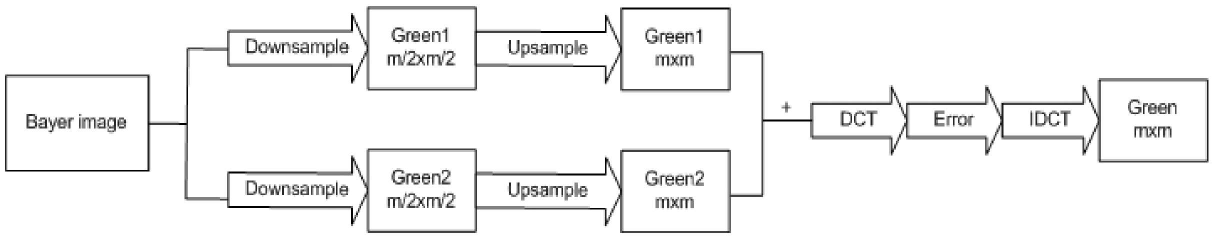
To conclude, the first Conversion scheme performs better in terms of image quality, however it has an important drawback, which has to be mentioned even though it is not the goal of this thesis. When the transformation is applied on the RGB image and the result is stored or transmitted through a network, then the storage capacity and transmission time of this image is rather high. Furthermore, the computational cost of the transformation on the RGB image is higher than the cost of the transformation on the Bayer. The inverse holds for the second Conversion scheme; the quality of the output image is low but computational cost and the transmission time of the transformed image are quite low too.

Additionally, this thesis examines and compares, in terms of image quality, three distinct implementations of the second Conversion scheme, in which the transformation is applied directly on the Bayer image. We focus on the Conversion scheme of figure 6.2 in order to take advantage of its benefits, which were mentioned earlier, and try to optimize its drawbacks. The distinct implementations of the second Conversion scheme differ on the way that the green layer is handled. The implementations are described by model cases 1, 2 and 3 whose block diagrams of the green layer are respectively shown below:

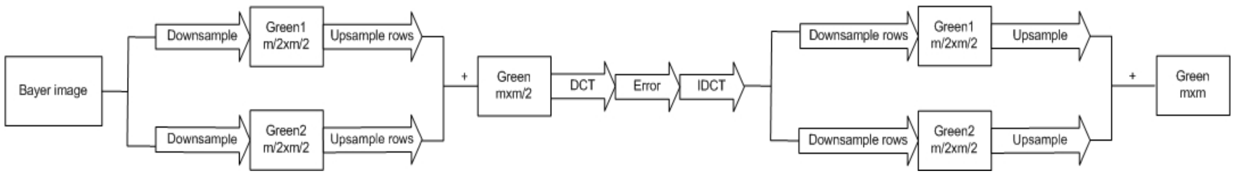
Case1



Case2



Case 3



In the implementation of model case 1 the green pixels of the odd lines of the Bayer image are separated from the ones of the even lines and the transformation is applied on each one of them. In model case 2 the green layer is of the same size of the Bayer image with zeros in the red and blue positions. Finally, in model case 3 the green pixels of the even columns are shifted one column left and the zero columns are discarded. The implementation of model case 2 does not perform well with compression algorithms, because of the high frequency content. The zeros insert highpass information which does not allow high compression rates or high quality of compressed images. As a result this case is not analyzed. The other two implementations are compared in terms of output image quality, when the transformation applied is the erroneous DCT.

The implementations of model case 1 and 3 have been compared on the quality performance of the output image. The outcome of the theoretical and experimental analysis is that model case 1 performs better than model case 3. The difference in performance is a result of the data on which the transformation is applied. In more detail, the shifting to the left of the even columns of the green layer in model case 3 results in the insertion of highpass information. As already mentioned, highpass information creates high levels of distortion in the output image.

The application of the improvements of model case 3 has resulted in its performance to converge to the performance of model case 1. This has been achieved in two ways. The first one is the utilization of the knowledge that the application of an error can be done on specific elements of the DCT, which generate minimum distortion. The second one is the implementation of an improved model, which combines the advantages of model case 1 with the main characteristics of model case 3.

Future work

In order to improve the quality of the output image as long as the computational cost and the transmission time, the following aspects could be further investigated. JPEG, as presented in Section 2, consists of transformations that are both linear, such as DCT and Zig-Zag ordering, and non-linear, such as Quantization. Quantization, however, is a transformation that cannot be expressed using algebraic operations, so another type of modeling is required.
Additionally, the steps of JPEG after DCT, such as Zig-Zag ordering and Entropy encoding could be further analyzed. The Conversion model could be used to explore the different situations that arise and their properties, in order to improve the quality of the output image.
The JPEG committee has released its new image coding standard, JPEG 2000, which has adopted an implementation of an entirely new way of compressing images. Rather than incrementally improving the original standard using the discrete cosine transform (DCT), the new coding standard is based on the wavelet transform. The wavelet transform is based on the time-frequency domain understanding of signal analysis, which is advantageous compared to current transforms where only one of the domains is available at any given time. The application JPEG 2000 and its wavelet transformation could also evolve the Conversion model.
Finally, a new modeling could result from the combination of the Conversion model of this thesis with different types of Colour Filter Arrays. The new modeling, although based on the same concept, could improve the quality of the output image by taking advantage of the characteristics of the new CFA.

7 APPENDIX A-Mathematical proof

1. the proof of equation $B' = U^T * B * U$ is:

   $(B \Delta B)(U \Delta U) = (U \Delta U)(B' \Delta B')$ Ÿ

   Ÿ $B*U \Delta B*U = U*B' \Delta U*B'$ Ÿ

   ή $B*U = U*B'$ Ÿ

   Ÿ $B' = U^T * B * U$

   the proof of equation 4.4 is:

   $(IN \Delta IN)(U \Delta U)(B' \Delta B') = (A \Delta A)(U \Delta U)(IN \Delta IN)$ Ÿ

   Ÿ $IN*U*B' \Delta IN*U*B' = A*IN*U \Delta A*IN*U$ Ÿ

   ή $IN*U*B' = A*IN*U$

2. the proof of equation 4.7 is:

$$(U \Delta U)(D \Delta D)(A \Delta A) = (B \Delta B)(U \Delta U)(D \Delta D) \text{ Ÿ}$$
$$\text{Ÿ } (U * D \Delta U * D)(A \Delta A) = (B \Delta B)(U * D \Delta U * D) \text{ Ÿ}$$
$$\text{Ÿ } U * D * A \Delta U * D * A = B * U * D \Delta B * U * D) \text{ ή}$$
$$\text{ή } U * D * A = B * U * D$$
$$Generally : D = I \Delta e_k$$
$$and : U = D^T = I \Delta e_k^T$$
$$So : D * A * D^T = U^T * B * U$$

3. The following equation is the one that we described for the green layer of model case 1:

$$(U_2 \Delta U_1)(D_2 \Delta D_1) + (U_1 \Delta U_2)(D_1 \Delta D_2) =$$
$$= (I \Delta e_1^T \Delta I \Delta e_0^T)(I \Delta e_1 \Delta I \Delta e_0) + (I \Delta e_0^T \Delta I \Delta e_1^T)(I \Delta e_0 \Delta I \Delta e_1) =$$
$$= I \Delta e_1^T e_1 \Delta I \Delta e_0^T e_0 + I \Delta e_0^T e_0 \Delta I \Delta e_1^T e_1 =$$
$$= I \Delta (e_1^T e_1 \Delta I \Delta e_0^T e_0 + e_0^T e_0 \Delta I \Delta e_1^T e_1) \text{ʹΗ}I$$

$$or : (I_1 \Delta I_0)^T (I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I_0 \Delta I_1)$$
$$with : I_0 = I \Delta e_0$$
$$and : I_1 = I \Delta e_1$$

And the next equation refers to the green layer of model case 3:

$$[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)][(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)] =$$
$$= (U_2 \Delta U_1)(I \Delta D_1)(I \Delta U_1)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)(I \Delta U_2)(D_1 \Delta D_2) =$$
$$= (I \Delta e_1^T \Delta I \Delta e_0^T)(I \Delta I \Delta e_0)(I \Delta I \Delta e_0^T)(I \Delta e_1 \Delta I \Delta e_0) +$$
$$+ (I \Delta e_0^T \Delta I \Delta e_1^T)(I \Delta I \Delta e_1)(I \Delta I \Delta e_1^T)(I \Delta e_0 \Delta I \Delta e_1) =$$
$$= I \Delta (e_1^T \Delta I \Delta e_0^T)(I \Delta e_0 e_0^T)(e_1 \Delta I \Delta e_0) + I \Delta (e_0^T \Delta I \Delta e_1^T)(I \Delta e_1 e_1^T)(e_0 \Delta I \Delta e_1) =$$
$$= I \Delta [(e_1^T \Delta I \Delta e_0^T)(I \Delta e_0 e_0^T)(e_1 \Delta I \Delta e_0) + (e_0^T \Delta I \Delta e_1^T)(I \Delta e_1 e_1^T)(e_0 \Delta I \Delta e_1)] \text{ʹŸŸŸŸŸŸ}\rightarrow^{e_0 e_0^T = e_1 e_1^T = [1]}$$
$$= I \Delta [(e_1^T \Delta I \Delta e_0^T)(e_1 \Delta I \Delta e_0) + (e_0^T \Delta I \Delta e_1^T)(e_0 \Delta I \Delta e_1)] =$$
$$= I \Delta (e_1^T e_1 \Delta I \Delta e_0^T e_0 + e_0^T e_0 \Delta I \Delta e_1^T e_1) \text{ʹΗ}I$$

4. The following equation is the green layer of model case 1 including the error. The right hand side of this equation has two parts. The first one is the ideal value and the second one is the erroneous value. The equation after that is the error alone in its general formation:

$$
\begin{aligned}
&(U_2 \Delta U_1)[I_{e1} \Delta (I_e + \varepsilon)](D_2 \Delta D_1) + (U_1 \Delta U_2)[I_{e1} \Delta (I_e + \varepsilon)](D_1 \Delta D_2) = \\
&= (U_2 \Delta U_1)(I_{e1} \Delta I_e)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I_{e1} \Delta I_e)(D_1 \Delta D_2) + \\
&+(U_2 \Delta U_1)(I_{e1} \Delta \varepsilon)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I_{e1} \Delta \varepsilon)(D_1 \Delta D_2)
\end{aligned}
$$

$$
\begin{aligned}
&error = (U_2 \Delta U_1)(I_{e1} \Delta \varepsilon)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I_{e1} \Delta \varepsilon)(D_1 \Delta D_2) = \\
&= (I \Delta e_1^T \Delta I \Delta e_0^T)(I_{e1} \Delta \varepsilon)(I \Delta e_1 \Delta I \Delta e_0) + (I \Delta e_0^T \Delta I \Delta e_1^T)(I_{e1} \Delta \varepsilon)(I \Delta e_0 \Delta I \Delta e_1) \\
&= (I_1 \Delta I_0)^T (I_{e1} \Delta \varepsilon)(I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I_{e1} \Delta \varepsilon)(I_0 \Delta I_1) \\
&with : I_0 = I \Delta e_0 \\
&and : I_1 = I \Delta e_1
\end{aligned}
$$

For a mxm Bayer image the size of I_0 and I_1 is m/2xm, I is m/2xm/2 and the size of the whole error matrix is $m^2/4xm^2/4$.

Similarly with the above equations, the first one that follows is the expression of the green layer of model case 3, including the error. The right hand side of this equation has two parts. The first one is the ideal value and the second one is the erroneous value. The equation after that is the error alone in its general formation:

$$
\begin{aligned}
&[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)][I_{e3} \Delta (I_e + \varepsilon)][(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)] = \\
&= [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta I_e)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)] + \\
&+[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta \varepsilon)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)]
\end{aligned}
$$

$$
\begin{aligned}
&error = [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta \varepsilon)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)] \\
&= (U_2 \Delta U_1)(I \Delta D_1)(I_{e3} \Delta \varepsilon)(I \Delta U_1)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)(I_{e3} \Delta \varepsilon)(I \Delta U_2)(D_1 \Delta D_2) = \\
&= (I \Delta e_1^T \Delta I \Delta e_0^T)(I \Delta I \Delta e_0)(I_{e3} \Delta \varepsilon)(I \Delta I \Delta e_0^T)(I \Delta e_1 \Delta I \Delta e_0) + \\
&+(I \Delta e_0^T \Delta I \Delta e_1^T)(I \Delta I \Delta e_1)(I_{e3} \Delta \varepsilon)(I \Delta I \Delta e_1^T)(I \Delta e_0 \Delta I \Delta e_1) = \\
&= (I_1^T \Delta I_0^T)(I \Delta I_0)(I_{e3} \Delta \varepsilon)(I \Delta I_0^T)(I_1 \Delta I_0) + (I_0^T \Delta I_1^T)(I \Delta I_1)(I_{e3} \Delta \varepsilon)(I \Delta I_1^T)(I_0 \Delta I_1) = \\
&= (I_1 \Delta I_0)^T (I \Delta I_0)(I_{e3} \Delta \varepsilon)(I \Delta I_0)^T (I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I \Delta I_1)(I_{e3} \Delta \varepsilon)(I \Delta I_1)^T (I_0 \Delta I_1) \\
&or : (I_1 \Delta I_0)^T \varepsilon_1 (I_1 \Delta I_0) + (I_0 \Delta I_1)^T \varepsilon_2 (I_0 \Delta I_1) \\
&with : I_0 = I \Delta e_0 \\
&and : I_1 = I \Delta e_1 \\
&and : \varepsilon_1 = (I \Delta I_0)(I_{e3} \Delta \varepsilon)(I \Delta I_0)^T \\
&and : \varepsilon_2 = (I \Delta I_1)(I_{e3} \Delta \varepsilon)(I \Delta I_1)^T
\end{aligned}
$$

For a Bayer image of size mxm the size of I_0 and I_1 is m/2xm, I is m/2xm/2 and the size of the whole error matrix is $m^2/2xm^2/2$. We observe that the size of the error matrix is different in both dimensions in compare to the one of model case 1.

5. The first expression is about the model in case 1 and the second is about the model in case 3:

$\|\Delta b_1\|_F = \|\Delta A_1 x\|_F \le \|\Delta A_1\|_F \|x\|_F$ ή $\frac{\|\Delta b_1\|_F}{\|x\|_F} \le \|\Delta A_1\|_F =$

$$= \left\|(I_1 \Delta I_0)^T (I_{e1} \Delta \varepsilon)(I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I_{e1} \Delta \varepsilon)(I_0 \Delta I_1)\right\|_F \le$$

$$\le \left\|(I_1 \Delta I_0)^T (I_{e1} \Delta \varepsilon)(I_1 \Delta I_0)\right\|_F + \left\|(I_0 \Delta I_1)^T (I_{e1} \Delta \varepsilon)(I_0 \Delta I_1)\right\|_F \le$$

$$\le \left\|(I_1 \Delta I_0)^T\right\|_F \left\|(I_{e1} \Delta \varepsilon)\right\|_F \left\|(I_1 \Delta I_0)\right\|_F + \left\|(I_0 \Delta I_1)^T\right\|_F \left\|(I_{e1} \Delta \varepsilon)\right\|_F \left\|(I_0 \Delta I_1)\right\|_F =$$

$$= \|I_1 \Delta I_0\|_F \|I_{e1} \Delta \varepsilon\|_F \|I_1 \Delta I_0\|_F + \|I_0 \Delta I_1\|_F \|I_{e1} \Delta \varepsilon\|_F \|I_0 \Delta I_1\|_F =$$

$$= \sqrt{(m^2/4)(k+l)(m^2/4*64)(m^2/4)} + \sqrt{(m^2/4)(k+l)(m^2/4*64)(m^2/4)} =$$

$$= \sqrt{k+l} * m^3/32$$

$\|\Delta b_2\|_F = \|\Delta A_2 x\|_F \le \|\Delta A_2\|_F \|x\|_F$ ή $\frac{\|\Delta b_2\|_F}{\|x\|_F} \le \|\Delta A_2\|_F =$

$$= \left\|[(I_1 \Delta I_0)^T (I \Delta I_0) + (I_0 \Delta I_1)^T (I \Delta I_1)](I_{e3} \Delta \varepsilon)[(I \Delta I_0)^T (I_1 \Delta I_0) + (I \Delta I_1)^T (I_0 \Delta I_1)\right\|_F \le$$

$$\le \left\|(I_1 \Delta I_0)^T (I \Delta I_0) + (I_0 \Delta I_1)^T (I \Delta I_1)\right\|_F \|I_{e3} \Delta \varepsilon\|_F \left\|(I \Delta I_0)^T (I_1 \Delta I_0) + (I \Delta I_1)^T (I_0 \Delta I_1)\right\|_F \le$$

$$\le [\|I_1 \Delta I_0\|_F \|I \Delta I_0\|_F + \|I_0 \Delta I_1\|_F \|I \Delta I_1\|_F]\|I_{e3} \Delta \varepsilon\|_F [\|I \Delta I_0\|_F \|I_1 \Delta I_0\|_F + \|I \Delta I_1\|_F \|I_0 \Delta I_1\|_F] =$$

$$= [\sqrt{(m^2/4)(m^2/4)} + \sqrt{(m^2/4)(m^2/4)}]\sqrt{(k+l)(m^2/2*64)}[\sqrt{(m^2/4)(m^2/4)} + \sqrt{(m^2/4)(m^2/4)}] =$$

$$= \sqrt{k+l} * m^5/45$$

We compare the results:

$\frac{\|\Delta b_1\|_F}{\|x\|_F} \le \frac{\|\Delta b_2\|_F}{\|x\|_F}$ ή $\sqrt{k+l} * m^3/32 \le \sqrt{k+l} * m^5/90$ ή $\frac{m^5}{m^3} \ge \frac{90}{32}$ ή

ή $m^2 \ge 3 \to true : m \ge 2$

6. The first expression is about the model in case 1 and the second is about the model in case 3:

$\|\Delta b_1\|_F = \|\Delta A_1 x\|_F \le \|\Delta A_1\|_F \|x\|_F$ ή $\frac{\|\Delta b_1\|_F}{\|x\|_F} \le \|\Delta A_1\|_F =$

$$= \left\|(I_1 \Delta I_0)^T (I_{e1} \Delta \varepsilon)(I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I_{e1} \Delta \varepsilon)(I_0 \Delta I_1)\right\|_F \le$$

$$\le \left\|(I_1 \Delta I_0)^T (I_{e1} \Delta \varepsilon)(I_1 \Delta I_0)\right\|_F + \left\|(I_0 \Delta I_1)^T (I_{e1} \Delta \varepsilon)(I_0 \Delta I_1)\right\|_F =$$

$$= \sqrt{(k+l)(m^2/4*64)} + \sqrt{(k+l)(m^2/4*64)} = \sqrt{k+l} * m/8$$

$\|\Delta b_2\|_F = \|\Delta A_2 x\|_F \le \|\Delta A_2\|_F \|x\|_F$ ή $\frac{\|\Delta b_2\|_F}{\|x\|_F} \le \|\Delta A_2\|_F =$

$$= \left\|(I_1 \Delta I_0)^T (I \Delta I_0)(I_{e3} \Delta \varepsilon)(I \Delta I_0)^T (I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I \Delta I_1)(I_{e3} \Delta \varepsilon)(I \Delta I_1)^T (I_0 \Delta I_1)\right\|_F \le$$

$$\le \left\|(I_1 \Delta I_0)^T (I \Delta I_0)(I_{e3} \Delta \varepsilon)(I \Delta I_0)^T (I_1 \Delta I_0)\right\|_F + \left\|(I_0 \Delta I_1)^T (I \Delta I_1)(I_{e3} \Delta \varepsilon)(I \Delta I_1)^T (I_0 \Delta I_1)\right\|_F =$$

$$= \sqrt{k(m^2/2*64)} + \sqrt{l(m^2/2*64)} = (\sqrt{k} + \sqrt{l})m/11.2$$

We compare the results:

$$\frac{\|\Delta b_1\|_F}{\|x\|_F} \le \frac{\|\Delta b_2\|_F}{\|x\|_F} \text{ ή } \sqrt{k+l} * m/8 \le (\sqrt{k}+\sqrt{l})m/11.2 \text{ ή } 1.4\sqrt{k+l} \le \sqrt{k}+\sqrt{l} \text{ ή }$$

$$\text{ή } 2(k+l) \le k+l+2\sqrt{kl} \text{ ή } k+l \le 2\sqrt{kl} \text{ ή } (\sqrt{k}+\sqrt{l})^2 \le 0 \to untrue$$

7. The first expression that follows represents the error in model case 1 and the second expression the error in model case 3:

$$\|\Delta b_1\|_F = \|\Delta A_1 x\|_F \le \|\Delta A_1\|_F \|x\|_F \text{ ή } \frac{\|\Delta b_1\|_F}{\|x\|_F} \le \|\Delta A_1\|_F =$$

$$= \left\|(I_1 \,\Delta\, I_0)^T (I_{e1} \,\Delta\, \varepsilon)(I_1 \,\Delta\, I_0) + (I_0 \,\Delta\, I_1)^T (I_{e1} \,\Delta\, \varepsilon)(I_0 \,\Delta\, I_1)\right\|_F \le$$

$$\le \left\|(I_1 \,\Delta\, I_0)^T (I_{e1} \,\Delta\, \varepsilon)(I_1 \,\Delta\, I_0)\right\|_F + \left\|(I_0 \,\Delta\, I_1)^T (I_{e1} \,\Delta\, \varepsilon)(I_0 \,\Delta\, I_1)\right\|_F =$$

$$= \sqrt{(\frac{1}{64}+\frac{2}{64})(\frac{m^2}{4*64})} + \sqrt{(\frac{1}{64}+\frac{2}{64})(\frac{m^2}{4*64})} = \frac{\sqrt{3}m}{64} \simeq 0.02m$$

$$\|\Delta b_2\|_F = \|\Delta A_2 x\|_F \le \|\Delta A_2\|_F \|x\|_F \text{ ή } \frac{\|\Delta b_2\|_F}{\|x\|_F} \le \|\Delta A_2\|_F =$$

$$= \left\|(I_1 \,\Delta\, I_0)^T (I \,\Delta\, I_0)(I_{e3} \,\Delta\, \varepsilon)(I \,\Delta\, I_0)^T (I_1 \,\Delta\, I_0) + (I_0 \,\Delta\, I_1)^T (I \,\Delta\, I_1)(I_{e3} \,\Delta\, \varepsilon)(I \,\Delta\, I_1)^T (I_0 \,\Delta\, I_1)\right\|_F \le$$

$$\le \left\|(I_1 \,\Delta\, I_0)^T (I \,\Delta\, I_0)(I_{e3} \,\Delta\, \varepsilon)(I \,\Delta\, I_0)^T (I_1 \,\Delta\, I_0)\right\|_F + \left\|(I_0 \,\Delta\, I_1)^T (I \,\Delta\, I_1)(I_{e3} \,\Delta\, \varepsilon)(I \,\Delta\, I_1)^T (I_0 \,\Delta\, I_1)\right\|_F =$$

$$= \sqrt{(\frac{1}{64}+\frac{32}{64})\frac{m^2}{4*64}} + \sqrt{(\frac{2}{64}+\frac{33}{64})\frac{m^2}{4*64}} = \frac{(\sqrt{33}+\sqrt{35})m}{128} \simeq 0.09m$$

8. The first expression that follows represents the error in model case 1 and the second expression the error in model case 3:

$$\|\Delta b_1\|_F = \|\Delta A_1 x\|_F \le \|\Delta A_1\|_F \|x\|_F \text{ ή } \frac{\|\Delta b_1\|_F}{\|x\|_F} \le \|\Delta A_1\|_F =$$

$$= \left\|(I_1 \,\Delta\, I_0)^T (I_{e1} \,\Delta\, \varepsilon)(I_1 \,\Delta\, I_0) + (I_0 \,\Delta\, I_1)^T (I_{e1} \,\Delta\, \varepsilon)(I_0 \,\Delta\, I_1)\right\|_F =$$

$$= \sqrt{(\frac{1}{64}+\frac{2}{64})(\frac{m^2}{4*64}) + (\frac{1}{64}+\frac{2}{64})(\frac{m^2}{4*64})} = \frac{\sqrt{6}m}{128} \simeq 0.0191m$$

$$\|\Delta b_2\|_F = \|\Delta A_2 x\|_F \le \|\Delta A_2\|_F \|x\|_F \text{ ή } \frac{\|\Delta b_2\|_F}{\|x\|_F} \le \|\Delta A_2\|_F =$$
$$= \|[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e3} \Delta \varepsilon)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)]\|_F =$$
$$= \|(U_2 \Delta U_1)(I \Delta D_1)(I_{e3} \Delta \varepsilon)(I \Delta U_1)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)(I_{e3} \Delta \varepsilon)(I \Delta U_2)(D_1 \Delta D_2)\|_F =$$
$$= \|(I_1 \Delta I_0)^T (I \Delta I_0)(I_{e3} \Delta \varepsilon)(I \Delta I_0)^T (I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I \Delta I_1)(I_{e3} \Delta \varepsilon)(I \Delta I_1)^T (I_0 \Delta I_1)\|_F =$$
$$= \sqrt{(\frac{1}{64} + \frac{32}{64})\frac{m^2}{4*64} + (\frac{2}{64} + \frac{33}{64})\frac{m^2}{4*64}} = \frac{\sqrt{68}m}{128} \simeq 0.0884m$$

9. The right hand side of this equation has two parts. The first one is the ideal value and the second one is the erroneous value. The equation after that is the error alone in its general formation:

$$(U_2 \Delta U_1)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_{e2} + \varepsilon)](I_{e1} \Delta DCT)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I_{e1} \Delta IDCT)[I_{e1} \Delta (I_{e2} + \varepsilon)](I_{e1} \Delta DCT)(D_1 \Delta D_2) =$$
$$= (U_2 \Delta U_1)(I_{e1} \Delta IDCT)(I_{e1} \Delta I_{e2})(I_{e1} \Delta DCT)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I_{e1} \Delta IDCT)(I_{e1} \Delta I_{e2})(I_{e1} \Delta DCT)(D_1 \Delta D_2) +$$
$$+ (U_2 \Delta U_1)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(D_1 \Delta D_2)$$

$$error = (U_2 \Delta U_1)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(D_2 \Delta D_1) + (U_1 \Delta U_2)(I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(D_1 \Delta D_2) =$$
$$= (I_1 \Delta I_0)^T (I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(I_0 \Delta I_1) =$$
$$= (I_1 \Delta I_0)^T (I_{e1} \Delta IDCT * \varepsilon * DCT)(I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I_{e1} \Delta IDCT * \varepsilon * DCT)(I_0 \Delta I_1)$$
$$with : I_0 = I \Delta e_0$$
$$and : I_1 = I \Delta e_1$$

For a mxm Bayer image the size of I_0 and I_1 is m/2xm, I is m/2xm/2 and the size of the whole error matrix, the whole DCT matrix and the whole IDCT matrix is $m^2/4xm^2/4$.

Similarly with the above equations, the first one that follows is the expression of the green layer of model case 3, including the error. The right hand side of this equation has two parts. The first one is the ideal value and the second one is the erroneous value. The equation after that is the error alone in its general formation:

$$[(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e1}' \Delta IDCT)[I_{e1}' \Delta (I_{e2} + \varepsilon)](I_{e1}' \Delta DCT)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)] =$$
$$= [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e1}' \Delta IDCT)(I_{e1}' \Delta I_{e2})(I_{e1}' \Delta DCT)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)] +$$
$$+ [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)]$$

$$error = [(U_2 \Delta U_1)(I \Delta D_1) + (U_1 \Delta U_2)(I \Delta D_2)](I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)[(I \Delta U_2)(D_1 \Delta D_2) + (I \Delta U_1)(D_2 \Delta D_1)] =$$
$$= (U_2 \Delta U_1)(I \Delta D_1)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta U_2)(D_1 \Delta D_2) +$$
$$+ (U_2 \Delta U_1)(I \Delta D_1)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta U_1)(D_2 \Delta D_1) +$$
$$+ (U_1 \Delta U_2)(I \Delta D_2)](I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta U_2)(D_1 \Delta D_2) +$$
$$+ (U_1 \Delta U_2)(I \Delta D_2)](I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta U_1)(D_2 \Delta D_1)] =$$
$$with : I_0 = I \Delta e_0$$
$$and : I_1 = I \Delta e_1$$

For a Bayer image of size mxm the size of I_0 and I_1 is m/2xm, I is m/2xm/2 and the size of the whole error matrix, the whole DCT matrix and the whole IDCT matrix is $m^2/2xm^2/2$. We observe that the size of the error, DCT and IDCT matrices are different in both dimensions in compare to the one of model case 1.

10. The first expression is about the model in case 1 and the second is about the model in case 3:

$$\|\Delta b_1\|_F = \|\Delta A_1 x\|_F \leq \|\Delta A_1\|_F \|x\|_F \text{ ή } \frac{\|\Delta b_1\|_F}{\|x\|_F} \leq \|\Delta A_1\|_F =$$
$$= \left\|(I_1 \Delta I_0)^T (I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(I_1 \Delta I_0) + (I_0 \Delta I_1)^T (I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(I_0 \Delta I_1)\right\|_F \leq$$
$$\leq \left\|(I_1 \Delta I_0)^T (I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(I_1 \Delta I_0)\right\|_F + \left\|(I_0 \Delta I_1)^T (I_{e1} \Delta IDCT)(I_{e1} \Delta \varepsilon)(I_{e1} \Delta DCT)(I_0 \Delta I_1)\right\|_F$$

$$\|\Delta b_2\|_F = \|\Delta A_2 x\|_F \leq \|\Delta A_2\|_F \|x\|_F \text{ ή } \frac{\|\Delta b_2\|_F}{\|x\|_F} \leq \|\Delta A_2\|_F =$$
$$= \left\| \begin{matrix} (I_1 \Delta I_0)^T (I \Delta I_0)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta I_1)^T (I_0 \Delta I_1) + \\ +(I_1 \Delta I_0)^T (I \Delta I_0)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta I_0)^T (I_1 \Delta I_0) + \\ +(I_0 \Delta I_1)^T (I \Delta I_1)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta I_1)^T (I_0 \Delta I_1) + \\ +(I_0 \Delta I_1)^T (I \Delta I_1)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta I_0)^T (I_1 \Delta I_0) \end{matrix} \right\|_F \leq$$
$$\leq \left\|(I_1 \Delta I_0)^T (I \Delta I_0)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta I_1)^T (I_0 \Delta I_1)\right\|_F +$$
$$+\left\|(I_1 \Delta I_0)^T (I \Delta I_0)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta I_0)^T (I_1 \Delta I_0)\right\|_F +$$
$$+\left\|(I_0 \Delta I_1)^T (I \Delta I_1)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta I_1)^T (I_0 \Delta I_1)\right\|_F +$$
$$+\left\|(I_0 \Delta I_1)^T (I \Delta I_1)(I_{e1}' \Delta IDCT)(I_{e1}' \Delta \varepsilon)(I_{e1}' \Delta DCT)(I \Delta I_0)^T (I_1 \Delta I_0)\right\|_F$$
$$with: I_0 = I \Delta e_0$$
$$and: I_1 = I \Delta e_1$$

11. The first equation represents the error of equation 5.9 and the second the error of equation 5.10:

$$\|(D_1 \Delta D_1)(I \Delta \varepsilon) + (D_2 \Delta D_2)(I \Delta \varepsilon) + (D_2 \Delta D_1)(I \Delta \varepsilon) + (D_1 \Delta D_2)(I \Delta \varepsilon)\|_F =$$
$$= \|[(D_1 \Delta D_1) + (D_2 \Delta D_2) + (D_2 \Delta D_1) + (D_1 \Delta D_2)](I \Delta \varepsilon)\|_F =$$
$$= \sqrt{2(\frac{m^2}{64})} = \frac{\sqrt{2}m}{8} \simeq 0.176m$$

$$\|(I'\Delta \varepsilon)(D_1 \Delta D_1) + (I'\Delta \varepsilon)(D_2 \Delta D_2) + (I'\Delta \varepsilon)(D_2 \Delta D_1) + (I'\Delta \varepsilon)(D_1 \Delta D_2)\|_F =$$
$$= \|(I'\Delta \varepsilon)[(D_1 \Delta D_1) + (D_2 \Delta D_2) + (D_2 \Delta D_1) + (D_1 \Delta D_2)]\|_F =$$
$$= \sqrt{4*2*(\frac{m^2}{4*64})} = \frac{\sqrt{2}m}{8} \simeq 0.176m$$

8 References:

[1] Tommy Olsen and Jo Steinar Strand "An improved image processing chain for mobile terminals" Agder University College 2002.

[2] Chin Chye Koh and Sanjit K. Mitra "Compression of Bayer Color Filter Array data" University of California 2003.

[3] Chin Chye Koh, Jayanta Mukherjee, Sanjit K. Mitra "New Efficient Methods of Image Compression in Digital Cameras with Color Filter Array" University of California 2003.

[4] Dipl. Ing. Uwe Furtner "Color Processing with Bayer Mosaic Sensors" Matrix Vision 2001.

[5] Ting Chen "A Study of Spatial Color Interpolation Algorithms for Single-Detector Digital Cameras" Stanford University 1999.

[6] Richard Lyon and Paul Hubel "Eyeing the camera: into the next century" Foveon Inc.

[7] www.jpeg.org Official JPEG Site

[8] Notes of "Digital Image Processing" Lesson of Technical University Of Crete

[9] Carl Mayer "Matrix Analysis and Applied Linear Algebra"

[10] Alan Laub "Matrix Analysis for Scientists and Engineers"

[11] Huaian Diao and Yimin Wei "Structured perturbations of group inverse and singular linear systemwith index one" Fundan University 2004

[12] Gene Golub and Charles Van Loan "Matrix Computations"

[13] Gilbert Strang "Linear Algebra and its Applications"

[14] Alan Oppenheim and Ronald Schafer "Discrete-Time Signal Processing"

[15] http://www.jpeg.org/jpeg2000, Official site of the JPEG Committee