AN FPGA IMPLEMENTATION OF REAL-TIME ELECTRO-OPTIC & IR IMAGE FUSION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

İBRAHİM MELİH ÇÖLOVA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2010

Approval of the thesis:

AN FPGA IMPLEMENTATION OF

REAL-TIME ELECTRO-OPTIC & IR IMAGE FUSION

submitted by **İBRAHİM MELİH ÇÖLOVA** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan ÖZGEN _____

Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet ERKMEN _____

Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Gözde Bozdağı AKAR _____

Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Assoc. Prof. Dr. Aydın ALATAN _____

Electrical and Electronics Engineering Dept., METU

Prof. Dr. Gözde Bozdağı AKAR _____

Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Mehmet Mete BULUT _____

Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. İlkay ULUSOY _____

Electrical and Electronics Engineering Dept., METU

Ali Erkin ARSLAN _____

MGEO ETM, ASELSAN

Date: 02.09.2010

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : İBRAHİM MELİH ÇÖLOVA

Signature :

ABSTRACT

AN FPGA IMPLEMENTATION OF
REAL-TIME ELECTRO-OPTIC & IR IMAGE FUSION

Çölova, İ.Melih
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Gözde Bozdağı Akar

September 2010, 110 pages

In this thesis, a modified 2D Discrete Cosine Transform based electro-optic and IR image fusion algorithm is proposed and implemented on an FPGA platform. The platform is a custom FPGA board which uses ALTERA Stratix III family FPGA. The algorithm is also compared with state of the art image fusion algorithms by means of an image fusion software application GUI developed in Matlab®.

The proposed algorithm principally takes corresponding 4x4 pixel blocks of two images to be fused and transforms them by means of 2D Discrete Cosine Transform. Then, the L2 norm of each block is calculated and used as the weighting factor for the AC values of the fused image block. The DC value of the fused block is the arithmetic mean of the DC coefficients of both input blocks. Based on this mechanism, the whole two images are processed in such a way that the output image is a composition of the processed 4x4 blocks.

The proposed algorithm performs well compared to the other state of the art image fusion algorithms both in subjective and objective quality evaluations. In hardware,

the implemented algorithm can accept input videos as fast as 65 MHz pixel clock with a resolution of 1024x768 @60 Hz.

Keywords: Imaging, Video Fusion, Image Fusion, Discrete Cosine Transform, VHDL, wavelet, FPGA

ÖZ

APKD ÜZERİNDE GERÇEK ZAMANLI ELEKTRO-OPTİK VE KIZILÖTESİ GÖRÜNTÜ FÜZYONU

Çölova, İ.Melih
Yüksek Lisans, Elektrik-Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Eylül 2010, 110 sayfa

Bu tezde, elektro-optik ve kızılötesi video kaynaklarının değiştirilmiş 2 boyutlu ayrık kosinüs dönüşümü temelli füzyon algoritması önerilmiş ve önerilen algoritma APKD üzerinde gerçeklenmiştir. Algoritmanın gerçeklenmesi için kullanılan platform, Altera Stratix III ailesinden bir APKD barındırmaktadır. Önerilen algoritma aynı zamanda Matlab üzerinde geliştirilen bir kullanıcı arayüz programı ile diğer kabul görmüş füzyon algoritmaları ile karşılaştırılmıştır.

Önerilen algoritma, öncelikle her giriş görüntüsünün ilişkili koordinatlarından 4x4 pixel boyutunda kesintiler alır ve bu kesintilere 2 boyutlu ayrık kosinüs dönüşümü uygular. Daha sonra, her iki kesinti için "L2 norm" değeri hesaplanarak, bulunan değer AC değerlerin füzyonunda ağırlık katsayısı olarak kullanılır. DC değerlerin füzyonu ise giren görüntülerin aritmetik ortalamasıdır. Bu mekanizma kullanılarak bütün görüntü 4x4'lük kesitler halinde işlenir.

Önerilen algoritma hem öznel, hem de nesnel olarak diğer kabul görmüş görüntü füzyonu algoritmaları ile benzer bir performans sergilemektedir. Donanımsal olarak,

önerilen algoritma 65 MHz saat frekansı hızına ve 1024x768 çözünürlüğe sahip videoları 60 fps hızda işleyebilir.

Anahtar Kelimeler: Video Füzyonu, Resim Füzyonu, Ayrık Kosinüs Dönüşümü, VHDL, Dalgacık, APKD(Alan Programlanabilir Kapı Dizisi)

To My Parents …

ACKNOWLEDGMENTS

Before begining; declaring sympathies and thankfullness to whom contributed this work is a great pleasure for me.

First and foremost, I would like to state gratefulness to my thesis supervisor. Gözde Bozdağı Akar who has maintained her support, knowledge and patience throghout the thesis study.

I also express my gratitude to my collagues Ali Erkin ARSLAN, Umur AKINCI and Serkan SÖZEN who helped me and supervised me with their strong theoritical background.

I offer my best appreciation to my brother Engin Çölova for his endless support and encourageous attitude which helped me think positively in any circumcitances.

Lastly, and most importantly, I wish to thank my parents, Gülseren and Nizamettin Çölova. They raised me, took care of me, taught me, support me. To them, I dedicate this thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AC	Alternating Current
CE	Cross Entropy
CPT	Contrast Pyramid Transform
DC	Direct Current
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DMT	Digital Monitor Timing
DSP	Digital Signal Processor (Processing)
DVI	Digital Visual Interface
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
FSD	Filter Subtract Decimate
GUI	Graphical User Interface
GPT	Gradient Pyramid Transform
HDL	Hardware Description Language
HDR	High Dynamic Range
HE	Histogram Equalization
HSI	Hue Saturation Intensity
HVS	Human Visual System
IDCT	Inverse Discrete Cosine Transform
IC	Integrated Circuit
IF	Image Fusion
IFS	Image Fusion Software
IR	Infrared
JPEG	Joint Picture Experts Group
LLP	Left Lane Processor

MAC	Multiply and Accumulate
MDBF	Multiscale Decomposition Based Fusion
MI	Mutual Information
MPT	Morphological Pyramid Transform
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PDF	Probability Density Function
PLD	Programmable Logic Device
RAM	Random Access Memory
RGB	Red Green Blue
RLP	Right Lane Processor
ROLP	Ratio of Low-pass Pyramid
RPT	Ratio Pyramid Transform
SD	Standard Deviation
SDRAM	Synchronous Dynamic Random Access Memory
SRAM	Synchronous Random Access Memory
UIQI	Universal Image Quality Index
VPB	Video Processing Block
VESA	Video Electronics Standards Association
VHDL	VHSIC Hardware Description Language
VME	Versa Module Europe
VPBD	Video Processing Board

CHAPTER 1

INTRODUCTION

1.1 General

Imaging and its applications are one of the primary concerns of today's technology and innovation. Various types of imaging tools and algorithms are being developed to have a better understanding of what is really happening in the visual scene which is observed on any media. On the other hand; in some applications, one image acquired from a specific scene or situation is not enough. Therefore engineers try to maximize the information carried on an image or video.

Data and image fusion algorithms have been developed for decades to meet the needs to gather utmost information from a video or an image. For this purpose, sometimes multiple cameras, each of which has different sensor characteristics are put together to investigate the visual situation thoroughly. In some cases just one capture device – possibly a camera– is used with different sensor settings. These tricks are applied just to gather more information from a field of view.

Data fusion is the key point for all the problems that cannot be handled with just one source. In imaging technology, data fusion is applied in such a way that multiple images are fed to a synthesizer which produces a single output while holding the information carried by all the inputs. Multiple images means that either one camera shoots more than one image with different exposures or two different cameras with different sensor characteristics (electro-optic, infrared camera etc.) capture the same

scene. In either case, before data fusion, mechanical or electrical registration should be done between images. Even though there is an extensive research can be found in literature on image registration (1)(2); this literature is not included to the thesis for the sake of simplicity since pre-registered images are used throughout the thesis.

The main idea in image fusion is to collect all the valuable information from each image and put them together in the resulting fused image. Valuable information refers to the parts of the picture that is attached to HVS (Human Visual System). It is known that HVS is more sensitive to the visually salient regions in an image. Usually, the salient regions are formed by the contrast differences, in other words the edges in the image. Signal processing theory tells that the edges in an image lie on the higher frequencies. In this thesis, we show that in order to achieve pertinent image fusion results, we need to reveal the high frequency components of the fusion images.

In pixel level, image fusion is carried in two ways; one being the multiscale and the other being the non-multiscale approaches. In multiscale approaches, the main idea is exactly the same which is stated above. For instance, in pyramid approaches, while building up the image pyramid, we assure that as the pyramid level goes higher, the high frequency components are exposed much more. In fact, if an infinite (that is the size of the low-pass filter) depth image pyramid was constructed then the image would be analyzed to its finest detail; per say to the tinniest high frequency component.

The situation is closely the same in wavelet approaches. While constructing the decomposition image, it is for sure that as the decomposition level goes higher, the wavelets reveal more edges in the detail images. One can go as deep as the size of the image is reduced to the size (in fact length) of the wavelet in use. However, in practice, it is not that applicable since the size of the image is most of the time at least 10-20 times greater than that of the wavelet. Actually, most of the time, the result is more or less the same after fourth or fifth level decomposition. Therefore,

while applying wavelet transform, 4-5 levels of decomposition gives satisfying results.

In monochrome image fusion, it is stated and proved in many papers (3)(4) that multiscale approaches give better results in comparison to the other pixel level image fusion methods. Despite this fact, applying multiscale transforms in real time is a rather difficult job since one has to process the entire video layer in order to move to the other one.

For instance; if the algorithm is required to fuse two input cameras with the refresh rate of 60 Hz then, naturally the algorithm has to process the video frame in less than 1/60 seconds which is 1.67 milliseconds. In most of the cases, if today's hardware technology is concerned; it is really impossible to proceed through the decomposition level since processing one layer occupies a significant amount of the one frame period which is 1.67 msec.

In 2D discrete wavelet transform and pyramid approaches, the problem is rather solvable since as the decomposition level increases, the size of the layers is decreased fourfold allowing to accomplish less computations in both decomposition and reconstruction stages.

In shift invariant A-Trous wavelet transform, the situation gets worse since the size of the picture remains the same regardless of which direction is traversed. This situation makes real time implementation of A-Trous wavelet transform nearly impossible.

Another major drawback of multiscale decomposition schemes is the memory allocation. In software, image processing on a recorded video is of no concern in terms of memory allocation and operation time. On the other hand, in real time operating image fusion, timing closure and memory allocation come into play and they do matter for realization.

Actually, there are two main ways to implement a signal processing algorithm operating in real time. The first way is to use DSP (Digital Signal Processor) board to implement signal processing algorithm. In order the image to be processed, it has to be kept somewhere; for instance in a RAM (Random Access Memory), on the board. In case of multiscale image fusion, apart from the original image, all the layers of the pyramid has to be stored in a RAM structure to either be further processed or decomposed. If we think of traditional signal processing electronic boards, it is not usual to have that much memory space which restricts the designer to limit the level of decomposition resulting an inefficient image fusion output.

The second way to implement a signal processing algorithm in real time is to use programmable gate array structures, shortly FPGAs. In this case, if a multiscale decomposition is concerned, then the FPGA (Field Programmable Gate Array) has to have an interface with an external memory to be able to keep the actual video frame and its siblings (pyramid layers). Even if such a development environment is present, realization of the algorithm with any type of hardware description language is not trivial and brings lots of difficulties to the designer. At least, the designer has to develop the external memory interface and construct a clever read write cycles. Again this type of design is cumbersome and costly.

Because of the above stated discrepancies of the current multiscale image transform techniques, an algorithm which works equally in software and has advantages in hardware implementation is of no doubt desirable.

This thesis is based on the fact that image fusion algorithms developed so far are not saturated in terms of performance and efficiency. Consequently, a new image fusion technique is proposed herein which will create a new dimension in image and video fusion technology.

1.2 Scope of the Thesis

This thesis firstly investigates most of the monochromatic state of the art image fusion techniques and then the proposed DCT (Discrete Cosine Transform) based algorithm is developed and the results are compared in terms of objective and subjective test methods.

The proposed algorithm works only for monochromatic images. The reason why such a restriction is applied to the scope is that the algorithm is designed to work in hardware basically. Plus, in many hardware platforms, especially in military applications, the output image is monochrome. Naturally, as the technology evolves forward, the data widths and speeds increase together allowing color image technology to take place of monochrome imaging technology

In recent fusion methods, it is observed that red, green and blue color channels come into play in order to dissipate the information such that RGB (Red Green Blue) color space spans the whole human visual perception. In fact, false color image fusion techniques serve best in terms of efficiency and speed. Chapter 2 discusses the state of the art three color band fusion techniques.

Followed by the analysis of current image fusion algorithms, a transform based image fusion algorithm is presented and implemented on FPGA.

The proposed algorithm is first evaluated in *Image Fusion Software* with respect to other fusion algorithms; it is then implemented on a real time video processing board. This thesis compares the derived algorithm in terms of both subjective and objective approaches, which are explained in chapter 3.

1.3 Outline of the Dissertation

This thesis is composed of five main chapters:

First chapter explains the main scope of the thesis and makes a brief introduction to the subject.

Second chapter deals with the literature of image fusion while giving utmost detailed explanation of state-of-the-art fusion methods. This chapter also prepares a base line for the next chapter stressing all the advantages and disadvantages of the well known fusion algorithms.

Third chapter constitutes the main metaphor of the whole thesis. It explains the proposed algorithm thoroughly and presents various objective and subjective performance metrics.

Fourth chapter explains the implemented hardware for the proposed algorithm. It makes a clear summary of the hardware realization making use of block diagrams and flowcharts.

Chapter 5 makes a particular summary of the whole idea presented in the thesis. It also points to the weaknesses of the developed an algorithm and its hardware implementation while offering alternative solutions accordingly.

In “APPENDIX A”, a basic FPGA design flow is described in to have a better understanding of Chapter 4.

“APPENDIX B” describes the Image Fusion Software developed to investigate the behavior of various image fusion approaches developed so far. It also implements the proposed algorithm to be able to see and benchmark the result with that of others.

“APPENDIX C” gives a short lecture about the hardware platform used to implement proposed fusion algorithm. And, “APPENDIX D” summarizes the resource usage of the implemented design.

CHAPTER 2

LITERATURE OF IMAGE FUSION

2.1 Introduction

Image fusion is basically a subset of data fusion which handles blending of multiple images into one. Image fusion literature hosts a wide literature index in several methods and aspects. Since the invention of the image fusion idea, there are multiple approaches developed, tested and verified. They all work in different manners although the main idea is the same for every one of them which is extract and transfer most valuable data to the output image (5).

There is a hierarchy present in image fusion methodologies starting from the most trivial one going to the most abstract one. The basic level of image fusion pyramid is the pixel level fusion where fusion is done in pixel basis; the second step is feature level which handles the subject by extracting certain details like edges and lines. The third layer of the pyramid is occupied by object level fusion which applies image fusion by defining and extracting certain, predefined shapes like human faces or trees. Symbol level fusion lies in the last and the top level of the pyramid. In this stage image fusion is done mainly theoretically trying to describe images in terms of symbols.

The image fusion pyramid can be drawn as:

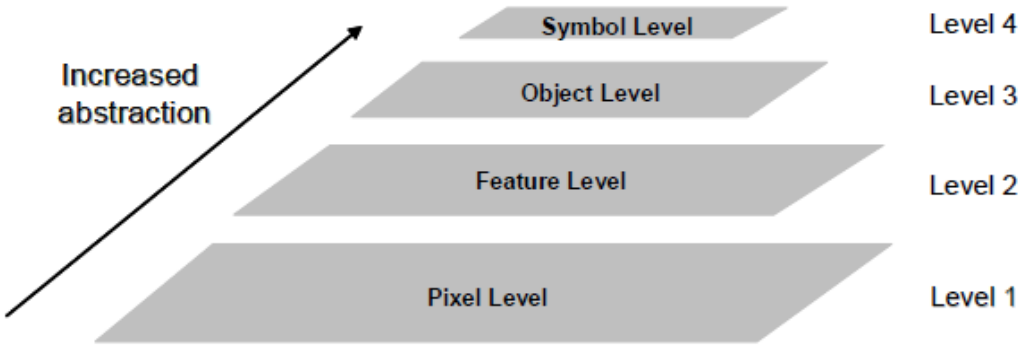


Figure 1 Image Fusion Hierarchy

In this framework, we will mostly deal with pixel based methods since they are the most common and their performance is proven in both software and hardware.

2.2 Pixel Based Methods

Let $I_1(x,y)$ *and* $I_2(x,y)$ are two input images acquired with different techniques. We assume that they are aligned in all aspects and ready to be fused. Further, assume that $I_f(x,y)$ is the output image. The process of blending $I_1(x,y)$ *and* $I_2(x,y)$ into $I_f(x,y)$ is called image fusion(6).

In pixel level fusion, there are two main methodologies namely, non-multiscale and multiscale decomposition. In non-multiscale decomposition, the input images $I_1(x,y)$ *and* $I_2(x,y)$ are combined in a function in order to create the fused output image $I_f(x,y)$. There is no transform domain crossing in the whole process. If function *f is* the fusion operator, then $I_f(x,y)$ is written as:

$$I_f(x,y) = f\big(I_1(x,y), I_2(x,y)\big) \qquad \textbf{2-1}$$

In multiscale transforms, the input images are first transformed to any transform domain, then inverse transformed to the time domain after image fusion rules are applied. If *T* denotes the transform operator:

$$I_f(x,y) = T^{-1}\{f(T\{I_1(x,y)\}, T\{I_2(x,y)\})\} \quad \textbf{2-2}$$

Here, both methods are formulated only with two inputs; however the cardinality of inputs can be increased as long as the fusion operator *f* complies with the inputs.

2.2.1 Non-multiscale Decomposition *Based Methods*

2.2.1.1 Pixel Averaging

This method is applicable to time domain pixel level fusion as well as to transform domain pixel level fusion. The idea is basically to calculate (weighted) average of the inputs and reflect the result directly to the output.

However, pure pixel averaging presents major problems such that an equal contribution of both images usually leads following situation: A poor quality (low content) sensor image obscures good quality (high content) sensor image data without adding any value to the image. (7) This result is unavoidable since one shall not know in any image fusion system that which capture device potentially serves more information than the other. Usually, this type of image fusion results with a distorted output which accommodates undetermined data unless the quality of the sensors is determined statistically. This determination is done by calculating the intensity standard deviation under the assumption that greater deviations result from more scene content present in the sensor image. (7) By doing so, the weighting constants can be tuned fairly.

The mathematical expression for pixel level averaging is:

Let c_0 and c_1 are weights which are pixel independent. Then the fused output for images I_1 and I_2 is:

$$I_F = \frac{(c_0 * I_1) + (c_1 * I_2)}{c_0 + c_1} \quad \textbf{2-3}$$

2.2.1.2 Max-Min Method

This method is a non-linear operation since it uses max or min operator. Basically, in this methodology image fusion is simply handled by taking the greatest absolute value of the two input images while keeping the sign at the output.(8) This method can be applied both to time domain and transform domain analysis.

The mathematical expression for max-min method is (9):

$$I_F(x,y) = max(I_1(x,y), I_2(x,y)) \quad \textbf{2-4}$$

or,

$$I_F(x,y) = min(I_1(x,y), I_2(x,y)) \quad \textbf{2-5}$$

2.2.1.3 Color Composite Fusion

The human visual system is very sensitive to the main colors red, green and blue. Many image fusion techniques manipulate this fact to enhance the information in the output fused image. Therefore, these methods are also named as false color techniques.

In 1996, Toet & Walvaren(10) came up with an idea of a false color technique which reveals the unique components of each input images at the time. This algorithm works only for 2 input images. If there are more than 2 input images, this algorithm is not applicable. The algorithm uses the common component of input images by taking the minimum value of pixels among all the input images. Then the common component is subtracted from all input images divulging the unique components of inputs. Each unique component is then subtracted from the other input image and assigned to one of the R and G channels. After, the B channel is rather assigned to

the absolute difference of the unique components to further emphasize the details or simply assigned to 0.

In 2002, Toet & Franklen (11) tried a simple way of assigning RGB colors to each of the input channels. In the case with 3 inputs, every input channel is assigned to one of the RGB color bands. In case of 2 input images, one image is assigned to the R channel, whereas the other input is assigned to both G and B channels. This algorithm has no solution when the input images are more than 3.

In 2007, Huang et al (12) based on Land's experiment. This algorithm also works for only 2 images. In this algorithm, two matrices called D (difference) and S (summation) are created by calculating the sum and the absolute difference of the input images. Then, D and S matrices are normalized to the 0-255 scale. Afterwards, normalized S matrix is assigned to the R channel and the normalized D matrix is assigned to B channel. G channel is occupied by the normalized electro-optic input image, assuming that the mean of the fused image lies on the electro-optic image not IR (Infrared).

In 2008, J. H. Jang and J. B. Ra (13) proposed an algorithm which first fuse the two monochrome images using pyramid transform and then uses the result as the I component of the HSI (Hue Saturation Intensity) color space. After creating the "I" component, the "H" component is created with a selection rule. Lastly, the "S" component is computed with an optimized formula using only the "H" component. This transformation enables the monochrome fusion images to be visible in RGB domain with contrasting the colors that has opponent information mutually.

Nunez et al (14) use color to fuse a high-resolution panchromatic image with a low-resolution multispectral image by adding the wavelet coefficients of the high-resolution image to the intensity component of the multispectral image.

Another technique utilizing color is based on opponent-color processing which maps opponent-sensors to human opponent colors (red vs. green, blue vs. yellow) (15). This technique has been used in the field of night time surveillance. For example; Waxman, Aguilar et al (16)(17), use a neural network to fuse a low-light visible image and a thermal IR image to generate a three-channel false color image used for night operation.

Lastly, a simple but effective false color fusion is developed within this thesis. It basically calculates the *min*, *max* and *mean* values of the two input images and assigns the *min* component to the R channel, *max* component to the G channel and *mean* to the blue channel. The related visual results of the explained algorithms are shown in chapter 3.4.3

2.2.1.4 Principal Component Analysis (PCA) Method

Linear methods such as average and principal component analysis are based on a weighted superposition of the source images.

In the PCA method, the optimal coefficients (in terms of information content and redundancy elimination) are calculated using a Karhunen – Loeve transform of the intensities. The coefficients for each source image are obtained from the normalized eigenvector associated with the largest eigenvalue of the covariance matrix of the two source images (15).

Theoretically (18), assume that image *I* is composed of *n* wave bands such that:

$$I = [i_1, i_2, i_3, \dots, i_n] \qquad \textbf{2-6}$$

And, the variance between bands is:

$$\delta_{ij}^2 = E\big[(i_i - m_i)\big(i - m_j\big)\big] \qquad \textbf{2-7}$$

Here, *i,j = 1,2,3,…,n* and m_i and m_j are the means of the corresponding band. Then the symmetric covariance matrix $\sum$ is:

$$\Sigma = \begin{bmatrix} \delta_{1,1} & \cdots & \delta_{1,n} \\ \vdots & \ddots & \vdots \\ \delta_{n,1} & \cdots & \delta_{n,n} \end{bmatrix} \quad \textbf{2-8}$$

Then, the eigen vectors are computed from the diagonalized covariance matrix and the eigenvectors vector is given as:

$$\emptyset_n = [\emptyset_1, \emptyset_2, \emptyset_3, \dots, \emptyset_n]^T \quad \textbf{2-9}$$

Then *I* is mapped onto $\emptyset_n$, resulting:

$$Y = I.\emptyset_n \quad \textbf{2-10}$$

Here *Y* is a vector composed of principal components y_i's where *i=1, 2, 3,…, n.*

The PCA based fusion applies just the same way as the false color method. The input images are matched with one of the principal components and the result is inverse PCA transformed to get the fused image. This operation can be formulated as follows:

Let I_1 and I_2 are two source images whose normalized principal components are Y_{I_1} and Y_{I_2}.

$$Y_{I_1} = [Y_{I_1 1}, Y_{I_1 2} \cdots Y_{I_1 n}] \quad \textbf{2-11}$$

and

$$Y_{I_2} = [Y_{I_2 1}, Y_{I_2 2} \cdots Y_{I_2 n}] \quad \textbf{2-12}$$

Therefore, the fused principal component vector I_F is formed such that:

$$I_{F_i} = \begin{cases} I_{1_i}.Y_{I_{1_i}}, & Y_{I_{2_i}} < Y_{I_{1_i}} \\ I_{2_i}.Y_{I_{2_i}}, & Y_{I_{2_i}} > Y_{I_{1_i}} \end{cases} \quad \textbf{2-13}$$

2.2.1.5 DCT Based Fusion Methods

There are two prominent papers implementing image fusion by means of 2D DCT technique. In chronological order, Jinshan Tang and Zafar et al developed primitive approaches to the image fusion.

2.2.1.5.1 Method by Jinshan Tang

In 2003, Jinshan Tang developed a new methodology about image fusion on DCT basis (19). His proposition was there is no difference between his algorithm and the classical wavelet approach in terms of visual output quality. Since DCT is more time saving and easier to implement; he stresses that his algorithm should be decent replica of the wavelet approach. He also claimed that from the nature of the DCT, the fused image in the transform domain is ready to be packed as a JPEG (Joint Picture Experts Group) image and sent through any protocol with a significant reduced size in terms of data.

Tang uses traditional $8x8$ DCT transformation method just like in JPEG framework. The rest of the algorithm is explained below (19):

Let there are *W* input images namely, I_1, I_2,⋯, I_w. Further, suppose that I_1^D,I_2^D,⋯, I_W^D are 2D DCT of , I_1, I_2,⋯, I_w. Here, the parameters *k,l* and *i* are defined such that I_1^D*(k,l,i)* represents *i*th DCT block of image I_1 where *(k,l)* denotes the 8x8 block inner index being max(*k*)= 7, and max(*l*)= 7.

In an image fusion technique utilizing DCT, there are two main constants remain to be defined. One of them is the DC coefficient and the second one is the AC coefficients. In an 8x8 block, the upper-left component of the DCT is the DC coefficient; the remaining 63 coefficients are the AC coefficients.

In this algorithm, Tang uses the following formula to extract the AC value of any block in the fused image.

Let $D_{i,k,l}^{F}$ denotes the ith block of the fused image D^{F} where (k, l) is the block index.

Then, then AC coefficients are:

$$D_{i,k,l}^{F} = max_{t}\{I_{i,k,l,t}^{F}\} \quad \textbf{2-14}$$

Here, the parameter t defines the index of inputs being $\max(t) = W$.

Then letting W is the number of input images to be fused, the DC coefficients become:

$$D_{n,0,0} = \frac{1}{W}\sum_{t=1}^{W} D_{n,0,0}^{t} \quad \textbf{2-15}$$

The above explained algorithm is the first way with DCT based image fusion; moreover he has yet another way for better optimization. He claims that the following one races better with DWT.

The algorithm is just a slight modification of the previous one. In the previous methodology, the maximum of the input AC coefficients were selected, however in

here, the arithmetic average of AC coefficients from all inputs is evaluated and reflected to the output.

2.2.1.5.2 Method by I. Zafar et al

In 2006, Zafar developed another algorithm about image fusion based on DCT (20). His goal was to fuse images which have different exposures. He claimed that building HDR (High Dynamic Range) systems is much more expensive than first capturing and then fusing all the input which have different exposures. This case is valid for stationary scenes that have temporal difference between taken frames.

Basically, his main goal was in line with that of the Tang. But, Zafar et al. wanted to optimize the fusion scheme so that it would be suitable for multi-exposure fusion. He used more than two images and generalized the idea so that it is able to handle five different inputs. In fact, the proposed method does not limit the number of inputs.

Apart from Zafar's method, he tried to estimate the value of information stored in each block of the input image. For this purpose, he used L2 norm of AC coefficients. His methodology can be explained symbolically as follows:

Let I^{--}, I^{-}, I, I^{+} and I^{++} are the set of input images with different exposure/focus settings (--) being the most under exposed and (++) the most overexposed image. The idea is to fuse these 5 images to get a HDR output. Furthermore, assume that B represents both AC and DC coefficients of a given kxk image block. To be more explicit:

$$B = (dc, ac_1, \cdots, ac_{k^2-1}) \qquad \textbf{2-16}$$

Where, *dc* is the DC coefficient and ac_n is the *n*th coefficient of the kxk sized block. Moreover, one can expand the above equation so that it covers the whole image.

If I^{--}, I^{-}, I, I^{+} and I^{++} are to be expressed in terms of B in transform domain, then:

$$B^{--} = (dc, ac_1, \cdots, ac_{k^2-1}) = (dc^{--}, ac^{--})$$
$$B^{-} = (dc, ac_1, \cdots, ac_{k^2-1}) = (dc^{-}, ac^{-})$$
$$B = (dc, ac_1, \cdots, ac_{k^2-1}) = (dc, ac)$$
$$B^{+} = (dc, ac_1, \cdots, ac_{k^2-1}) = (dc^{+}, ac^{+})$$
$$B^{++} = (dc, ac_1, \cdots, ac_{k^2-1}) = (dc^{++}, ac^{++}) \quad \textbf{2-17}$$

Here, B^{--}, B^{-}, B, B^{+} *and* B^{++} are DCT transform of any specific block of I^{--},I^{-},I, I^{+} *and* I^{++}. As the last step, the fused block B^{*} is defined such that:

$$B^{*} = (dc^{*}, ac^{*}) \quad \textbf{2-18}$$

The fusion decision rule is defined as follows:

$$dc = average(dc^{--}, dc^{-}, dc, dc^{+}, dc^{++}) \quad \textbf{2-19}$$

and

$$ac = \max(|ac^{--}|, |ac^{+}|, |ac|, |ac^{+}|, |ac^{++}|) \quad \textbf{2-20}$$

Here, $|ac|$ is defined as the L2 norm of *ac*.

L2 norm of a vector is also defined as *Euclidean norm*. Suppose that *x* is a complex vector defined as:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \textbf{2-21}$$

The L2 norm of *x* is defined as:

$$|x| = \sqrt{\sum_{k=1}^{n} |x_k|^2} \qquad \textbf{2-22}$$

The above explained fusion process is repeated for all the blocks, which have the same block index, of the input images The result is then inverse transformed using Inverse Discrete Cosine Transform (IDCT) and finally HE (Histogram Equalization) is applied to the resulting image. This procedure can also be expanded for use of image compression. Just before the IDCT step, the fused image can be quantized and entropy coded for transmission or storage.

2.2.2 Multiscale Decomposition Based Fusion Methods

Fusion with multiscale decomposition uses the fact that images are formed in multiple layers in terms of detail. At each level, there is a hidden detail. Once these details from each layer come together, they form the original image with full detail. MDBF (Multiscale Decomposition Based Fusion) methods firstly decompose all input images – in our case two – and perform a fusion technique for each decomposition level.

2.2.2.1 Advantages of Multiscale Decomposition

Multiscale signal analysis has been one of the primary concerns of the image processing since it has various useful relations with the physical life.

There are mainly four reasons which make multiscale signal decomposition valuable (21):

i. HVS threats the information in a multiscale fashion
ii. Signals are composed of different structures at different scales.
iii. Sensors provide information from the same scene at different scales.

iv. Multiscale approaches seem to be computationally efficient and robust.

2.2.2.2 Theory of Multiscale Decomposition

Multiscale decomposition can be explained mathematically as follows (21):

Let J be a scalar and finite number showing the decomposition scheme. Moreover, assume that there exists a domain V_j assigned at each level J. In this framework, assume that signal analysis proceeds forwards with increasing J and this is achieved by analysis operators $\Psi\uparrow_j$ such that:

$$\Psi\uparrow_j : \mathrm{V}_j \rightarrow \mathrm{V}_{j+1} \qquad \textbf{2-23}$$

On the other hand, assume that signal synthesis proceeds backwards with decreasing J and this is achieved by *synthesis* operators $\Psi\downarrow_j$ such that:

$$\Psi\downarrow_j : \mathrm{V}_{j+1} \rightarrow \mathrm{V}_j \qquad \textbf{2-24}$$

Here, $\Psi\uparrow_{\mathrm{j}}$ operator reduces the signal scale and reveals the information of signal at one level higher, whereas $\Psi\downarrow_j$ works just in opposite way. By doing so, one can construct an image pyramid by successively using $\Psi\uparrow_j$ operator. The image pyramid has as many levels as the integer J.

The *analysis* operator naturally reduces signal information and synthesis operator recovers the signal. Since the analysis operator is not invertible; then the signal reconstruction is not lossless in general. Therefore, we define the approximation composition operator $\widehat{\Psi}_{i,j}$ such that:

$$\widehat{\Psi}_{\mathrm{i,j}} = \Psi\downarrow_{\mathrm{j,i}} \ \Psi\uparrow_{\mathrm{i,j}} \qquad \textbf{2-25}$$

Multiscale decomposition methods mostly rely on the above equation. One can construct the pyramid by means of $\Psi\uparrow_{i,j}$ and synthesize it back with $\Psi\uparrow_{i,j}$ operator. The methods described below uses this fact for pyramid construction. The only difference is how the analysis and synthesis operators are derived.

Multiscale image transforms are divided into two main groups which are pyramid transforms and wavelet transforms. Both of them utilize the same mathematics described above; however the former uses various filters to construct the multiscale architecture and the latter uses wavelets to form the multiscale architecture.

Schematically, the above described framework can be shown like following (22):

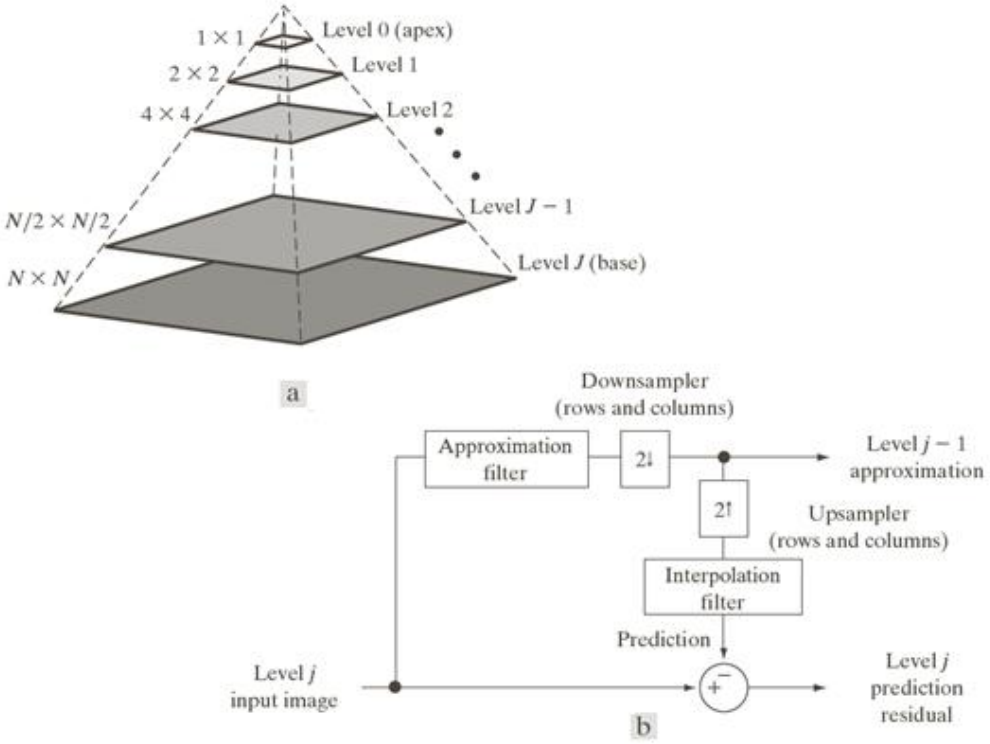


Figure 2 (a) An Image Pyramid (b) A System for Image *analysis* & *synthesis*

2.2.2.3 Pyramid Transforms

The pyramid approach utilizes the mathematical proof of *perfect reconstruction*. However, to derive a computationally efficient and pyramid there are mainly six

pyramid methods developed which are Laplacian pyramid transform, filter-subtract-decimate pyramid transform, ratio pyramid transform, contrast pyramid transform, gradient pyramid transform and morphological pyramid transform. As a first step, all of these methods create the Gaussian pyramid; however, they differ in the later phase which is constructing the image pyramid.

The Laplacian pyramid transform is the first known multiscale image transform introduced by Burt and Adelson (23). The idea was to derive a difference, in fact; a detail pyramid from Gaussian pyramid decomposition. Gaussian pyramid is obtained by a recursive *reduction* of the image data set. By *reduction*; it is meant that the image is low-pass filtered and then decimated.

The idea of *analysis* and *synthesis* operation for Laplacian pyramid can be mathematically explained as follows:

Let I be an image satisfies the condition $I \in V_j$. From equation 2-25 we know that;

$$\hat{I} = \Psi_{j,j+1}(I) = \Psi\downarrow_j \Psi\uparrow_j (I) \in \hat{V}_j \qquad \textbf{2-26}$$

Assume further that there exists a *subtraction operator* $(I, \hat{I}) \rightarrow I \dot{-} \hat{I}$ mapping $V_j \times \hat{V}_j$ into a set Y_j . Here, Y_j is a difference space. Furthermore, we introduce an *addition operator* $(\hat{I}, y) \rightarrow \hat{I} \dot{+} y$ mapping $\hat{V}_j \times Y_j$ into V_j .The detail signal $y = I - \hat{I}$ represents the information which is not present in $\hat{x}$. Here, we can write the following:

$$I_j = y_j + I_{j+1} \qquad \textbf{2-27}$$

This leads us to the recursive signal analysis scheme:

$$I_0 \rightarrow \{y_0, I_1\} \rightarrow \{y_0, y_1, I_2\} \rightarrow \cdots \rightarrow \{y_0, y_1 \ldots, y_j, I_{j+1}\} \qquad \textbf{2-28}$$

Here

$$\begin{cases} I_0 = I \in V_0 \\ I_{j+1} = \Psi \uparrow_j (I_j) \in V_{j+1} \\ y_j = I_j \div \Psi \uparrow_j (I_{j+1}) \in Y_j \end{cases} \quad \textbf{2-29}$$

Notice that, signal *I* can be perfectly reconstructed from I_{j+1} and $y_0, y_1, \ldots, y_j$.

A Laplacian pyramid of a picture is shown below:

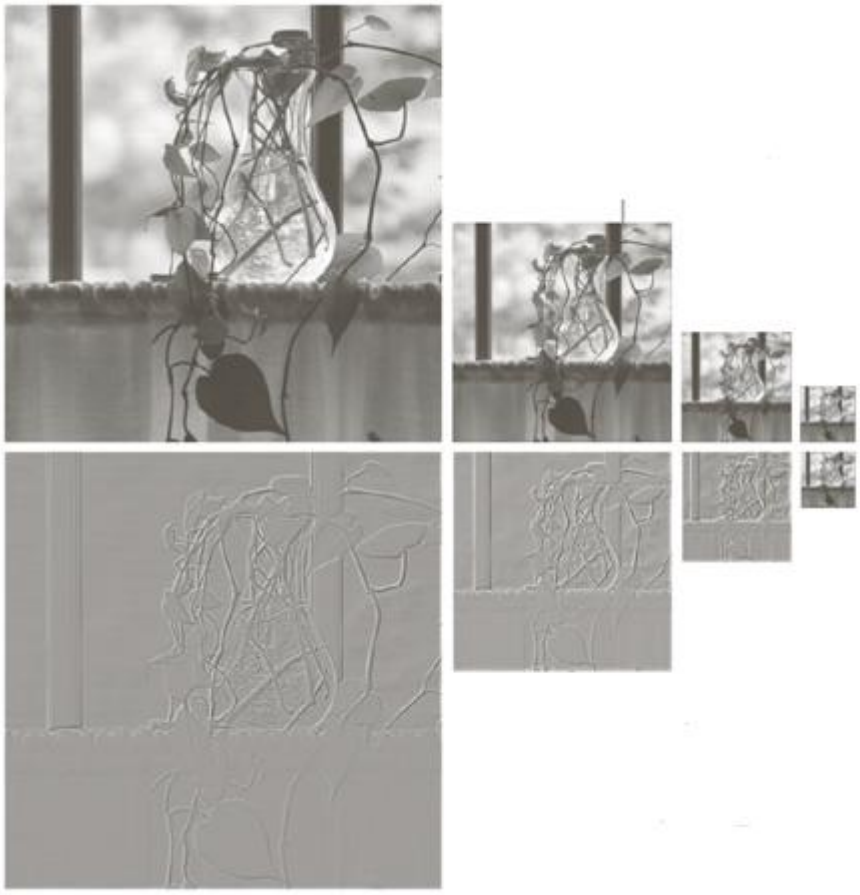


Figure 3 Gaussian & Laplacian Pyramids of an Image(22)

Filter-subtract-decimate pyramid approach is similar to Laplacian pyramid; however it is computationally more efficient (24). FSD (Filter Subtract Decimate) pyramid approach is found and patented by Charles H. Anderson in 1988.

Instead of the Laplacian filter, simpler 1D filter is applied to the image at each layer. The result is subtracted from the lower layer image and decimated by two. Recursively, the pyramid is constructed. (25) In the synthesis part, the highest level of the pyramid is undecimated and interpolated recursively until the bottom of the pyramid.

Ratio pyramid transform is another Gaussian based pyramid transform, invented by A.Toet, which has levels that is the ratio of two successive levels of Gaussian Pyramid (24). This transform is also known as ROLP (Ratio of Low-pass Pyramid).

The difference between Laplacian pyramid and RPT (Ratio Pyramid Transform) is that in Laplacian pyramid, the pyramid is constructed in such a way that a layer is the difference of two successive layers of Gaussian pyramid; here, a layer is constructed by taking the *ratio* of two successive layers.

The synthesis stage is also similar to the idea described above such that the pyramid is undecimated and interpolated and added together with the *ratios* found in the forward step.

Contrast Pyramid Transform is very similar to RPT; however, there is a slight difference in the pyramid construction. CPT (Contrast Pyramid Transform) is invented by Alexander Toet et al. in 1989. In the RPT, the pyramid is constructed by taking the ratio of two successive layers, but in CPT, the pyramid construction is made by the following formula (26) (27):

$$Con_k = \frac{G_k - EXPAND(G_{k+1})}{EXPAND\ (G_{k+1})} \qquad \textbf{2-30}$$

where Con_k represents the contrast between two successive levels G_k and G_{k+1} in the Gaussian pyramids, and operation *EXPAND* consists of a simple up-sampling followed by a low-pass filtering.

Gradient Pyramid Transform also makes use of Gaussian pyramids. Gradient pyramid of an image is obtained by applying gradient operators to each level of the Gaussian pyramid (24).

In GPT (Gradient Pyramid Transform), four different (horizontal, vertical, two diagonal) gradient operators are used to derive the Gradient pyramid from Gaussian pyramid. These four gradient operators are put together to form a single gradient. Unlike Laplacian, this single gradient is used to form the pyramid rather than taking the direct difference.

Milad Ghantous et al. used GPT to fuse IR and electro-optic images with a hybrid image fusion scheme (28).

Morphological Pyramid Transform is yet another way of modifying the Gaussian pyramid. In MPT (Morphological Pyramid Transform), a similar approach to Laplacian pyramid transform is applied. In MPT, first Gaussian pyramid is constructed. After Morphological pyramid is constructed in such a way that a certain morphological (non-linear) filter is applied to the Gaussian pyramid and the result is subtracted from the Gaussian at each level. This iterative method forms the morphological pyramid instead of Laplacian Pyramid.

Morphological filters are usually used for image smoothing and noise removal. In this case, they exhibit a low pass effect yet, they do not affect the salient features in the image (26).

2.2.2.4 2D Discrete Wavelet Transform

In general signal processing, wavelets are used to extract information from an unknown signal. In image processing, wavelets are used to find the salient information of an image.

Wavelets are applied to the image to extract horizontal, vertical and diagonal high frequency terms. In order to achieve this, two different wavelets are applied to the image one being high and the other is low pass effect on the image. The high-pass wavelet is usually denoted with the letter *h* and the low-pass wavelet is denoted with *g*. In order to transform the image to the wavelet domain, the image is convolved with *h* and *g* in both horizontal and vertical directions in turn. The convolution operation has four outputs of different bands which are low-low (LL), low-high (LH), high-low (HL) and high-high (HH) components. Here, LL means the image is filtered in both horizontal and vertical directions by wavelet *g*. LH means the image is convolved with *g* along the rows and convolved with wavelet *h* along the columns and so on. The LL component of the image is the low-pass of filtered version of the image hence; it does not contain any valuable data in terms of salient feature. Instead, it is used as a basis image for a higher depth analysis. The same procedure explained above is applied to each level of the transformation. By doing so the image is divided to its frequency components at each layer.

Below; a 2D wavelet transforms is drawn (22):

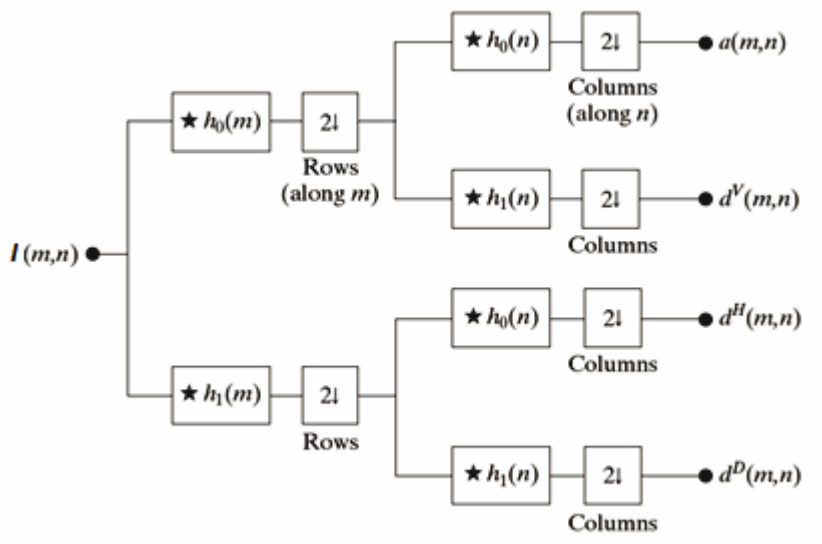


Figure 4 2D Wavelet Filter Bank Implementation

Here, *I(m,n)* represents the input image, h_0 represents the low pass effect wavelet, h_1 represents the high pass effect wavelet, *a* represents LL, d^V represents LH(vertical), d^H represents HL(horizontal) and d^D represents HH(diagonal) detail components of the image.

A decimation stage is applied after each convolution step in 2D wavelet transform. In the above figure, it is illustrated that the image is sub-sampled by two at each wavelet convolution. This down-sampling reduces the data to be processed by 4 times and the lost data is of no importance since it is the replica of the neighbor pixel. With the help of sub-sampling, the image size does not change regardless of the wavelet transform depth. Since, as the transform level increases, the size of the image is decreased by 4 allowing the image size to stay the same at total eventually.

Below, one level of 2D wavelet transform is depicted (22):

Figure 5 One Level of 2D Wavelet Transform

The upper left part of the picture is the LL part of the original picture. It is said that this part is the approximation part of the image and the other three parts are the detail components. Observe that the original image size does not change thanks to the decimation. If one wants to go one level deeper, than he needs to pick up the LL component of the image and apply the same procedure. Below, the third level of transform is shown (22):

Figure 6 Three level 2D Wavelet Transform

The inverse wavelet transform is basically the reverse of the above explained procedure. This process is called *reconstruction.* At each level of the transformation, the DWT coefficients are up-sampled. Up-sampling is mainly inserting zeros between each coefficient of the image. This is also known as *interpolating.* After this step, each image is convolved with the reconstruction filter. Reconstruction filter is

basically the mirror of the original wavelet. In other words, it is the flipped version of the original wavelet. After convolving, all four images are added together to reach one level higher LL image. This process is iterated until the first decomposition level is achieved.

Wavelet transform serves as an excellent multilayer decomposition method; however traditional 2D discrete wavelet transform suffers from shift-variance which affects the resulting image considerably when one or both of the input images are shifted slightly. Shift-variance ambiguity arises from the decimation of the input coefficients after convolution.

In order to eliminate the above explained disadvantage of 2D discrete wavelet transform a shift invariant wavelet transform is developed. Shift invariant A-Trous wavelet transform skips sub-sampling step at each decomposition level and uses a different set of wavelets at each decomposition level (26). This precaution makes 2D wavelet transform immune against any shift in the input images. However, as the decomposition level goes deeper, the space allocated by the transform coefficients grow 4 times at each level which results in shortcomings in memory allocation and calculation efficiencies at most software and hardware implementations.

2.2.2.1 Image Fusion with Multiscale Decomposition

Above mentioned multiscale image decomposition methods serve as a platform where pixel based image fusion rules can apply. After decomposing the input images, a single multiscale image is constructed in terms of fusion and it is inverse transformed to get the final image. The fusion procedures for both pyramid and wavelet based transformation methods are explained below.

2.2.2.1.1 Image Fusion with Pyramid Transforms

Image fusion through pyramid transform is handled after applying one of the above stated pyramid construction methods. After the pyramids are constructed for the input images, first the lowest level of the fused pyramid is calculated. Since the lowest levels of the pyramids represent a low-pass version of the image, usually the mean value of the lowest levels of all the input images is calculated on pixel basis. But in some cases, maximum value is also used. For the higher layers of the fusion pyramid, generally the maximum value among all corresponding layers is selected in pixel basis. The reason why the maximum values are used in the higher levels is that the maximum value represents a high-pass component existence at a given pixel location. After applying fusion, the fused image pyramid is back transformed using inverse pyramid decomposition technique creating the output fused image. This procedure can be formulated as follows:

Let I_1 and I_2 are two input images and I_F is the fused image. Further, assume that ${I_1}^T$, ${I_2}^T$ and ${I_F}^T$ represents the pyramid transforms of the images. Then, ${I_F}^T$ can be rewritten as follows:

$$ {I_F}_n^T(x,y) = \begin{cases} \max\left({I_1}_n^T(x,y), {I_2}_n^T(x,y)\right), & n \neq k \\ \text{mean}\left({I_1}_n^T(x,y), {I_2}_n^T(x,y)\right), & n = k \end{cases} \qquad \textbf{2-31} $$

Here, index *n* determines the *n*th layer component, which can be any integer from 1 to *k*. After, ${I_F}^T$ is constructed, I_F is calculated by inverse pyramid transform.

2.2.2.1.2 Image Fusion with Wavelet Transforms

In 2D DWT based image fusion, the images are first transformed using either shift variant or shift invariant wavelet transforms. After transformation, for the lowest level LL component of the fused image usually the mean value among all inputs is calculated. For the LH, HL and HH components of the fused image at a given layer

the pixel with a maximum value among all inputs is selected. More clearly, the LL components are calculated by the arithmetic average, whereas the other bands are calculated with the local maximum method. This situation enables the DC value of the output fused image has equal contribution from all inputs; however for the AC value, only the high frequency terms in input images are reflected to the output. The resulting fused image is than inverse transformed with the above explained algorithm forming the output fused image. This procedure can be formulated as follows:

Let I_1 and I_2 are two input images and I_F is the fused image. Further, assume that ${I_1}^T$, ${I_2}^T$ and ${I_F}^T$ represents the wavelet transforms of the images. Then, ${I_F}^T$ can be rewritten as follows:

$$ {I_F}^T_{n_i}(x,y) = \begin{cases} \max\left({I_1}^T_{n_i}(x,y), {I_2}^T_{n_i}(x,y)\right), & i \neq LL \\ \text{mean}\left({I_1}^T_{n_i}(x,y), {I_2}^T_{n_i}(x,y)\right), & i = LL \end{cases} \qquad \textbf{2-32} $$

Here, index n_i determines the *n*th layer component, which can be LL, LH, HL or HH. After, ${I_F}^T$ is constructed, I_F is calculated by inverse wavelet transform.

CHAPTER 3

PROPOSED ALGORITHM

3.1 Introduction

As stated in previous sections, image fusion is of high importance topic in image processing. However; it is kind of saturated in terms of methodology framework. It is easy to realize that in order to go further in image fusion, this topic has to be analyzed in all aspects allowing all state of the art techniques to be applied on. Following up this reasoning; this thesis has come up with a new approach to image fusion methodology. It makes use of block-wise operation and 2D DCT. Below, the detailed explanation of the algorithm is depicted and for a better understanding of the theory of the proposed algorithm, a detailed explanation of 2D DCT is also given.

3.2 2D Discrete Cosine Transform

A DCT expresses a finite set of data in terms of cosine waves oscillating at different frequencies with different amplitudes (29). DCT has several applications in science and engineering where frequency spectrum and compression of data is under consideration. This is usually the case for audio and video applications. The reason why DCT is that popular in image processing is the ability of energy compaction.

To be exact, DCT is a modified Fourier Transform Technique, in particular DFT (Discrete Fourier Transform). The only difference between DFT and DCT is the

former uses both real and complex numbers (complex exponentials) whereas the latter uses only real numbers.

The formal definition of DCT is given below:

Suppose a finite set of data namely, $x_0, x_1, \ldots, x_{N-1}$ exists and there exists a function F which maps set of real numbers to set of real numbers.

$$F: R^N \rightarrow R^N \quad \textbf{3-1}$$

Then the function F maps the data set $x_0, x_1, \ldots, x_{N-1}$ again to a finite set of real numbers $X_0, X_1, \ldots, X_{N-1}$ with the below given relation:

$$X_k = F(x_k) = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{\mathrm{N}}\left(n + \frac{1}{2}\right)k\right] \quad \textbf{3-2}$$

where; k = $0, 1, \ldots, N-1$. Here, it is said that the sequence $X_0, X_1, \ldots, X_{N-1}$ is the Discrete Cosine Transform of $x_0, x_1, \ldots, x_{N-1}$. DCT transform is an orthogonal and real transform. The inverse of the DCT is formulated as follows:

$$x_{k1} = \frac{2}{N}\left[\frac{1}{2}X_0 + \sum_{n_1=1}^{\mathrm{N}_1-1} X_{n1} \cos\left[\frac{\pi}{\mathrm{N}_1} n_1 \left(\mathrm{k}_1 + \frac{1}{2}\right)\right]\right] \quad \textbf{3-3}$$

The relation given above is valid for one dimensional case. When it comes to 2D, the equation is changed accordingly. 2D form of DCT equation can be written as:

$$X_{k1,k2} = F\left(x_{k1,k2}\right) \quad \textbf{3-4}$$

$$X_{k1,k2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n1,n2} \cos\left[\frac{\pi}{N_1}\left(n_1 + \frac{1}{2}\right)k_1\right]\left[\frac{\pi}{N_2}\left(n_2 + \frac{1}{2}\right)k_2\right] \quad \textbf{3-5}$$

Thanks to the *separability* property of the 2D DCT, it can also be accomplished by first applying one dimensional DCT to the rows and then to the columns or vice versa.

Notice that the DCT matrix is not dependent on the input, hence for given length of matrix, the DCT matrix is fixed and it can be used for any other input matrix with the same dimensions.

If we try to derive the inverse 2D DCT, we need to write down the properties of DCT one by one:

Suppose that *T* is the 2D DCT transform matrix and *X* is the 2D DCT transform of *x*. First, we know that DCT of x is:

$$X = T.x \quad \textbf{3-6}$$

From the *orthogonality* of DCT, we can write:

$$T^{-1} = T^T \quad \textbf{3-7}$$

and

$$TT^{-1} = 1 \text{ or equivalently } TT^T = 1$$

Since the DCT is real, we can also write:

$$T = T^* \quad \textbf{3-8}$$

From equation 3-3, for the IDCT, we can write (30):

$$x = (T)^{-1}.X = T^{T}.X \quad \textbf{3-9}$$

This equation can be expanded for 2D transform such that:

i. For forward 2D DCT:

$$X = TxT^{T} \quad \textbf{3-10}$$

ii. For reverse 2D DCT (IDCT):

$$x = T^{T}X\,T \quad \textbf{3-11}$$

where, *X* and *x* are two dimensional matrices of the same size and *X* is the 2D DCT of *x*.

As stated above, the DCT transform matrix of any input of the same size is identical. This fact can be used as an advantage to make the DCT easier to implement. Such that, suppose a picture is desired to be transformed to the DCT domain with 2D DCT. Then, the image needs to be divided into blocks with the same size –preferably 4x4 or 8x8 – and each block is processed independently. However, there is no need to derive the DCT matrix for every block since it is the same for any of them thanks to the fixed block image size and *separability* of input and the transform matrix of the DCT.

In general, when the classical image compression methods are concerned, e.g. JPEG, blocks of size 4x4 or 8x8 are preferred. So, if one needs to use DCT, it is enough to use the DCT formula to derive DCT transformation matrix. The situation is the same for the two dimensional case. In fact, In Matlab®, there is a special function called *dctmtx* which outputs the 2D DCT matrix for a give input matrix size.

These matrices are derived using the following constants extracted from the 2D DCT and IDCT formula.

i. For the forward DCT, the transformation matrix *T* is:

$$T = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \cdots & \cos\frac{\pi(2n-1)}{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \cos\frac{\pi(n-1)}{2n} & \cos\frac{3\pi(n-1)}{2n} & \cdots & \cos\frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix} \quad \textbf{3-12}$$

ii. For the inverse DCT, the transformation matrix is:

$$T^{-1}=T^{T}= \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos\frac{\pi}{2n} & \cdots & \cos\frac{\pi(n-1)}{2n} \\ \frac{1}{\sqrt{2}} & \cos\frac{3\pi}{2n} & \cdots & \cos\frac{3\pi(n-1)}{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{\sqrt{2}} & \cos\frac{\pi(2n-1)}{2n} & \cdots & \cos\frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix} \quad \textbf{3-13}$$

In practice 4x4 and 8x8 types of 2D DCT are the most popular ones and their corresponding DCT matrices are:

i. For 4x4 :

$$\begin{bmatrix} 0.5000 & 0.5000 & 0.5000 & 0.5000 \\ 0.6533 & 0.2706 & -0.2706 & -0.6533 \\ 0.5000 & 0.5000 & 0.5000 & 0.5000 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{bmatrix}$$

ii. For 8x8 :

$$\begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}$$

There is usually a curiosity about why DCT is invented and used instead of FFT (Fast Fourier Transform) in image processing. There are actually two answers for this question: First, DCT operates on real numbers whereas FFT uses complex exponentials in calculations. Second, DCT can approximate the sequences better with the same number of coefficients. This is a big advantage in compression since the actual signal is expressed nearly in original quality with reduced number of coefficients. Below, an example comparison of DCT and FFT is shown (31):

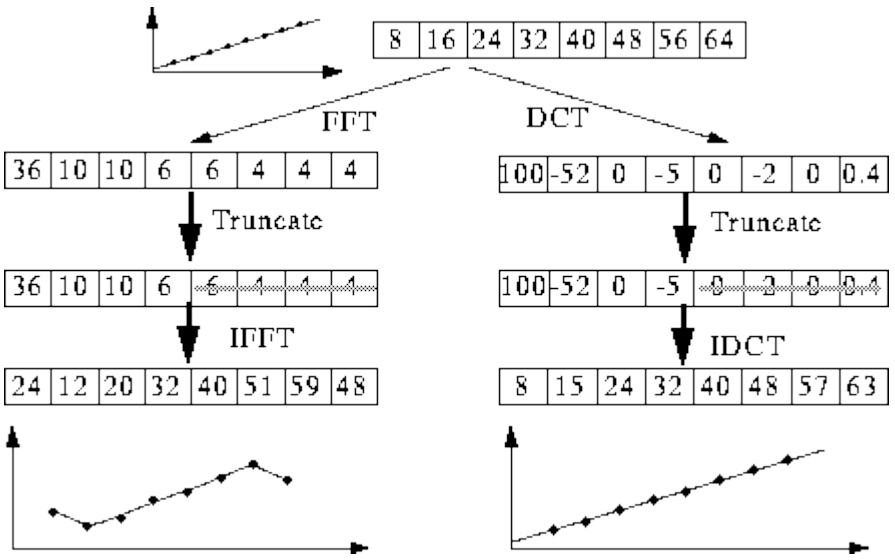


Figure 7 Comparison of DCT and FFT with reduced coefficients

3.3 Explanation of the Proposed Algorithm

As stated in the fusion literature, the best way of performing image fusion is to transfer the *edge* information to the output as much as possible while keeping the local means of the output image as a true combination of the input images. It is

known that the edges of an image lie in the high frequency components of the image. Following this fact, we may conclude that in order to implement a decent fusion technique, the frequency components of the input images should be decomposed and the high frequency components should be transferred to the output.

On the other hand, if we think of the fused image output, it is a synthetic output by all means. More clearly, in reality, such a picture or a video cannot be captured anyhow. This fact brings the necessity that the output image must not look artificial. This can be achieved by distributing the luminance information of the image in a homogenous way; meaning that there shall not be any absurd steps or discontinuities in the image. Discontinuities in an image may arise from blocking effect where the image is processed in blocks that are mutually independent from each other. One block has a totally independent output with respect to its neighbors; however in reality, no such situation exits. The reality behaves analog, so if it is digitized, the digitization must not be visible in order to make the output seem real. Even if discontinuities exist in an image, they must be minimized and be under the detail perception of HVS.

So, if above conditions are set, a proper image fusion algorithm can be achieved. High level image fusion techniques in fact succeed above mentioned bottlenecks by making use of the transform domain and decomposing the input images into frequency bands. However, it is most of the time not appropriate for real time implementation. So, there is an obvious need for an adequate algorithm which is suitable for real time implementation, at the same time has an appealing visual performance.

DCT is one of the best ways to handle above discussed purpose. In DCT theory, the DC value of a nxn block is its upper-left component, merely. The rest of the n^2-1 terms are the AC coefficients.

A satisfactory image fusion technique should blend the DC and AC coefficients from all outputs in such way that the local mean of the output is homogenous; at the same time all the high frequency terms are reflected to the output.

In the light of above assertions a new image fusion algorithm is proposed as follows:

Let X and Y are two images of the same size of mxn and let k denotes the k^{th} 4x4 block of X and Y. Further, assume that X^D and Y^D denotes the 2D DCT of images X and Y and i and j denote the index of block k of images X^D and Y^D.

Here, $0 \leq i \leq 3$ and $0 \leq j \leq 3$. The point $(i , j) = (0 , 0)$ denotes the DC coefficient of each block of transform images X^D and Y^D.

Assume that F denotes the fused image and F^D denotes the 2D DCT of it. Naturally, F is of size mxn and F^D is composed of k 4x4 blocks.

The method for calculating DC and AC values of the blocks is accomplished in the following way:

i. DC Coefficient:

$$F^D_{(k,0,0)} = average\left(X^D_{k,0,0}, Y^D_{k,0,0}\right) \qquad \textbf{3-14}$$

ii. AC Coefficient:

Let, $\|X^D_k\|$ and $\|Y^D_k\|$ be the L2 norm of k^{th} block AC coefficients of X and Y. Then:

$$F^D_{k,i,j} = \frac{\left(X^D_{k,i,j} \times \|X^D_k\|\right) + \left(Y^D_{k,i,j} \times \|Y^D_k\|\right)}{\|X^D_k\| + \|Y^D_k\|} \qquad \textbf{3-15}$$

Where L2 norm of a vector x is defined as:

$$\|x\| = \sqrt{\sum_{k=1}^{n} |x_k|^2} \qquad \textbf{3-16}$$

where

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \textbf{3-17}$$

The DC term is averaged in order not to lose the homogeneity across the whole picture. By doing so, the output is image is assured to contain low frequency constants at equal measures.

The AC coefficient calculation is accomplished based on weighted average technique so that the output image does not suffer from blocking effect. As a weighting constant, L2 norm is used. The reason why L2 norm is selected is that L2 norm represents the overall weight of the block. If a direct point to point weighting average technique was adopted, then there would be discontinuities on the edges of each block. By making use of L2 norm weighting, the image is smoothened as much as possible.

By using a DCT based technique in image fusion, the system is automatically made available for image compression and transmission. If desired, additional quantization and entropy coding can be applied to the output of the fusion step making it JPEG compatible.

3.4 Benchmarking

The developed image fusion algorithm needs to be tested with respect to the other image fusion algorithms from several ways to prove its efficiency. There are actually two main performance measure areas for benchmarking. The first one is the visual evaluation where the outputs of the fusion algorithms are evaluated visually. Secondly, several performance metrics are applied to the fused image outputs of all fusion methods and the results are given numerically. The first method is rather subjective while the second one depends only on the mathematical formulas. Below, the results of the two performance measures are analyzed in detail.

3.4.1 Visual Results

While developing the proposed image fusion algorithm, several image fusion methods are implemented and tested in software. In development process, test images which have different image characteristics are used. All of the implemented algorithms are tested among all test images. There are about 12 image sets used throughout this thesis. However, only the results of 3 image sets are given for visual comparison due to space drawback. For each image set, there are 20 different image fusion algorithm results shown. The order of fusion algorithm among all 3 inputs image set is the same and given below:

Table 1 Index for Fusion Results of Different Algorithms

Fusion Method	Corresponding Image Fusion Algorithm
FC-TF*	*False Color IF** by Toet & Franken* (32)
FC-TW*	*False Color IF by Toet & Walraven* (10)
FC-HZ*	*False Color IF by Huang & Zhang* (12)
FC-JR*	*False Color IF by Jang & Ra* (13)
FC-PA*	*Proposed False Color IF Algorithm*
DCT*	*Proposed DCT Based IF.*
PA*	*Pixel Averaging IF*
SMAX*	*Select Maximum IF*
SMIN*	*Select Minimum IF*
LP*	*Laplacian Pyramid IF*
FSD*	*FSD Pyramid IF*
RP*	*Ratio Pyramid IF*
CP*	*Contrast Pyramid IF*
GP*	*Gradient Pyramid IF*
SV-DWT*	*Shift Variant DWT IF*
SI-DWT*	*Shift Invariant DWT IF*
PCA*	*Principal Component Analysis IF*
BSV_DWT*	*Block -wise Shift Variant DWT IF*
DCT-J*	*Jinshan's IF*
DCT-Z*	*Zafar's IF*

** The full-form of the acronym is given in Appendix B.*

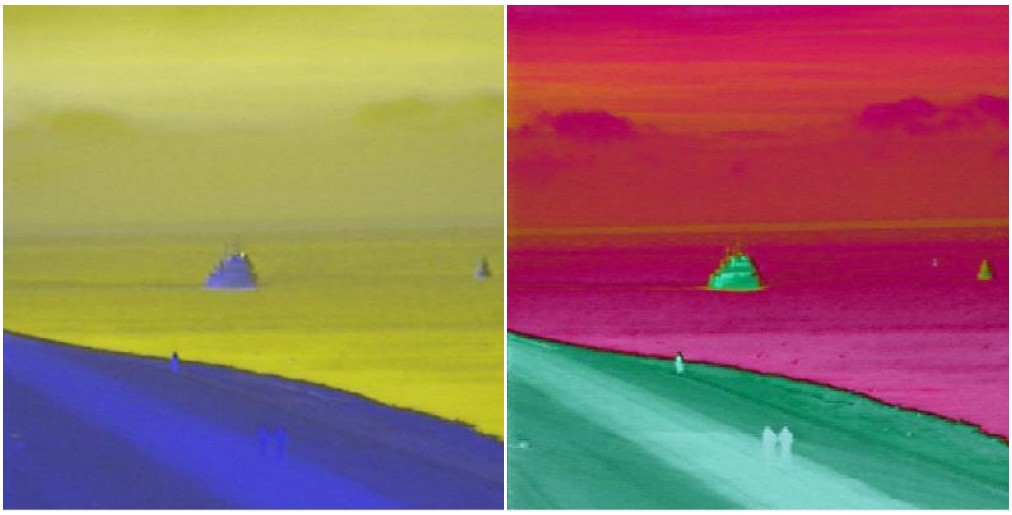
*** Image Fusion*

3.4.1.1 Results Acquired with the First Image Set

The first image set is a compilation of two images one being an electro-optic image captured from a sea coast with a couple of people and a ship; and the other one being the IR image of the same scene. The original images are displayed below while the first one being electro-optic and the second one being the IR image:

Original Picture – Electro-optic **Original Picture - IR**

Since the first 5 results are outputs of *color composite* fusion algorithms, they are not monochrome. Excluding *PCA Based IF*, the other 14 results are very similar to each other. They only differ in minor points. The proposed IF algorithm performs well in this image set.



FC-TF **FC-TW**

FC-HZ

FC-JR

FC-PA

DCT

PA

SMAX

SMIN

LP

FSD

RP

CP

GP

SV-DWT **SI-DWT**

PCA **BSV-DWT**

DCT-J **DCT-Z**

Figure 8 The First Image Set

3.4.1.2 Results Acquired with the Second Image Set

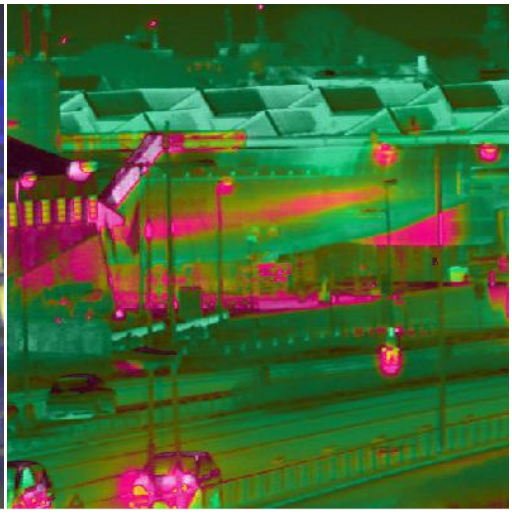
The second image set is a compilation of two images one being an electro-optic image captured from an intercity road with couple of cars and pedestrians with buildings by it; and the other one being the IR image of the same scene. The original images are displayed below while the first one being electro-optic and the second one being the IR image:

Original Picture – Electro-optic

Original Picture - IR

The related outputs are displayed below:

FC-TF



FC-TW

FC-HZ

FC-JR

FC-PA

DCT

PA

SMAX

SMIN

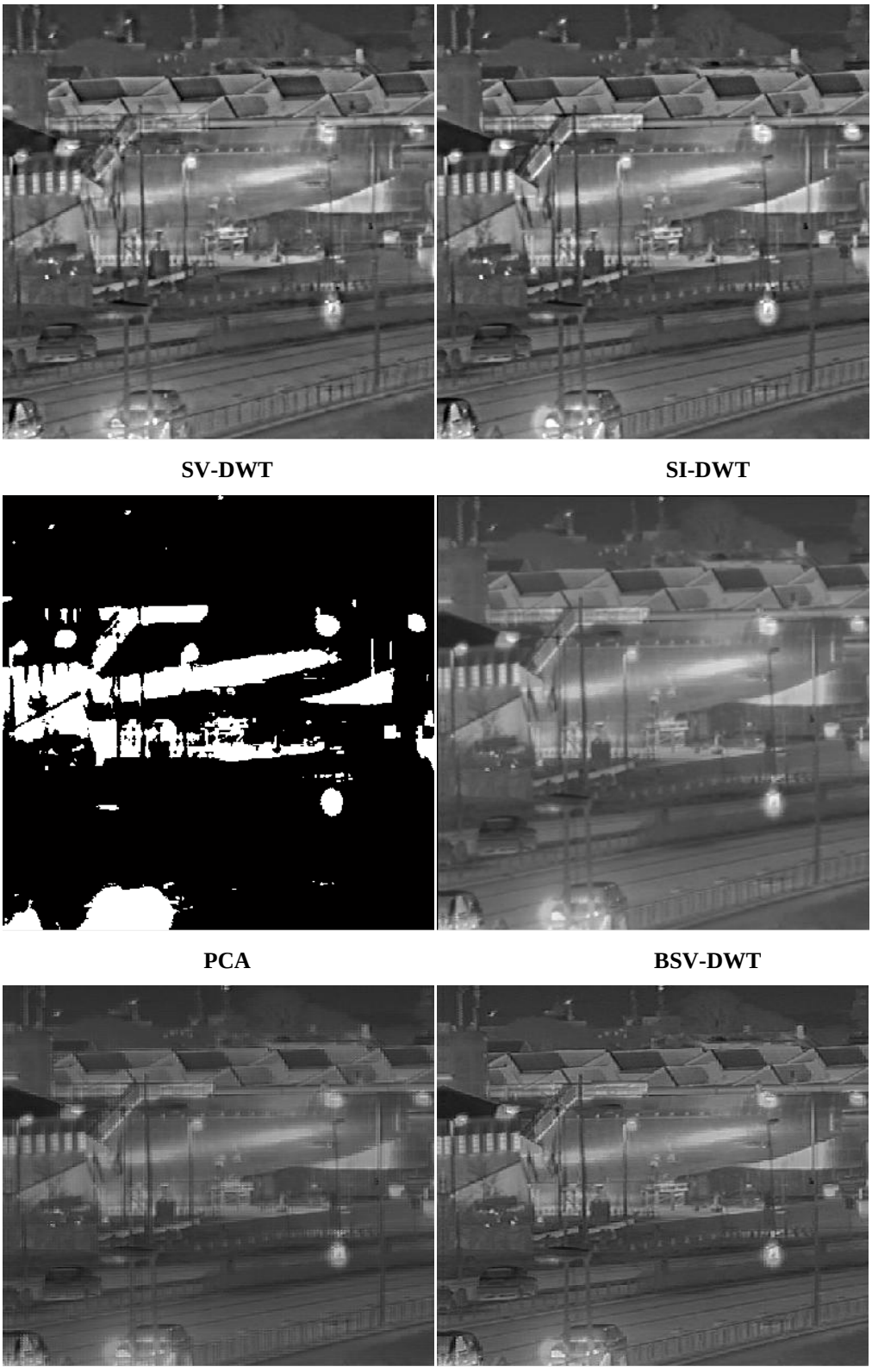
LP

FSD

RP

CP

GP



SV-DWT **SI-DWT**

PCA **BSV-DWT**

DCT-J **DCT-Z**

Figure 9 The Second Image Set

3.4.1.3 Results Acquired with the Third Image Set

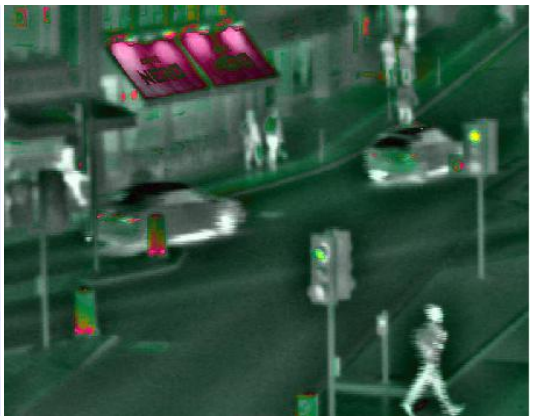
The third image set is a compilation of two images one being an electro-optic image captured from an intercity crossroads with traffic lights and a café in the corner and the other one being the IR image of the same scene. The original images are displayed below while the first one being electro-optic and the second one being the IR image:

Original Picture – Electro-optic

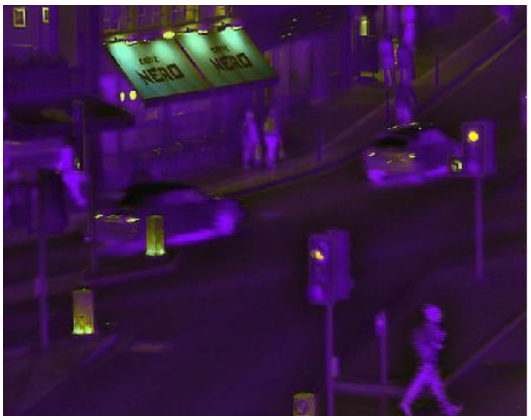
Original Picture - IR

The related outputs are displayed below:

FC-TF



FC-TW



FC-HZ

FC-JR

FC-PA

DCT

PA

SMAX

SMIN

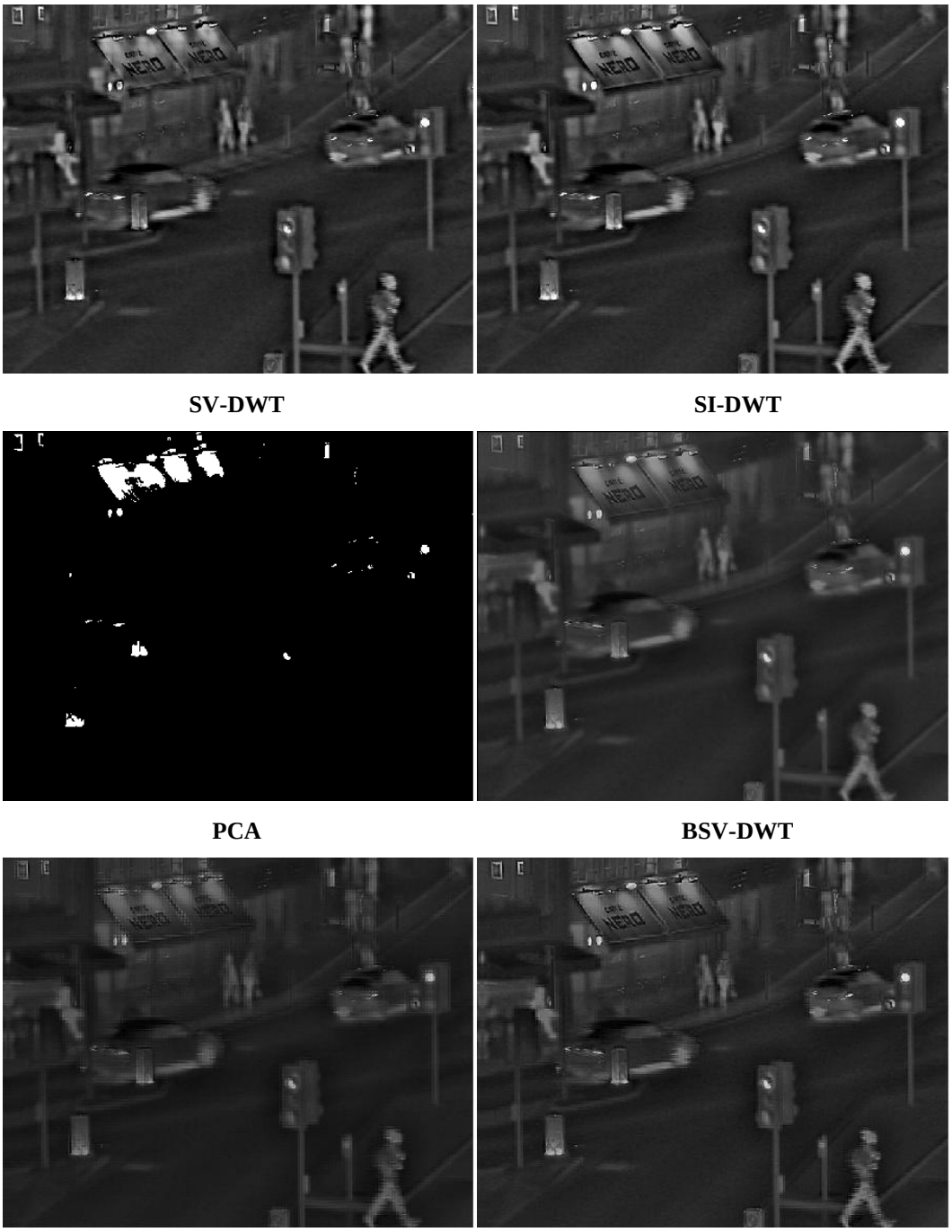
LP

FSD

RP

CP

GP



SV-DWT **SI-DWT**

PCA **BSV-DWT**

DCT-J **DCT-Z**

Figure 10 The Third Image Set

3.4.2 Performance Metrics

In order to analyze and evaluate the proposed fusion scheme, objective measure techniques should be introduced and used. Below; a total of 6 fusion evaluation metrics are introduced and explained thoroughly in theory.

3.4.2.1 A Review of Performance Metrics

In image fusion, performance evaluation is yet another subject that needs to be further analyzed and put on solid grounds for objective performance analysis. Today, mostly, quality assessment of fused images is carried out by human visual inspection (32). Objective performance assessment is very complicated since;

i. The requirements for each application of fusion differ significantly.
ii. There is no ground-truth for relative comparison.

Due to lack of ideal reference picture, engineers try to achieve a novel benchmarking method without making use of a ground truth. Several algorithms have been developed in last decades in order to meet the rising need of proper quality assessment tools. Below, the leading performance metric algorithms are discussed:

3.4.2.1.1 Standard Deviation

Standard deviation is the easiest and simplest image quality evaluation procedure. Then main discrepancy of SD (Standard Deviation) is that it does not evaluate the result with respect to the input images, instead it only evaluates the detail (in fact contrast) level in the output image. The formulation for SD is as follows (33):

$$SD = \sqrt{\sum_{i=0}^{L} \left(i - \bar{i}\right)^2 h_{I_f}(i)}$$

Here,

$$\bar{i} = \sum_{i=0}^{L} i\, h_{I_f}$$

Here, $h_{I_f}(i)$ is the normalized histogram of the fused image $I_f(x, y)$ and L is the number of frequency bins in the histogram.

3.4.2.1.2 Entropy

In communication theory, entropy is defined as the measure of the uncertainty of an outcome. In fact, entropy is a measure of PDF (Probability Density Function), the higher the values is the higher the randomness of a random variable. In image processing, this fact can be utilized as higher the entropy, higher the information content stored in an image. This way of thinking helps fused image outputs are evaluated in return. The higher value means more information is stored in one image than the other.

Entropy is defined as (33):

$$He = -\sum_{i=0}^{L} h_{I_f}(i) log_2\, h_{I_f}(i) \qquad \textbf{3-18}$$

3.4.2.1.3 Cross Entropy

Cross entropy is a measure of correlation between two images. In image fusion it is used for evaluating similarities the input images and the output image. Cross entropy is a mutual operator which means; it accepts two inputs and returns one output. In image fusion, the cross entropy between the inputs and the output is calculated for each IF methodology and the one gives the higher result means better fusion. In order

to calculate CE (Cross Entropy), we need to calculate CE mutually between each input and the output, and then we need to take the arithmetic mean to reach the final result. The formulation is as follows (33):

$$CE\left(I_1, I_2; I_f\right) = \frac{CE\left(I_1; I_f\right) + CE\left(I_2; I_f\right)}{2} \quad \textbf{3-19}$$

Here,

$$CE\left(I_1; I_f\right) = \sum_{i=0}^{L} h_{I_1}(i) log \frac{h_{I_1}(i)}{h_{I_f}(i)} \quad \textbf{3-20}$$

$$CE\left(I_2; I_f\right) = \sum_{i=0}^{L} h_{I_2}(i) log \frac{h_{I_2}(i)}{h_{I_f}(i)} \quad \textbf{3-21}$$

3.4.2.1.4 Mutual Information

Mutual Information (MI) is a widely used technique in statistics which gives information about the dependence between two random variables. In image fusion, MI is interpreted as if MI is high, then it means that the amount of information in one input image is transferred to the output image is that much high.

The calculation of MI is similar to that of CE, and it is written as follows (33):

$$MI = MI_{I_1 I_f} + MI_{I_2 I_f} \quad \textbf{3-22}$$

Here,

$$MI_{I_1 I_f} = \sum_{i=1}^{M} \sum_{j=1}^{N} h_{I_1 I_f}(i,j) log_2 \left(\frac{h_{I_1 I_f}(i,j)}{h_{I_1}(i,j) h_{I_f}(i,j)} \right) \quad \textbf{3-23}$$

$$MI_{I_2I_f} = \sum_{i=1}^{M}\sum_{j=1}^{N} h_{I_2I_f}(i,j) log_2\left(\frac{h_{I_2I_f}(i,j)}{h_{I_2}(i,j)h_{I_f}(i,j)}\right) \quad \textbf{3-24}$$

3.4.2.1.5 Universal Image Quality Index

UIQI (Universal Image Quality Index) is by far one of the most accepted objective evaluation criteria developed for quality evaluation of image fusion. It is first proposed by Alan C. Bovik (34). It is further developed and expanded to cover evaluation with no ground truth case by Gemma Piella and Henk Heijmans (35). This paper brings also a saliency measure to the proposed algorithm. Later, the algorithm is further developed by Nedeljko Cvejic et al (36). It additionally calculates locally how much of the important data is transferred to the fused image. The latter algorithm also represents a major improvement in the computational complexity.

Below, the state of the art UIQI is explained theoretically:

Let $I_1 = \{I_{1_i} | i = 1,2,\cdots,N\}$ and $I_2 = \{I_{2_i} | i = 1,2,\cdots,N\}$ be the original and test images respectively. UIQI is defined as:

$$Q = \frac{4\sigma_{xy}\bar{I_1}\bar{I_2}}{\left({I_1}^2 + {I_2}^2\right)\left[\bar{I_1}^2 + \bar{I_2}^2\right]} \quad \textbf{3-25}$$

Here,

$$\bar{I_1} = \frac{1}{N}\sum_{i=1}^{N} I_{1_i} \quad , \quad \bar{I_2} = \frac{1}{N}\sum_{i=1}^{N} I_{2_i} \quad \textbf{3-26}$$

$$\sigma_{I_1}^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(I_{1_i} - \bar{I_1}\right)^2 \quad , \quad \sigma_{I_2}^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(I_{2_i} - \bar{I_2}\right)^2 \quad \textbf{3-27}$$

$$\sigma_{I_1 I_2} = \frac{1}{N-1}\sum_{i=1}^{N}\left(I_{1_i} - \bar{I_1}\right)\left(I_{2_i} - \bar{I_2}\right) \quad \textbf{3-28}$$

The dynamic range of Q is [-1, 1]. In fact, UIQI models three different image distortions independently, those of which are loss of correlation, luminance distortion and contrast distortion. The UIQI equation can be separated into three each representing one of the distortion factors.

$$Q = \frac{\sigma_{I_1 I_2}}{\sigma_{I_1}\sigma_{I_2}} \frac{\bar{2}\bar{I_1}\bar{I_2}}{\bar{I_1}^2 + \bar{I_2}^2} \frac{2\sigma_{I_1}\sigma_{I_2}}{\sigma_{I_1}^2 + \sigma_{I_2}^2} \quad \textbf{3-29}$$

The first fraction represents correlation coefficient between I_1 and I_2. The second fraction shows how close the mean luminance is between I_1 and I_2. Finally, the third fraction shows how close the contrast is between I_1 and I_2.

However, it is not safe to measure all those three components across the whole image since the images are most of the time not stationary. Instead, small regions of the images are to be considered for evaluation for a realistic result. Here, first the local measure is calculated for all the segments in the image, after that all the outputs are summed and averaged to find the quality of the whole image. The equation is as follows:

$$Q\ (I_1, I_2) = \frac{1}{|W|}\sum_{w \in W} Q_0(a, b|w) \quad \textbf{3-30}$$

Here, W is the family of all windows and $|W|$ is the cardinality of W.

In order to apply this equation to image fusion Gemma Piella and Henk Heijmans (35) added salient information to the metric.

$$Q_p\,(I_1, I_2, I_F) = \sum_{w \in W} c(w)\big(\lambda Q(I_1, I_F|w) + (1-\lambda)Q(I_2, I_F|w)\big) \qquad \textbf{3-31}$$

Here, *F* is the fused image, *c(w)* is the saliency of a window and λ is defined as:

$$\lambda = \frac{s(I_1|w)}{s(I_1|w) + s(I_2|w)} \qquad \textbf{3-32}$$

Here, $s(I_1|w)$ denotes the saliency of image I_1 in window *w*. Gemma Piella and Henk Heijmans added one last parameter to the equation which adds the edge information of the input images with a relative parameter α.

$$Q_E(I_1, I_2, I_F) = Q_p(I_1, I_2, I_F)^{1-\alpha} Q_p(I_1{}', I_2{}', I_F{}')^{\alpha} \qquad \textbf{3-33}$$

Gemma Piella and Henk Heijmans ended their assertions to the UIQI idea with the above equation. Lastly, Nedeljko Cvejic et al (36) modified the UIQI as follows:

$$Q_b(I_1, I_2, I_F) = \sum_{w \in W} sim(I_1, I_2, I_F|w).\big(Q(I_1, I_F|w) - Q(I_2, I_F|w)\big) + Q(I_2, I_F|w) \qquad \textbf{3-34}$$

Here, $sim(I_1, I_2, I_F|w)$ is defined as:

$$sim(I_1, I_2, I_F|w) = \begin{cases} 0 & if \quad \dfrac{\sigma_{xf}}{\sigma_{I_1} + \sigma_{I_2}} < 0 \\ \dfrac{\sigma_{I_1}}{\sigma_{I_1} + \sigma_{I_2}} & if \quad 0 \le \dfrac{\sigma_{xf}}{\sigma_{I_1} + \sigma_{I_2}} \le 1 \\ 1 & if \quad \dfrac{\sigma_{xf}}{\sigma_{I_1} + \sigma_{I_2}} > 1 \end{cases} \qquad \textbf{3-35}$$

Here,

$$\sigma_{uv} = \frac{1}{N-1}\sum_{i=1}^{N}(u_i - \bar{u})(v_i - \bar{v}) \qquad \textbf{3-36}$$

If UIQI is exspressed in this form, then there is no need for an *edge image* for evaluation hence allowing a decrease in computational complexity. Additionally, Gemma Piella and Henk Heijmans used UIQI with a fixed block size of 8x8, whereas Nedeljko Cvejic et al evaluated each image with 4x4, 8x8 and 16x16 window sizes separately.

3.4.2.1.6 Performance Measure by C. S. Xydeas & V. Petrović

Petrović et al proposed a different perceptually meaningful performance measure depending on edge preservation (37). Petrović based his assumption on the following facts:

A decent fusion method should satisfy below criterias:

i. Maximize the contribution of each input image to the output image in terms of valuable information.
ii. Control the efficiency when different types of inputs are applied and also monitor how well the contents of the inputs are transferred to the output.

The formal definition of the algorithm can be defined as:

Let I_1 and I_2 are input images and I_F is the fused image and *p(m,n)* denote the pixel located in position *(m,n).* And, let *g(m,n)* denote Sobel edge strength and *α(m,n)* denote the Sobel orientation of an *M*x*N* sized image. Put it this way:

$$g_{I_1}(m,n) = \sqrt{S_{I_1}^x(m,n)^2 + S_{I_1}^y(m,n)^2} \qquad \textbf{3-37}$$

and

$$\alpha_{I_1} = tan^{-1}\left(\frac{S_{I_1}^y(m,n)}{S_{I_1}^x(m,n)}\right) \qquad \textbf{3-38}$$

In this equation, $S_{I_1}^{x}$ *(n,m)* and $S_{I_1}^{y}$ *(n,m)* are the Sobel gradients of image I_1's intensity in both horizontal and vertical directions.

The relative strength $G^{I_1 I_F}$*(n,m)* and orientation $A^{I_1 I_F}$*(n,m)* of input image I_1 with respect to I_F are formed as:

$$G^{I_1 I_F} = \begin{cases} \dfrac{g_{I_F}(m,n)}{g_{I_1}(m,n)}; & if\ g_{I_1}(m,n) > g_{I_F}(m,n) \\ \dfrac{g_{I_1}(m,n)}{g_{I_F}(m,n)}; & otherwise \end{cases} \quad \textbf{3-39}$$

$$A^{I_1 I_F}(m,n) = \frac{\left| \left| \alpha_{I_1}(m,n) - \alpha_{I_F}(m,n) \right| - \frac{\pi}{2} \right|}{\frac{\pi}{2}} \quad \textbf{3-40}$$

Using above equations one can derive edge strength and orientation preservation values:

$$Q_g^{I_1 I_F}(m,n) = \frac{\Gamma_g}{1 + e^{\kappa_g (G^{I_1 I_F}(m,n) - \sigma_g)}} \quad \textbf{3-41}$$

$$Q_\alpha^{I_1 I_F}(m,n) = \frac{\Gamma_\alpha}{1 + e^{\kappa_\alpha (G^{I_1 I_F}(m,n) - \sigma_\alpha)}} \quad \textbf{3-42}$$

$Q_g^{I_1 I_F}(m,n)$ and $Q_\alpha^{I_1 I_F}(m,n)$ signify perceptual loss of information in I_F. "The constants Γ_g, κ_g, σ_g and Γ_α, κ_α, σ_α determine the exact shape of the sigmoid functions used to form the edge strength and orientation preservation values. (37)" The relative edge information preservation values are then defined as:

$$Q^{I_1 I_F}(m,n) = Q_g^{I_1 I_F}(m,n) Q_\alpha^{I_1 I_F}(m,n) \quad \textbf{3-43}$$

Then the normalized performance metric is defined as:

$$Q^{(I_1 I_2 | I_F)} = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} Q^{I_1 I_F}(m,n) w^{I_1}(m,n) + Q^{I_2 I_F}(m,n) w^{I_2}(m,n)}{\sum_{i=1}^{M} \sum_{j=1}^{N} (w^{I_1}(i,j) + w^{I_2}(i,j))} \quad \textbf{3-44}$$

Here, $w^{I_1}(m,n) = \left(g_{I_1}(m,n)\right)^L$ and $w^{I_2}(m,n) = \left(g_{I_2}(m,n)\right)^L$ where L is a constant. Also note that $0 < Q^{(I_1 I_2 | I_F)} < 1$, which means a value of 1 means exact match on the other hand a value of 0 means a loose match between the input images I_1 and I_2 and the output image I_F.

3.4.3 Empirical Results

Below, a table which summarizes the results of the entire objective evaluation criterion applied on the first image set with all 20 different image fusion methods is given:

Table 2 Performance Measure of Implemented Algorithms

	SD	Entropy	CE	MI	UIQI	Petrović
FC-TF*	N/A	N/A	N/A	N/A	N/A	N/A
FC-TFW*	N/A	N/A	N/A	N/A	N/A	N/A
FC-HZ*	N/A	N/A	N/A	N/A	N/A	N/A
FC-JR*	N/A	N/A	N/A	N/A	N/A	N/A
FC-PA*	N/A	N/A	N/A	N/A	N/A	N/A
DCT-P*	2.3111	6.1031	-1.0542	3.3987	0.9315	0.4204
PA*	2.0018	6.1940	-3.5271	3.6587	0.9271	0.2268
SMAX*	1.6522	6.0312	-1.8363	7.0993	0.8964	0.0241
SMIN*	2.8825	6.0330	-5.6179	7.3136	0.8131	0.4274
LP*	2.3648	6.2512	-1.7012	3.9285	0.9388	0.5477
FSD*	2.4016	6.1033	-1.7576	3.2779	0.9309	0.1675
RP*	1.9515	5.7253	-3.6763	2.9649	0.8594	0.1567
CP*	2.7544	6.0584	-2.0131	2.9305	0.8189	0.1725
GP*	2.4950	6.1065	-1.8166	3.3101	0.9309	0.1707
SV-DWT*	2.9505	6.1318	-1.9913	3.0322	0.9161	0.3041
SI-DWT*	2.8507	6.0806	-0.9265	3.1412	0.9148	0.3288
PCA*	0.8163	0.8404	-10.9834	1.5925	0.1389	0.0544
BSV-DWT*	2.5511	6.3103	-3.8597	2.9783	0.9136	0.2455
DCT-J*	2.5050	5.8358	-2.2633	2.6061	0.5377	0.2476
DCT-Z*	2.8310	6.0816	-0.8355	2.5498	0.5694	0.2527

*** Acronym for Table 1 Index for Fusion Results of Different Algorithms**

3.4.4 Overall Evaluation

3.4.4.1 Subjective Quality Evaluation

First of all, the first 5 output images are false color images and they possess rather convincing results with respect to others. However, the main concern is the

monochromatic results. So, a subjective assessment should be carried out between the remaining 15 monochromatic results.

In image set 1; DCT-P algorithm and pyramid based algorithms gives appealing results. However, wavelet based algorithms have non-realistic results since the thick line which connects the shore and the sea is too visible. The reason is that the wavelet based algorithms work in gradient basis which emphasizes the edges on the image.

In image set 2; the wavelet based techniques deliver the best results. The reason is the wavelet based algorithms enhance the edge information in the input images and this image sets is very rich in terms of high frequency. DCT-P algorithm gives comparably good results also since this algorithm relies on the fact that the details in the input images are transferred to the output as much as possible. Pyramid based approaches also have appealing results however they are relatively weak in terms of detail; but still, the integrity of the results are not synthetic.

In image set 3; wavelet based techniques performs well together with DCT-P algorithm and DCT-Z. Here, the IR image is a little bit distorted with zigzag shapes. This, in fact, increases the amount of high frequency components. So, it is expected that the algorithms based on high frequency extraction performs better than the other algorithms. This explains the success of wavelet and DCT based algorithms in this case.

3.4.4.1 Objective Quality Evaluation

The first 5 algorithms are *composite color fusion* techniques. Therefore, the mentioned evaluation metrics are not applicable. For the, rest of the algorithms, the discussed metrics are applicable and related results are given below:

In *standard deviation* metric; it is observed that wavelet based methods performs better with respect to others. However, the difference is not very remarkable since they differ only 3% with the next best output which is DCT-Z. The wavelet based method outputs are sharper. This increases contrast and as well as standard deviation. PCA based fusion is the worst in this case because the visual contrast is very low with respect to the other algorithms.

In *entropy* metric; all of the outputs have nearly equal values except the PCA based fusion. The reason why PCA based fusion gives the worst output is, it has a limited image histogram which decreases the probability of randomness.

In *cross entropy* metric; the results are very similar to that of the entropy metric since it uses the same equation with the addition of the *entropy* of the fused image.

In *mutual information* metric; the results are similar to that of *cross entropy* because both metrics represent the quantity of common information between the input and output images. But, SMIN algorithm gives a very high output in this metric. This can be explained the mean of the images. In other words, the input images are relatively dark images and since the SMIN algorithm returns the *min* value of the input images, the output is a nearly true reflection of the input images. That is why SMIN algorithm performs well in this case.

UIQI metric; is a true illustration of non-reference image quality which represents how well the important information of input images are represented by the fused image. So, the output of this metric can be used as a solid performance measure. The worst output in *UIQI* metric is again the PCA based IF. This is reasonable because of the above stated reasoning. Tang's and Zafar's method have relatively low values. Actually, the visual results do not have a distinct difference with respect to others, however; they are still worse than the other multiscale algorithms in subjective evaluation. In parallel with the subjective assessment, multiscale based approaches

perform best in *UIQI*. LP has the highest score, DCT-P has the second and FSD has the third highest result.

Petrovic's metric is based on the measure of how exactly the edge information of input images are transferred to the fused image in the fusion process. Based on this focus, it is expected that the fused image which has the most edge information, score highest in *Petrovic's* metric. Here, LP has again the highest score. This is expected, since LP has scored well in other metrics and subjective assessment of LP is also relatively good. SMIN has the second highest score which can also be explained with the above reasoning. DCT-P has the third highest score which is reasonable since the edge content of the DCT-P is noticeable also in subjective evaluations. Wavelet based techniques have the next highest scores which is also usual because the edge contents of SI-DWT and SV-DWT are clearly visible in subjective assessment.

As a final comparison; the process time for all of the algorithms are calculated. Below the elapsed processing time for each algorithm is given:

Table 3 Timing Chart for Implemented Algorithms*

Algorithm	Elapsed Time (sec)	Algorithm	Elapsed Time (sec)
FC-TF	0.061935	**FSD**	0.325056
FC-TFW	0.886511	**RP**	0.372840
FC-HZ	0.911738	**CP**	0.372860
FC-JR	1.067808	**GP**	1.019098
FC-PA	0.116901	**SV-DWT**	0.570385
DCT-P	1.535491	**SI-DWT**	2.203851
PA	0.041257	**PCA**	0.039491
SMAX	0.029444	**BSV_DWT**	1.382988
SMIN	0.043923	**DCT-J**	1.490140
LP	0,596087	**DCT-Z**	1.615464

** Tests are run on an AMD Athlon X2 5200+ computer with 2GB 800MHz RAM.*

As seen in the table above, the proposed algorithm (DCT-P) has the 3rd longest process time among all algorithms after SI-DWT and DCT-Z. In fact, this is not an unexpected result since the proposed algorithm works block-wise and in each block

there are 3 DCTs are calculated and L2 norm is calculated for each input images. However, this long process time is not a problem for our application since the hardware implementation is independent of the process time.

CHAPTER 4

IMAGE FUSION HARDWARE IMPLEMENTATION

4.1 Introduction

In today's technology, the problems are handled either by software or hardware solutions either of which has different pros and cons separately. In fact, in known perception, hardware serves as a platform for software and this is the main reason for its existence. However, with the recent developments in technology during the last decades this perception has changed letting the hardware systems serve a full solution to most of the electronic systems.

In software methodology; there are set of instructions that prompts the processor a specific function to be done in a sequential perspective. Today's processors have been increased in operating frequency and developed rapidly in architecture. But, by all means, processors work in a sequential manner and they accomplish the awaiting jobs in order. This situation prohibits a possible parallel operation which can work much faster.

The need of embedded hardware solutions have emerged due to the fact described above. It is desired to have a device which performs all the requested instructions at the same time in a parallel fashion not letting any instruction disturb or interrupt any other one.

Following the fact depicted above, the solutions for electronic systems are twofold; first one being the software and the second one is hardware.

In signal processing, software solutions are handled by DSP cards. DSP cards are used together with a computer or standalone. Standalone DSP cards still need data and control ports in order to control the data flowing through and the applications running on the board. These data and interface control units are accomplished by PLDs (Programmable Logic Device) such as FPGAs or CPLDs. On the other hand, computer backplane DSP boards do not need to deal with custom interfaces; instead, they make use of computer digital data busses such as PCI, PCMCIA and VME etc. In this case, DSP board only makes the calculations and returns the result back to the computer so that the data can be transferred to an another device with a specific protocol. Since DSPs are powerful in terms of floating point calculations, they can operate signal processing applications with a high level of accuracy. Moreover, there is no need to deal with the hardware challenges such as timing and capacity overflow since DSP boards work in the software level which means, the desired application is coded in a software language such as systemC, and after the compile and build stages the program is run on the computer where the DSP board is installed.

The situation is a little bit different in the hardware solution case. Hardware solutions are usually addressed by FPGAs or CPLDs depending on the complexity of the design. An FGPA board makes no use of a software language such as C, or Assembly. Instead, FPGAs use one of the hardware description languages to create a meaningful hardware design from millions of gates it possesses. This language is typically VHDL (VHSIC Hardware Description Language) in industry and Verilog in academics. Not like the DSP programming, HDL (Hardware Description Language) language is written in a very basic form. Every function is implemented only by using gates. This can harden even the simplest designs a couple of times. Especially in signal processing, the functions used are rather complex which are hard to implement with logical gates.

However, FPGAs have one big advantage against DSPs and that is the parallel processing ability. One part of the FPGA can carry out the signal processing tasks where one other part can deal with a complete different scenario. This makes FPGAs more flexible than DSPs. This fact also encourages embedded designs where only single chip handles all the signal processing and control algorithms whereas DSPs are designed to process a single thread at a time.
In the light of all above explained reasoning, FPGAs seem to be more compact and flexible covering broad range of applications. Despite the deficiency against DSPs in terms of clock speed and floating point performance, FPGAs are state of the art design technology in the newest signal processing applications.

FPGA technology has been developed enormously during recent decades. Clock rates have increased, capacities have expanded, power dissipation reduced, more IPs made available and the global "know how" has reached to a satisfying state encouraging engineers to start designs based on FPGA architecture.

If we have a close look to newest signal processing applications, it is easy to say that nearly all of the research and development subjects are further investigated in FPGA architecture allowing decent performance and capacity improvements.

In this thesis, a hardware solution to the proposed image fusion methodology is investigated. The solution is implemented on a Stratix III series FPGA which is a product of the World's second biggest FPGA vendor Altera.

4.2 Setup Structure & Principles of Operation

The reason why FGPAs are preferred in recent signal processing applications is they perform much better in real time applications. So, if a real time application is desired to work on a FPGA board, a setup with real time input and output media needed to be constructed. In our case, there is one input carrying two fusion-ready video inputs, one FPGA board and one display device screening the output.

Input device is actually a computer with two DVI (Digital Visual Interface) outputs. The first output is used for desktop screening, while the second one is the input to the FPGA board. The FPGA board is called *Video Processing Board* which is further explained in APPENDIX C.

The input video is a 1024x768 @60Hz digital video. The upper 512 lines of the video frame is occupied with two side by side 512x512 video sub-frames, the first one being the electro-optic and second one is the IR video. Two 512x512 video sub-frames occupies the entire 1024 pixels horizontally, however, there is a 256 (768-512) lines of gap with no valuable data. This part of the frame is useless and it is colored with black.

The input video frame structure can be drawn like:

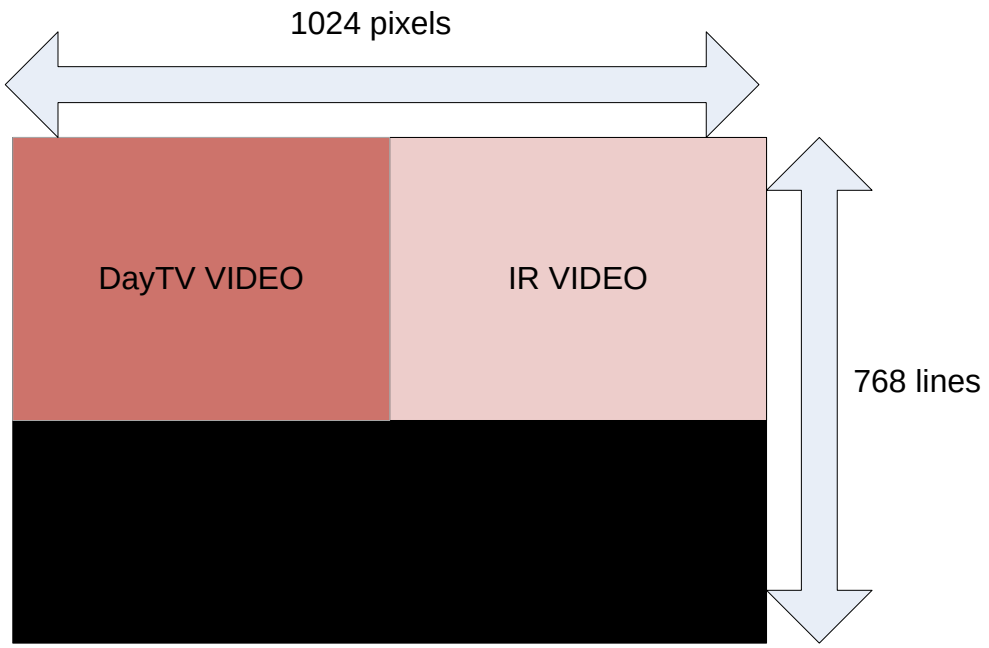


Figure 11 Input Video Structure

Here, it is observed that there is only one input to the system physically. Since this thesis is mostly a theoretical framework, two separate inputs are not connected.

Instead, two video sources are sent through one physical layer. Theoretically, this way of implementation has no difference with doing it in other way around. Accepting two inputs from two different physical devices is only practical in application and if desired, the implemented topology can easily be migrated to two separate inputs.

The output stage is rather simple than the input stage. The processed input video streams are fused inside the FPGA and displayed at the output with the video resolution of 1024x768 @ 60 Hz. The size of the output video is the same with that of the separate input video sub-frames which is 512x512 pixels.

4.3 Hardware Implementation of the Proposed Algorithm

4.3.1 Input Video Acquisition

The input video is grabbed from a DVI input channel through a decoder interface on the board. The video is grabbed along with its clock, sync and data signals.

In a 1024x768 @60Hz video structure, there are 768 active lines composed of 1024 pixels in one video frame. (38) One video frame is transferred and displayed in approximately 16.7 msec. This forces the algorithm to be completed in less than 16.7 msec. In fact, this is the main obstacle in many pyramid based fusion methods. Many of them suffer the time budget at a given pyramid depth.

For sake of simplicity, the input video is transferred to another domain which is created within the FPGA. In many applications, it is desirable to transfer all the inputs to a common inner domain. This method enables the designer to work in one single domain regardless of how many different input domains are applied in. In order to succeed this goal, the input video shall be stored somewhere in or out of the FPGA and reread with internal timings from the corresponding memories. This can be done in two ways; with frame buffers or with line buffers. The former one stores

the whole input video in an external frame buffer (e.g. any type of RAM with sufficient size) and the FPGA reads the frames from the RAM with its own synchronization and clock signals. This topology is easier in understanding; however, very much costly both in terms of time and budget. Because of this fact, this method is not preferred in many applications. The latter methodology uses sufficient number of line buffers to transfer the input video domain to that of the FPGA. In this approach, an optimal number of three line buffers are used for domain crossing. Since line buffers are far smaller than the frame buffers, they can be implemented within the FPGA. This methodology works in a way that input video lines are written to the line buffers in order by using input sync and clock signals and the filled line buffers are again read with the internally generated clock and sync signals. Once this step is accomplished, the internal FPGA design is immune against outer distortions. The above explained procedure can be explicitly drawn like:

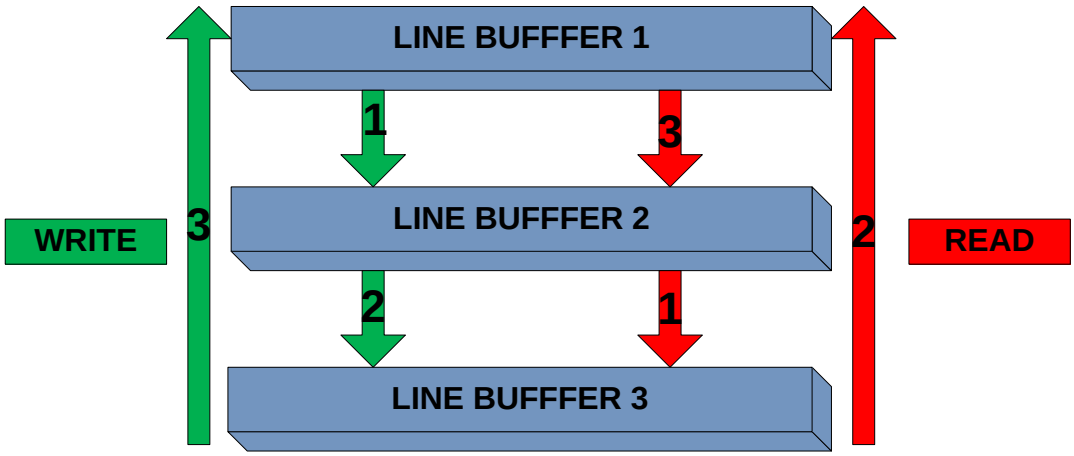


Figure 12 Triple Line Buffer Implementation

After transferring the input to the native domain, the video is divided into three parts, the first one being the electro-optic video, the second part is the IR video and the last part being discarded. The discarded part of the video is the lower part of the video which carries no valuable data.

After processing the input video, the two decomposed videos are passed to the video processing block where the proposed fusion algorithm applied.

4.3.2 Video Processing Block

VPB (Video Processing Block), is the part where the two input videos are fused and written to the external frame buffer. This block is firstly divided into two identical blocks while one block processes the first 256 columns of the input videos where the other one processes the next 256 columns of the two input videos. Such a division is a must since it is not possible to complete the fusion of a line in a time interval of a horizontal sync signal. VPB looks like:

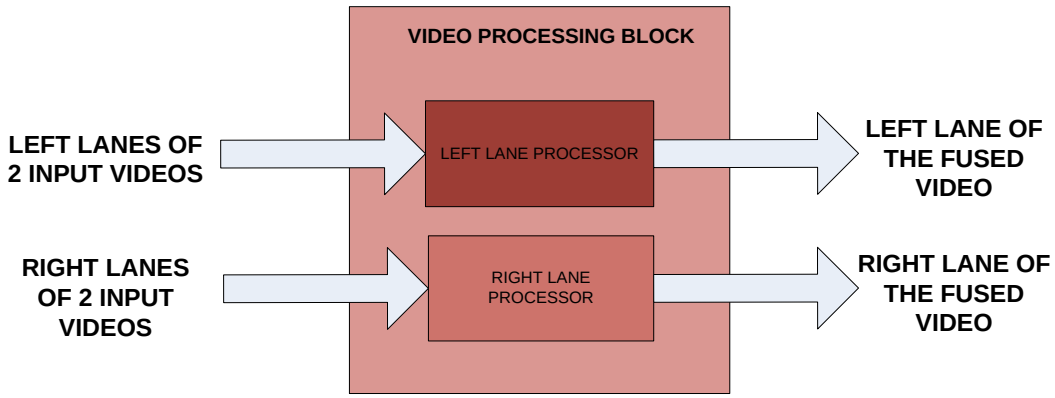


Figure 13 Internal Structure of VPB

Here, the left and right lane are described together since both handle the same algorithm, while the LLP (Left Lane Processor) handles the left and RLP (Right Lane Processor) handles the right lane of two input videos. The next section describes the inner structure of both LLP and RLP.

4.3.2.1 Inner Structure of LLP and RLP

LLP and RLP are the main processing blocks of the design. The whole 2D DCT, video fusion and reverse 2D DCT implementations are done within these blocks. The block diagram of the LLP and RLP is shown below:

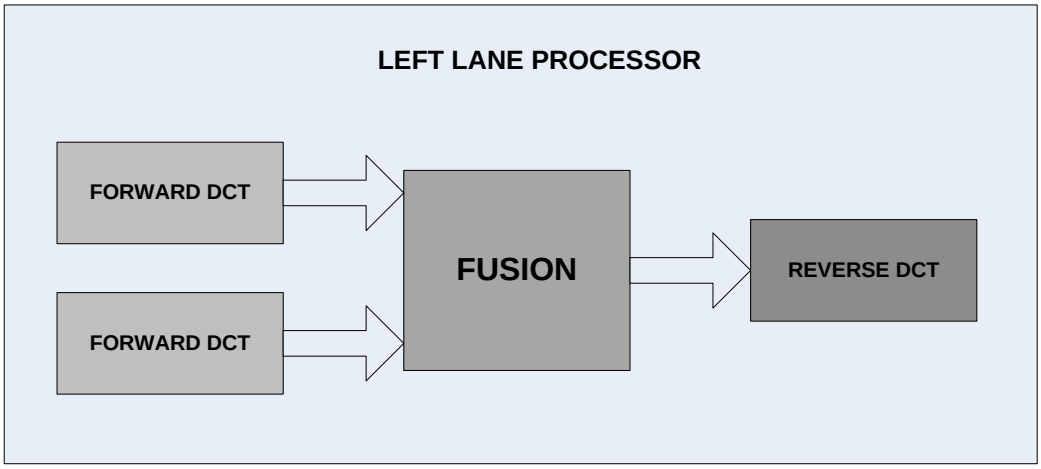


Figure 14 Internal Structure of LLP

As shown in the figure above; LLP and RLP are composed of three main blocks. The input images are first transformed to the DCT domain, after the proposed fusion algorithm is applied in the *fusion* block; the fused video is back transformed to the time domain and conveyed to the *output video write-read* block. Below, the inner structures of LLP and RLP are explained:

4.3.2.1.1 Forward 2D DCT Implementation

In the theory of the DCT transform, the output of the DCT can be written as an independent product of the input and the transformation matrix. In 2D DCT; the equation is:

$$X = T.x.T^T \qquad \textbf{4-1}$$

Here, X is the 2D DCT transform of matrix x, and T is the transformation matrix. The transformation matrix is definite for a given 2D DCT block size. In many applications including JPEG, 8x8 DCT blocks are used as a rule of thumb, however in the proposed algorithm; it is observed that 4x4 DCT block performs better than 8x8 in terms of *blocking effect*. The fixed 4x4 2D forward DCT transformation matrix can be found by means of *dctmtx* function of Matlab®. Below, the decimal

representation of the 4x4 2D DCT matrix; which is symbolized with letter *T*, is given:

$$T = \begin{bmatrix} 0.5000 & 0.5000 & 0.5000 & 0.5000 \\ 0.6533 & 0.2706 & -0.2706 & -0.6533 \\ 0.5000 & -0.5000 & -0.5000 & 0.5000 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{bmatrix} \quad \textbf{4-2}$$

The implementation of forward DCT in this design is handled in two steps. First the Tx term is calculated and shown by *X'*, after $X'T^T$ term is calculated. The calculation is in the following way:

Let *x* be a for $4x4$ matrix such that:

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \quad \textbf{4-3}$$

Then, the *X'* becomes:

$$X' = T. \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \quad \textbf{4-4}$$

The block diagram of the implemented VHDL code for the equation 4-4 can be drawn as an inner structure of a *Multiply & Accumulate (MAC)* block. For the first term of the matrix X', the block diagram is shown below:

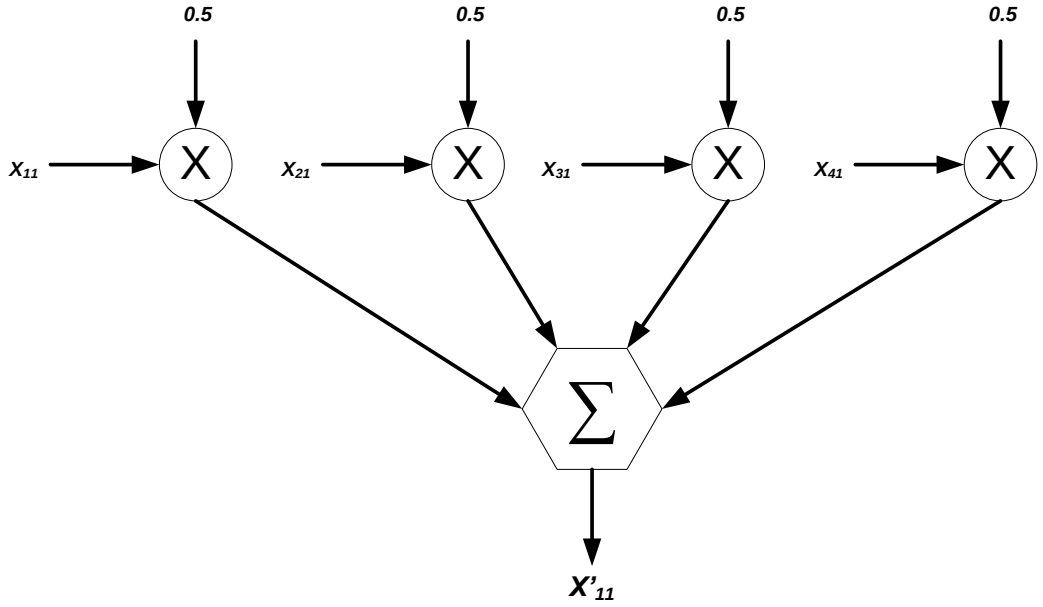


Figure 15 Multiply & Accumulate Block for Forward 2D DCT

The other terms of X' is calculated by using the above MAC block while changing the inputs to the 4 multipliers.

Since *T* is a decimal matrix, we can calculate the value directly:

$$X'_{11} = 0.5\,x_{11} + 0.5\,x_{21} + 0.5\,x_{31} + 0.5\,x_{41}$$
$$X'_{12} = 0.5\,x_{12} + 0.5\,x_{22} + 0.5\,x_{32} + 0.5\,x_{42}$$
$$X'_{13} = 0.5\,x_{13} + 0.5\,x_{23} + 0.5\,x_{33} + 0.5\,x_{43}$$
$$X'_{14} = 0.5\,x_{14} + 0.5\,x_{24} + 0.5\,x_{34} + 0.5\,x_{44}$$
$$X'_{21} = 0.6533\,x_{11} + 0.2706\,x_{21} - 0.2706\,x_{31} - 0.6533\,x_{41}$$
$$X'_{22} = 0.6533\,x_{12} + 0.2706\,x_{22} - 0.2706\,x_{32} - 0.6533\,x_{42}$$
$$X'_{23} = 0.6533\,x_{13} + 0.2706\,x_{23} - 0.2706\,x_{33} - 0.6533\,x_{43}$$
$$X'_{24} = 0.6533\,x_{14} + 0.2706\,x_{24} - 0.2706\,x_{34} - 0.6533\,x_{44}$$
$$X'_{31} = 0.5\,x_{11} + 0.5\,x_{21} + 0.5\,x_{31} + 0.5\,x_{41}$$
$$X'_{32} = 0.5\,x_{12} + 0.5\,x_{22} + 0.5\,x_{32} + 0.5\,x_{42}$$
$$X'_{33} = 0.5\,x_{13} + 0.5\,x_{23} + 0.5\,x_{33} + 0.5\,x_{43}$$
$$X'_{34} = 0.5\,x_{14} + 0.5\,x_{24} + 0.5\,x_{34} + 0.5\,x_{44}$$

$$X'_{41} = \ 0.2706\, x_{11} - 0.6533\, x_{21} + 0.6533\, x_{31} - 0.2706\, x_{41}$$
$$X'_{42} = \ 0.2706\, x_{12} - 0.6533\, x_{22} + 0.6533\, x_{32} - 0.2706\, x_{42}$$
$$X'_{43} = \ 0.2706\, x_{13} - 0.6533\, x_{23} + 0.6533\, x_{33} - 0.2706\, x_{43}$$
$$X'_{44} = \ 0.2706\, x_{14} - 0.6533\, x_{24} + 0.6533\, x_{34} - 0.2706\, x_{44}$$

4-5

After calculating *X'*, *X* term can be calculated by $X'.T^T$. The result is depicted below:

$$X_{11} = \ 0.5\, x_{11} + 0.5\, x_{12} + 0.5\, x_{13} + 0.5\, x_{14}$$
$$X_{12} = \ 0.6533\, x_{11} + 0.2706\, x_{12} - 0.2706\, x_{13} - 0.6533\, x_{14}$$
$$X_{13} = \ 0.5\, x_{11} - 0.5\, x_{12} - 0.5\, x_{13} + 0.5\, x_{14}$$
$$X_{14} = \ 0.2706\, x_{11} - 0.6533\, x_{12} + 0.6533\, x_{13} - 0.2706\, x_{14}$$
$$X_{21} = \ 0.5\, x_{21} + 0.5\, x_{22} + x_{23} + 0.5\, x_{24}$$
$$X_{22} = \ 0.6533\, x_{21} + 0.2706\, x_{22} - 0.2706\, x_{23} - 0.6533\, x_{24}$$
$$X_{23} = \ 0.5\, x_{21} - 0.5\, x_{22} - 0.5\, x_{23} + 0.5\, x_{24}$$
$$X_{24} = \ 0.2706\, x_{21} - 0.6533\, x_{22} + 0.6533\, x_{23} - 0.2706\, x_{24}$$
$$X_{31} = \ 0.5\, x_{31} + 0.5\, x_{32} + 0.5\, x_{33} + 0.5\, x_{34}$$
$$X_{32} = \ 0.6533\, x_{31} + 0.2706\, x_{32} - 0.2706\, x_{33} - 0.6533\, x_{34}$$
$$X_{33} = \ 0.5\, x_{31} - 0.5\, x_{32} - 0.5\, x_{33} + 0.5\, x_{34}$$
$$X_{34} = \ 0.2706\, x_{31} - 0.6533\, x_{32} + 0.6533\, x_{33} - 0.2706\, x_{34}$$
$$X_{41} = \ 0.5\, x_{41} + 0.5\, x_{42} + 0.5\, x_{43} + 0.5\, x_{44}$$
$$X_{42} = \ 0.6533\, x_{41} + 0.2706\, x_{42} - 0.2706\, x_{43} - 0.6533\, x_{44}$$
$$X_{43} = \ 0.5\, x_{41} - 0.5\, x_{42} - 0.5\, x_{43} + 0.5\, x_{44}$$
$$X_{44} = \ 0.2706\, x_{41} - 0.6533\, x_{42} + 0.6533\, x_{43} - 0.2706\, x_{44}$$

4-6

Here, the matrix *X* is the DCT of the input matrix. After, calculating *X* in two steps, the result is send to the fusion block. Since there are two input images, there are two DCT blocks one for each.

4.3.2.1.2 Fusion Stage

In this application, there are two input images, so there are two different DCT domain data is input to the fusion block. The fusion block, basically takes the two inputs and implements the proposed fusion algorithm and sends the output to next block which is the reverse 2D DCT implementation.

Internally, fusion block is composed of two identical sub fusion blocks which work in a ping-pong manner. While one sub block processes the incoming 4x4 block, the other one receives the next 4x4 block relaxing the computation time budget. By doing so, the source usage increases by 2, at the same time the computational time demand is reduced by a factor of 2.

The block diagram of ping-pong based structure of the fusion block is drawn below:

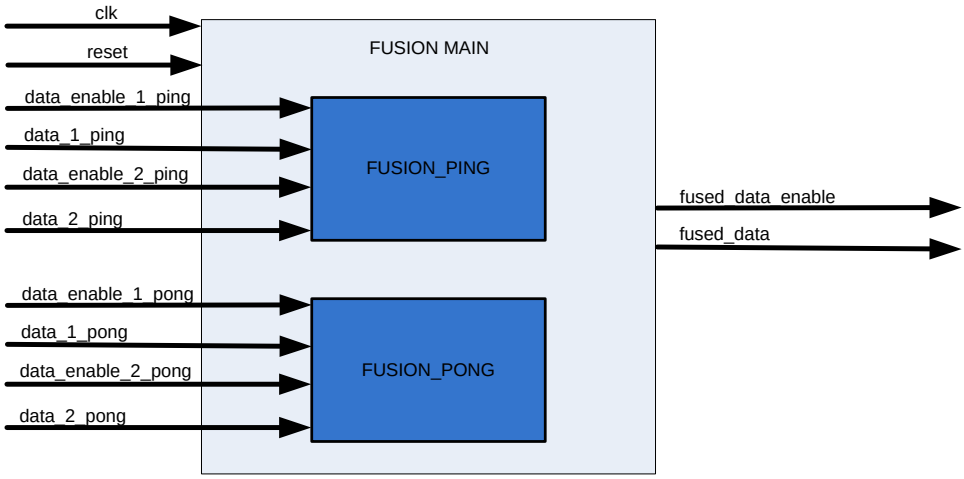


Figure 16 Ping-Pong Based Fusion Scheme

The *ping* and *pong* fusion blocks basically implement the same algorithm, however; they work in the opposite time slots. The main collaborative working principle is drawn below:

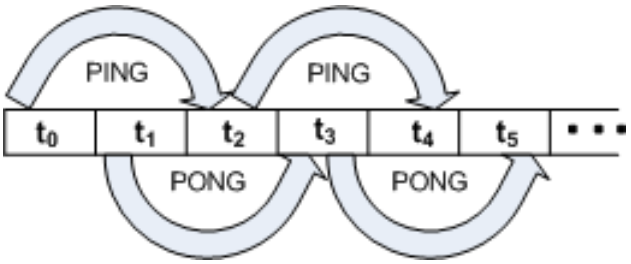


Figure 17 A Ping-Pong Based Architecture

In time slot t_0, *fusion ping* block gets the data and processes it until the time slot t_2; in time slot t_1, *fusion pong* block gets the data and processes it until the time slot t_3. By doing so, the processing time is relaxed by the factor of two while doubling the resource usage.

For sake of simplicity, only the working principle of *fusion ping* block is described herein because *ping* and *pong* blocks differ only in the data latch and execute time slots.

Fusion ping block is composed of 5 main states:

i. Input data from two sources are latched.
ii. The L2 Norms of the input matrices are calculated.
iii. The DC value of the output DCT matrix is calculated by averaging.
iv. The AC values of the DCT matrix are calculated pixel by pixel with the weighting coefficient calculated in the third step.
v. The DC and AC values of the output matrix are sent to the IDCT block.

In state machine notation the operation of the fusion block can be drawn as:

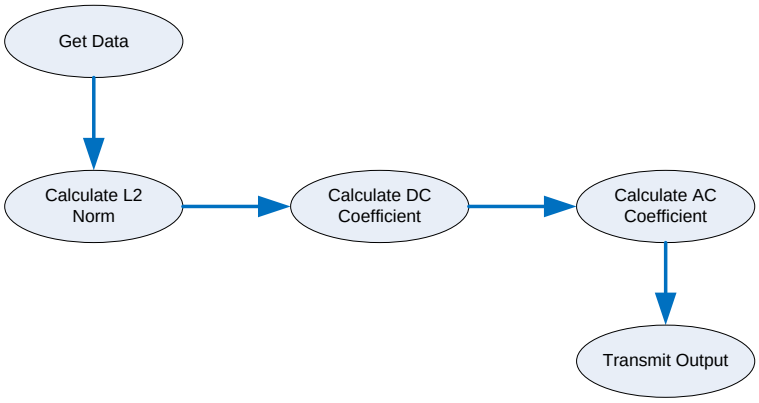


Figure 18 Fusion Block State Transition Diagram

The calculated 4x4 fused data matrix is then transmitted to the *reverse 2D IDCT* block to be transferred back to the time domain for displayable output.

4.3.2.1.3 Reverse 2D DCT Implementation

The 4x4 matrix sequence sent form the *ping-pong* based fusion block is input to the IDCT block which implements the reverse 2D DCT. This block collects both *ping* and *pong* fusion block outputs and calculates the inverse DCT transform of them. The working principle of this block is very similar to that of the forward DCT block. The only difference is the formulation. In IDCT the equation is as follows:

$$x = T^T.X.T \qquad \textbf{4-7}$$

Here, x is the time domain representation of $4x4$ matrix and X is the DCT of it. The above equation is handled in two steps; first ($T^T.X$) expression is calculated, after the result is multiplied with T. If x' represents $T^T.X$ then:

$$x'_{11} = 0.5\,X_{11} + 0.6533\,X_{21} + 0.5\,X_{31} + 0.2706\,X_{42}$$
$$x'_{12} = 0.5\,X_{12} + 0.6533\,X_{22} + 0.5\,X_{32} + 0.2706\,X_{42}$$
$$x'_{13} = 0.5\,X_{13} + 0.6533\,X_{23} + 0.5\,X_{33} + 0.2706\,X_{43}$$
$$x'_{14} = 0.5\,X_{14} + 0.6533\,X_{24} + 0.5\,X_{34} + 0.2706\,X_{44}$$

$$x'_{21} = 0.5\,X_{11} + 0.2706\,X_{21} - 0.5\,X_{31} - 0.6533\,X_{42}$$
$$x'_{22} = 0.5\,X_{12} + 0.2706\,X_{22} - 0.5\,X_{32} - 0.6533\,X_{42}$$
$$x'_{23} = 0.5\,X_{13} + 0.2706\,X_{23} - 0.5\,X_{33} - 0.6533\,X_{43}$$
$$x'_{24} = 0.5\,X_{14} + 0.2706\,X_{24} - 0.5\,X_{34} - 0.6533\,X_{44}$$
$$x'_{31} = 0.5\,X_{11} - 0.2706\,X_{21} - 0.5\,X_{31} + 0.6533\,X_{42}$$
$$x'_{32} = 0.5\,X_{12} - 0.2706\,X_{22} - 0.5\,X_{32} + 0.6533\,X_{42}$$
$$x'_{33} = 0.5\,X_{13} - 0.2706\,X_{23} - 0.5\,X_{33} + 0.6533\,X_{43}$$
$$x'_{34} = 0.5\,X_{14} - 0.2706\,X_{24} - 0.5\,X_{34} + 0.6533\,X_{44}$$
$$x'_{41} = 0.5\,X_{11} - 0.6533\,X_{21} + 0.5\,X_{31} - 0.2706\,X_{42}$$
$$x'_{42} = 0.5\,X_{12} - 0.6533\,X_{22} + 0.5\,X_{32} - 0.2706\,X_{42}$$
$$x'_{43} = 0.5\,X_{13} - 0.6533\,X_{23} + 0.5\,X_{33} - 0.2706\,X_{43}$$
$$x'_{44} = 0.5\,X_{14} - 0.6533\,X_{24} + 0.5\,X_{34} - 0.2706\,X_{44}$$

4-8

After constituting *x'*, *x* is calculated as *x'.T*:

$$x_{11} = 0.5\,x_{11} + 0.6533\,x_{12} + 0.5\,x_{13} + 0.2706\,x_{14}$$
$$x_{12} = 0.5\,x_{11} + 0.2706\,x_{12} - 0.5\,x_{13} - 0.6533\,x_{14}$$
$$x_{13} = 0.5\,x_{11} - 0.2706\,x_{12} - 0.5\,x_{13} + 0.6533\,x_{14}$$
$$x_{14} = 0.5\,x_{11} - 0.6533\,x_{12} + 0.5\,x_{13} - 0.2706\,x_{14}$$
$$x_{21} = 0.5\,x_{21} + 0.6533\,x_{22} + 0.5\,x_{23} + 0.2706\,x_{24}$$
$$x_{22} = 0.5\,x_{21} + 0.2706\,x_{22} - 0.5\,x_{23} - 0.6533\,x_{24}$$
$$x_{23} = 0.5\,x_{21} - 0.2706\,x_{22} - 0.5\,x_{23} + 0.6533\,x_{24}$$
$$x_{24} = 0.5\,x_{21} - 0.6533\,x_{22} + 0.5\,x_{23} - 0.2706\,x_{24}$$
$$x_{31} = 0.5\,x_{31} + 0.6533\,x_{32} + 0.5\,x_{33} + 0.2706\,x_{34}$$
$$x_{32} = 0.5\,x_{31} + 0.2706\,x_{32} - 0.5\,x_{33} - 0.6533\,x_{34}$$
$$x_{33} = 0.5\,x_{31} - 0.2706\,x_{32} - 0.5\,x_{33} + 0.6533\,x_{34}$$
$$x_{34} = 0.5\,x_{31} - 0.6533\,x_{32} + 0.5\,x_{33} - 0.2706\,x_{34}$$
$$x_{41} = 0.5\,x_{41} + 0.6533\,x_{42} + 0.5\,x_{43} + 0.2706\,x_{44}$$
$$x_{42} = 0.5\,x_{41} + 0.2706\,x_{42} - 0.5\,x_{43} - 0.6533\,x_{44}$$
$$x_{43} = 0.5\,x_{41} - 0.2706\,x_{42} - 0.5\,x_{43} + 0.6533\,x_{44}$$

$$x_{44} = \quad 0.5\, x_{41} - 0.6533\, x_{42} + 0.5\, x_{43} - 0.2706\, x_{44}$$

4-9

The above calculated *x* matrix is sent to the *output video memory write block* to be written to the frame buffer created on the SRAM (Synchronous Random Access Memory).

4.3.3 Output Video Memory Write-Read

The *output video write-read* block is composed of two main blocks namely; *output video write* and *output video read and display*. The block diagram of *Output Video Memory Write-Read* block is drawn below:

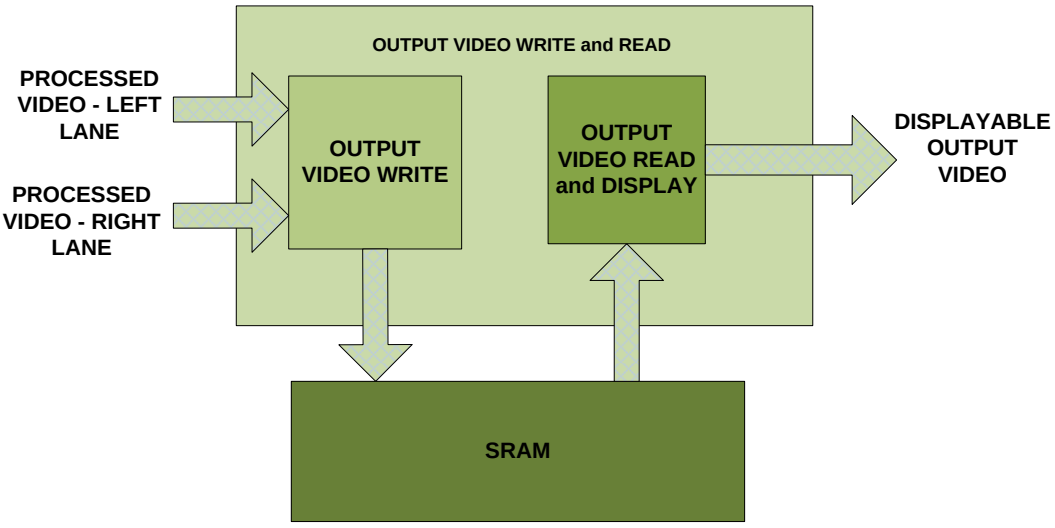


Figure 19 Output Video Write & Read

There are two video outputs from the VBP. One of these outputs is the left lane, the other one is the right lane of the fused video. These outputs are of the form of 4x4 matrices which makes them unavailable for output monitor display. The digital monitors accept videos which comply with the DMT (Digital Monitor Timing) standard. In DMT standard, the video is composed of frames. The frames are formed by lines. The timing values for frames and lines are strictly defined. In order to act in accordance with the DMT standard, the video should be sent line by line with the

corresponding sync signals. So, the fused video is written to an external SRAM. There is a definite address space inside the SRAM according to the output video size. The incoming video is written to the address space and stored until it is read.

For reading and displaying the video frame stored inside the SRAM, *output video read* block is constructed. This block reads the fused video inside the SRAM with its own generated sync signals. Then the video is sent to the *DVI transmitter chip* for digital monitor display.

4.3.3.1 Output Video Memory Write

The Reverse 2D DCT Implementation block creates and transmits 4x4 inverse DCT transformed blocks. However, DMT standard requires the video to be sent in the form of complete video lines. This prevents the output video to be sent to an ordinary DVI transmitter IC (Integrated Circuit) since the incoming data sequence is not in the form of video lines. So, the incoming data should be collected in order to form a complete line of a video frame. The visual video transmission diagram for VESA (Video Electronics Standards Association) DMT standard is drawn below:

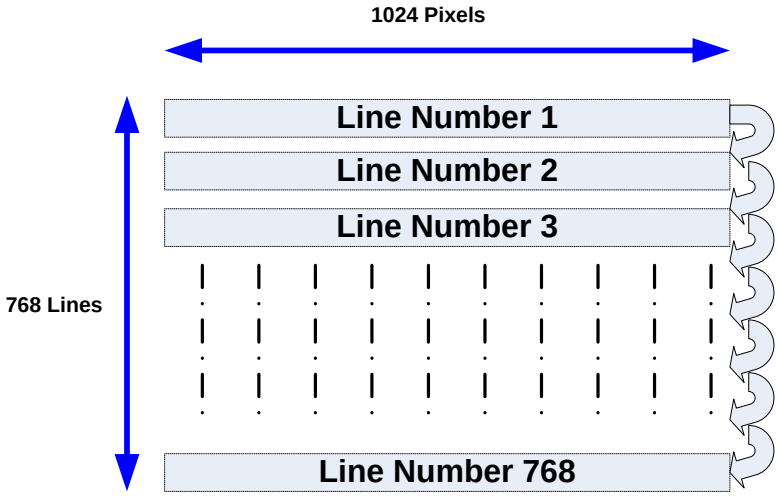


Figure 20 Video Transmission Topology

The above transmission methodology is not consistent with the incoming data sequence from the *Reverse 2D DCT Implementation block.* So, we need to store the incoming data until a full video line is arrived. In this case, a total of 4 lines are received since data is transferred in terms of 4x4 pixel blocks. At this point, the video lines are ready to transmit; however, since we have to supply a consistent flow to the output video stream, the incoming data stream might not be consistent with the DMT in terms of timing issues. Instead, the incoming data can be stored in a frame buffer eliminating this ambiguity. If the video is stored in a frame buffer and read from then there will be no transmission timing violation since the video is read form the buffer frame by frame separately.

A frame buffer is composed of a full video frame. Its size is exactly the same with a full video frame. The capacity of a frame buffer of 1024x768 RGB video is calculated as such:

$$768(\text{lines})\text{x}\ 1024(\text{pixels})\text{x}\ 3(\text{RGB}) = 2359296\ \text{Bytes} \qquad \textbf{4-10}$$

Here, each color channel is expressed with 8 bits. However, in our case, the video is monochrome which eliminates the third multiplier, resulting with 786432 Bytes. The needed memory space for a frame buffer allocation is practically 8MB. In today's technology, it is very expensive to implement such a large memory inside the FPGA, instead external memories are used as frame buffers.

There are mainly two types of RAM to be used as a frame buffer, namely SRAM and SDRAM (Synchronous Dynamic Random Access Memory). The main difference between the two in terms of architecture is the former one being static whereas the latter one is dynamic. Dynamic RAMs are capacitor based RAMs which cannot store the value without refresh cycles. Refresh cycles refills the capacitors in the RAM preventing them to lose charge because of leakage current. Apart from the refresh cycles, SDRAMs have much higher capacity and speed ratings compared to the

SRAMs. However, SDRAM control is difficult with respect to SRAM because SDRAM addressing topology differs from that of SRAM.

In applications where small memory space is enough, usually SRAMs are preferred thanks to the easiness in control and implementation. On the other hand, SDRAMs serves better in demanding applications while increasing the design time.

In this application, SRAM is used as a frame buffer because of the above stated reasoning. The SRAM IC present on the board has 20 bits of address bus which corresponds to 1 million addresses. The frame buffer needed in this case will occupy 1024x768 address locations if one address stores one pixel. In our implementation, the address bus is divided into two where the first 10 bits are used to address frame columns (pixels) and the last 10 bits are used to address frame rows (lines). The schematic view of the frame buffer is shown below:

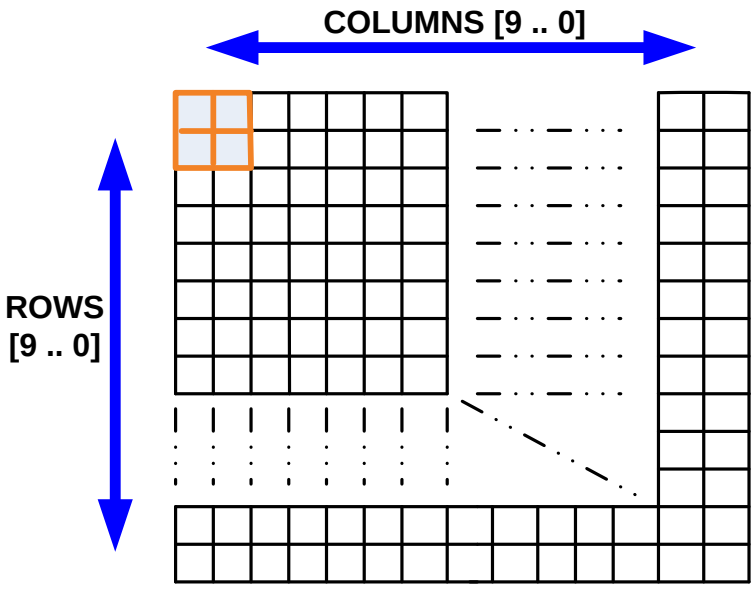


Figure 21 Frame Buffer Address Scheme

In the above figure; the first incoming 4x4 data set is placed in the first 4x4 sub block in the frame buffer which is shown with orange color. The next upcoming data sets are put next to each other until the end of the column is reached. When the last

column is addressed, then the next lower 4x4 sub block is addressed starting from the first column.

In our implementation, the input video size is limited to 512x512. So, the addressed space is 512x512 locations which use only most significant 9 bits of the address.

4.3.3.1 Output Video Memory Read

After the video frame is written to the frame buffer, it is read with respect to the 1024x768 @60Hz video transmission standard of VESA DMT.

In order to construct a digital video, there are certain signals to be created along with the clock signal. In DMT standard, there are 3 control signals along with the video data which are *horizontal sync*, *vertical sync*, and *data enable* (negated blank). The *vertical sync* signal controls the video frames, *horizontal sync* signal controls the video lines and the *data enable* signal controls the active video lines within a *horizontal sync*.

Vertical sync and horizontal sync signals are periodic signals whereas data enable signal is high only when transmitting active pixels. The timing diagram for sync signals is shown below:

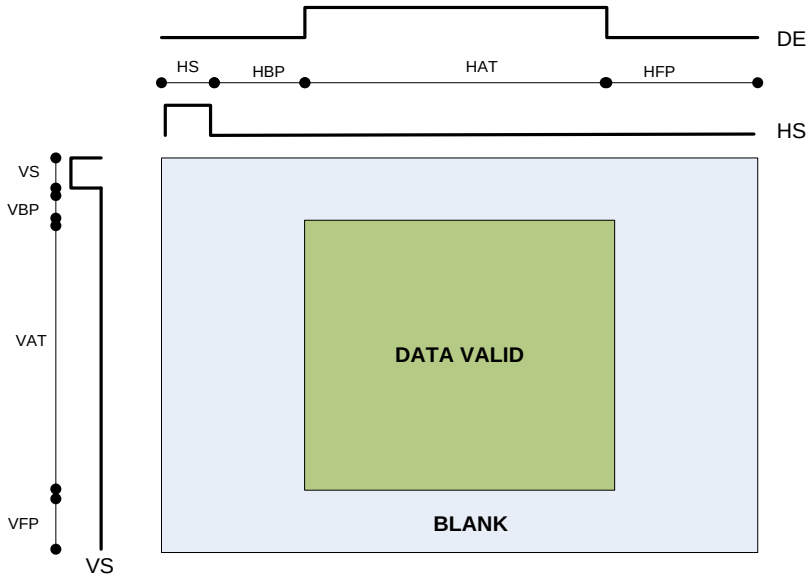


Figure 22 DMT Convention

Here, VS denotes *vertical sync*, HS denotes *horizontal sync* and DE denotes *data enable*. All of these signals are generated and sent with respect to the system clock which is 65MHz. Below; the numerical values for the timing abbreviations at 1024x768 @60Hz resolution are given:

Table 4 DMT values for 1024x768 @60 Hz Video

Parameter	***Value***	***Unit***
VS (Vertical Sync)	6	*lines*
VBP (Vertical Back Porch)	29	*Lines*
VAT (Vertical Address Time)	768	*Lines*
VFP (Vertical Front Porch)	3	*Lines*
HS (Horizontal Sync)	136	*Pixels*
HBP (Horizontal Back Porch)	160	*Pixels*
HAT (Horizontal Address Time)	1024	*Pixels*
HFP (Horizontal Front Porch)	24	*Pixels*

Since the read video frame is of size 512x512, then less than a half of the screen remains empty. Hypothetically, the output video is placed at the center of the screen

while the other pixels are white. The output fused image looks like in the figure shown below:

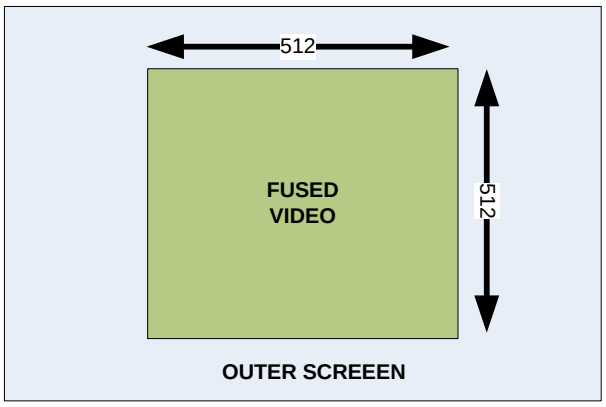


Figure 23 Appearance of Output Screen

4.3.4 Malfunctions & Related Reasoning

In FPGA based designs, the output of the system is not always the same as the theoretically calculated output. The reason for this is FPGA platform is a real time operating platform with combination of millions of logical gates. The connection between the gates in the design is most of the time very complex resulting with race conditions, state machine malfunctioning and several timing issues. However, in today's technology; many of these obstacles can be eliminated somehow. Complex FPGA designs are firstly implemented in hardware and then timing constraints and design safety issues are handled in order to create a robust functioning design.

Apart from the above stated reasoning, FPGAs have another downside compared to DSPs and other processors which is the disability of floating point operation. This situation complicates the implementation of algorithms which uses considerably small decimal numbers. In this case, the algorithm is modified such that the small decimal numbers are transferred to the nearest integer.

Below, the troubles encountered during hardware design are classified and characterized explicitly.

4.3.4.1 Output Video Quality Reduction due to Fixed Point Arithmetic

As seen from the equations 4-5, 4-6, 4-8 and 4-9 the transform coefficients are small decimals which are very hard to work within FPGA. So, an appropriate manipulation on the calculation matrices is needed in order not to lose the valuable information traversing among the signal processing blocks.

In all computer software (e.g. Matlab®, Mathematica®) and in DSPs the computations are handled by floating point algorithms where the decimal part of the numbers does not matter no matter how long they are. However, in FPGAs decimal computations are very hard to handle since the inner structure of the FPGAs are optimized for fixed point integer computations. Hence, in order to implement 2D DCT in FPGA, the given matrix should be manipulated. In order to overcome this ambiguity, at the forward DCT calculation block, *T* matrix is multiplied by 2^{13}. All the calculations are handled with this multiplied matrix. At the end of second step, the output is divided by 2^{10} and the result sent to the fusion block is the 8 times greater than the actual result. Data is processed in the fusion block without normalization and sent to the reverse DCT calculation block. The first step of the reverse DCT calculation block recognizes the multiplied input and processes it after multiplying with 2^{10}. At the output stage of the first step reverse DCT calculation block, the result is divided by 2^{10} and sent to the second stage reverse DCT calculation block. At this step, the data on the bus is 8 times greater than the original data. As a last step, the second stage reverse DCT calculation block accepts the incoming multiplied data and calculates equation 4-7 after multiplying the *T* matrix with 2^{10}. At the output, the result is divided by 2^{13} to get the actual final result. This manipulation decreases the round up error.

As seen from the above explanation, there is always a round-up error introduced at each signal processing block in the process even if the division and multiplication factors are optimized to reduce the division error. The optimization is made in the view of DCT transformation matrix *T*. The possible output range from all blocks is estimated and appropriate multiplication and division factors are calculated and applied accordingly.

4.3.4.2 Output Video Quality Reduction due to Output Clock Speed

The SRAM component used as frame buffer is an external peripheral of FPGA. Since, it is located outside the FPGA, direct access to the device is not possible. A simple controller is needed to establish read and write cycles between FPGA and SRAM. This controller takes the data generated within the FPGA and sends it to the SRAM with the appropriate timings. SRAM controller behaves as a two sided buffer between FPGA and SRAM. Since it is a double sided buffer, there is a certain amount of performance loss and hence it has certain efficiency. This efficiency determines the actual performance of the system. The implemented design has a simple SRAM controller with an average efficiency of 75%.

In order to have a relaxed frame buffer topology, the read and write frequencies should be compatible with the SRAM controller clock speed and efficiency. The implemented design has a write and read frequency of 65MHz. Since, the efficiency of the design is 75% and the read and write cycles has a total speed of 130MHz; the minimum required SRAM frequency is 174MHz. The SRAM located on the board is running at a speed of 175 MHz which is slightly greater than the required value.

As mentioned, the efficiency of the controller is varying by time and the efficiency is calculated approximately. This situation causes defects at the output image. The pixel based flicks on the output image are seen because of the inefficiency of the frame buffer read and write cycles.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

This thesis investigates an alternative approach to the video and image fusion literature. The amount of papers and knowledge about image fusion based on DCT technique has not been saturated yet. The known papers about DCT based IF are written by Zafar et al and J. Tang. However neither of them proposes an appealing and decent algorithm which utilizes DCT technique with all aspects. Here, a different and more sophisticated algorithm is presented and applied on hardware.

It is depicted in the performance algorithms that the proposed algorithm challenges with the state of the art IF algorithms well, even in some cases it results better. Apart from this competitive performance, the proposed algorithm exploits rather easy hardware implementation logic in terms of performance and speed. The most noticeable distinction in hardware is the proposed algorithm does not need any frame buffers of any type since it does not work in a multi scale manner which eliminates the need of external memory and the read and writes cycles between the memory and the FPGA.

In hardware design, memory placement and performance issues demands a serious effort which increases the cost and time to market delay. That's why the proposed algorithm is definitely preferred in any type of hardware design.

As future work, the algorithm can be further optimized in software. These optimizations can include 8x8 2D DCT with less discontinuity; achieve better

blocking effect reductions even with less number of computations. Moreover, the algorithm can be expanded in order to cover RGB images while applying the same methodology in all three color bands.

In terms of hardware optimizations, the forward and reverse DCT calculation technique can be shrunk to use less multiplier which reduces the cost dramatically. Optionally, the frame buffer used at the output stage can be eliminated. This reduction reduces both the design cost and eliminates the digital noise arise from the read and write cycles. In fact, the elimination of the output frame buffer is a must in this case because; the board cannot perform well with the output buffer at the speed of 65 MHz

In a terrestrial application of this algorithm, the input sources will be two different cameras most likely. In this case, the input video acquisition block must be divided in to two to capture two independent inputs. This operation makes no significant effect on the design cost and methodology.

Apart from all the advised improvements on both the software and hardware implementation, the proposed algorithm serves well as a baseline for the further DCT based image fusion algorithms. The future work can be built on this fundamental basis.

REFERENCES

1. **Barbara Zitova, Jan Flusser.** Image registration methods: a survey. *Image and Vision Computing.* October 2003, Vol. 21, 11.

2. **Bartoli, Guido.** *Image Registration Techniques: A Comprehensive Survey.* Siena : Universita degli Studi di Siena, June 2007.

3. **H. Li, B. S. Manjunath and S. K. Mitra.** Multisensor Image Fusion Using the Wavelet Transform. *Graphical Models and Image Processing.* May 1995, Vol. 57, 3.

4. *Edge-preserving wavelet-based multisensor image fusion approach.* **Lahouari Ghouti, Ahmed Bouridane and Mohammad K. Ibrahim.** Vienna, Austria : XII. European Signal Processing Conference, 2004.

5. *Real time implementation of image alignment and fusion.* **D. Dwyer, D. L. Hickman et al.** Orlando : Defense and Security Symposium, 2006.

6. *Pixel-based and region-based image fusion schemes using ICA bases.* **Nikolaos Mitianoudis, Tania Stathakia.** 2, s.l. : Information Fusion , April 2007, Vol. 8.

7. *Evaluation of Algorithms for Fusing Infrared and Synthetic Imagery.* **Philippe Simard, Norah K. Link and Ronald V. Kruk.** Montreal : SPIE Enhanced and Synthetic Vision 2000, 2000.

8. **Z. Xue, R S. Blum, and Y. Li.** *Fusion of Visual and IR Images for Concealed Weapon Detection.* Bethlehem : ISIF, 2002.

9. **Nikolaos Mitianoudis, Tania Stathaki.** Pixel based and Region based Image Fusion schemes using ICA bases. *Elsevier Science.* December, 2007.

10. **Alexander Toet, & Walraven, J.** New false colour mapping for image fusion. *Optical Engineering.* 35, 1996, Vol. 3.

11. **Alexander Toet, Eric M. Franken.** *Perceptual evaluation of different image fusion schemes.* [24] The Netherlands : Elsevier Science, 2003. S0 14 1 -9 38 2 (0 2) 00 0 69 -0.

12. **Huang, Guanghua, Ni, Guoqiang and Zhang, Bin.** Visual and infrared dual-band false color image fusion method motivated by Land's experiment. *Optical Engineering.* 46, 2007, Vol. 02, 027001.

13. *Pseudo-Color Image Fusion Based on Intensity-Hue-Saturation Color Space.* **Ra, J. H. Jang and J. B.** Seoul, Korea : Proceedings of IEEE, 2008. 978-1-4244-2144-2/08.

14. *Multiresolution based based image fusion with additive wavelet decomposition.* **Nunez, J., Otazu, X., Fors, O., Prades, A., Pala, V., and Arbiol, R.** 3, s.l. : IEEE Transactions, Geoscience Remote Sens., 1999, Vol. 37.

15. **Liu, Rick S. Blum and Zheng.** *Multi-Sensor Image Fusion and Its Applications.* Boca Raton : Taylor & Francis Group, LLC, 2006. 2005041824.

16. *Opponent-color fusion of multi-sensor imagery: visible, IR and SAR,.* **Waxman, A. M., Aguilar, M., Baxter, R. A., Fay, D. A., Ireland, D. B., Racamato, J. P., and Ross, W. D.** 43-61, s.l. : IRIS Passive Sens. proc., 1998, Vol. 1.

17. *Real-time fusion of low-light CCD and uncooled IR imagery for color night vision.* **Aguilar, M., Fay, D. A., Ross, W. D., Waxman, A. M., Ireland, D. B., and Racamato, J. P.** 124 – 135, s.l. : Proc. SPIE, 1998, Vol. 3364.

18. **Wei Liu, Jie Huang and Yongjun Zhao.** Image Fusion Based on PCA and Undecimated Discrete Wavelet Transform. *Neural Information Processing.* Hong Kong, China : Springer Berlin / Heidelberg, 2006.

19. **Tang, Jinshan.** A contrast based image fusion technique in the DCT domain. *www.elsevier.com/locate/dsp.* [Online] 08 30, 2003. [Cited: 21 01, 2010.] linkinghub.elsevier.com/retrieve/pii/S105120040300040X.

20. *Multi-Exposure & Multi-Focus Image Fusion in Transform Domain.* **Zafar, I. Edirisinghe, E.A. Bez, H.E.** Bangalore, India : IEEE Xplore, 2006 . 978-0-86341-671-2 .

21. *Multiresolution Signal Decomposition Schemes Part1*. **J. Goutsias, H.J.A.M. Heijmans.** USA, The Netherlands : IEEE Transactions on Image Processing, 2000.

22. **Gonzalez, Woods.** Wavelets and Multiresolution Processing. *Digital Image Processing.* s.l. : Prentice Hall, 2002.

23. *The Laplacian pyramid as a compact image code.* **Adelson, P.J. Burt and E.H.** 4, s.l. : IEEE Transactions on Communications, 1983, Vol. 31.

24. *Comparative Image Fusion Analysis.* **Sadjadi, Firooz.** s.l. : Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. 1063-6919/05.

25. **Anderson, Charles H.** *Filter-subtract-decimate hierarchical pyramid signal analyzing and synthesizing technique. 4718104* US, January 5, 1988.

26. *Review of Image Fusion Algorithms for Unconstrained Outdoor Scenes.* **Jiacnhao Zeng, Aya Sayedelahl, Tom Gilmore, and Mohamed Chouikha.** Washington, DC : ICSP2006 Proceedings, 2006.

27. **Alexander Toet, van Ruyven, J.J. & Valeton, J.M.** Merging thermal and visual images by a contrast pyramid. *Optical Engineering.* 7, 1989, Vol. 28.

28. *A Gradient-Based Hybrid Image Fusion Scheme Using Object Extraction.* **Milad Ghantous, Soumik Ghosh and Magdy Bayoumi.** Lafayette : ICIP, 2008. 978-1-4244-1764-3/08.

29. **Filth, Oli.** Wikipedia. [Online] Wikimedia Foundation Inc., May 13, 2010. [Cited: 05 20, 2010.] en.wikipedia.org/wiki/Discrete_Cosine_Transform.

30. **Wang, Ruye.** Definition of DCT. [Online] HMC Engineering, 10 13, 2009. [Cited: 05 20, 2010.] http://fourier.eng.hmc.edu/e161/lectures/dct/node1.html.

31. **Marshall, Dave.** The Discrete Cosine Transform (DCT). [Online] Cardiff University, 10 4, 2001. [Cited: 20 05, 2010.] http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html.

32. *Perceptual evaluation of different image fusion schemes.* **Franken, A. Toet and E. M.** I, s.l. : Displyvs, February 2003, Vol. 24.

33. **Raol, Jitendra R.** Pixel and Feature Level Image Fusion Concepts and Algorithms. *Multi-Sensor Data Fusion with MATLAB®.* Boca Raton : CRC PRess, 2010.

34. **Zhou Wang, Alan C. Bovik.** A Universal Image Quality Index. *IIIE Signal Processing Letters.* Y, March 2002, Vol. XX.

35. *A new quality metric for image fusion.* **Heijmans, Gemma Piella and Henk.** Barcelona, Spain : Proc. IEEE Int. Conf. on Image Processing, 2003.

36. **Nedeljko Cvejic, Artur Łoza, David Bull, and Nishan Canagarajah.** A Novel Metric for Performance Evaluation of Image Fusion Algorithms. *World Academy of Science.* August, 2005, 7.

37. *Objective Image Fusion Performance Measure.* **Petrović, C.S. Xydeas and V.** 4, s.l. : IEEE Electronics Letters, 17 Feb 2000, Vol. 36.

38. **Association, Video Electronics Standards.** *Computer Display Monitor Timing Standard.* [electronic] Milpitas, CA : VESA, October 29, 2004.

39. *Stratix III Device Family Overview.* San Jose : Altera Corporation, 2007. SIII51001-1.3.

40. *Merging Infrared and Color Visible Images with a Contrast Enhanced Fusion Method.* **Guangxin Li, Ke Wang.** s.l. : Proc. of SPIE, 2007. 657108-1.

41. en.wikipedia.org. *Wikipedia.* [Online] 5 8, 2010. http://en.wikipedia.org/wiki/Image_registration.

42. **Blum, Rick.** Investigations of Image Fusion. *Electrical Engineering and Computer Science Department* . [Online] Lehigh University. [Cited: May 18, 2010.] http://www.ece.lehigh.edu/SPCRL/spcrl.htm.

43. *Improved Human Detection Using Image Fusion.* **E. Thomas Gilmore III, Preston D. Frazier, M. F. Chouikha.** Kobe, Japan : Proceedings of the IEEE ICRA 2009, 2009, May.

44. *An iterative image registration technique with an application to stereo vision.* **Kanade, B.D. Lucas and T.** Washington, DC : Proc. Imaging Understanding Workshop, 1981.

APPENDIX A

FPGA DESIGN FLOW

FPGA design flow is mainly composed of three stages namely; coding, simulation, synthesis, device configuration and verification. These stages can be combined into one block diagram as follows:

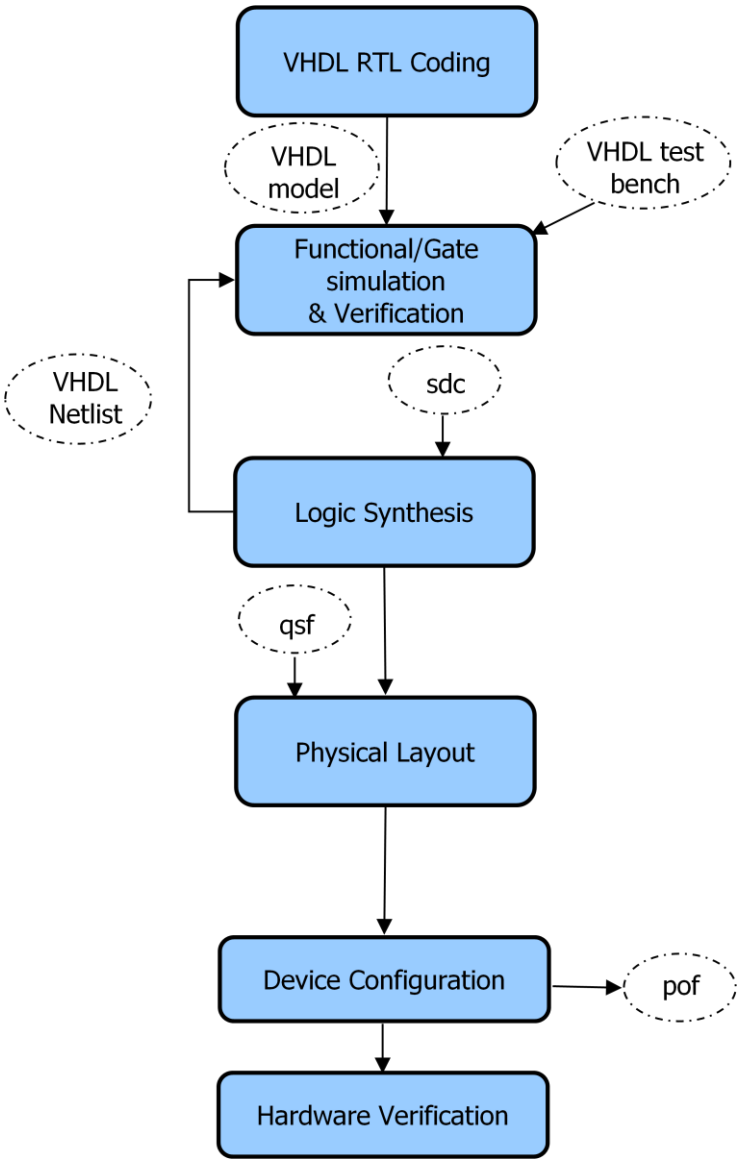


Figure 24 FPGA Design Flow

First of all, FPGA design starts with block diagram architecture. The function of the FPGA is clearly stated and the most convenient block diagram is determined. Later, VHDL coding of the pre-determined block diagram is completed. After accomplishing VHDL coding, the design is simulated functionally by means of VHDL simulators. While simulating, the behavioral simulation models of the peripheral blocks are coded and used as a test bench. Test bench is a non-synthesizable code which represents the behavior of the peripheral components of FPGA. The design is simulated so that its functional behavior is tested. If the

simulation is successful then the design is logically synthesized with timing constraints. After logical synthesis, the synthesis tool creates a net list file which includes all the delays created within the FPGA during logical synthesis. At this step, the design is re-simulated to see the logical synthesis defects. If the design works after logical synthesis then it can be synthesized physically. In other words, the net list file is implemented physically on FPGA. If the design is successfully fit on the FPGA, the programming files can be created afterwards. The programming file is a bit-wise streaming object file which is downloaded to the EEPROM device located outside the FPGA. This programming device automatically configures FPGA at each power up. The extension of the programming file is *pof* and it stands for *Programming Object File*.

There are several files created throughout the design process. "*Sdc*"; the acronym for *Synopsis Design Constraints file*, is created by the user and contains timing constraints about the design. "*qsf"* file is used throughout the synthesis process and it contains all the project information and settings. The pin layout of the FPGA is also present at *qsf*. The acronym *qsf* stands for *Quartus Settings File*.

In order to create a logical and synthesizable design, the designer should be aware of the internal construction of FPGA. The VHDL coding is constructed in accordance with the FPGA's internal architecture. Since this thesis is implemented on an Altera Stratix III FPGA, the internal structure of Stratix III series is explained herein.

A.1 Stratix III Architecture

Stratix III family FPGAs are industry's one of the fastest and largest FPGAs including adaptive logic modules (ALMs) and high-performance, flexible memory and DSP blocks. It has also fast external memory interfaces for designs require external buffers for video frame buffering. Below, the internal layout of a Stratix III FPGA is given:

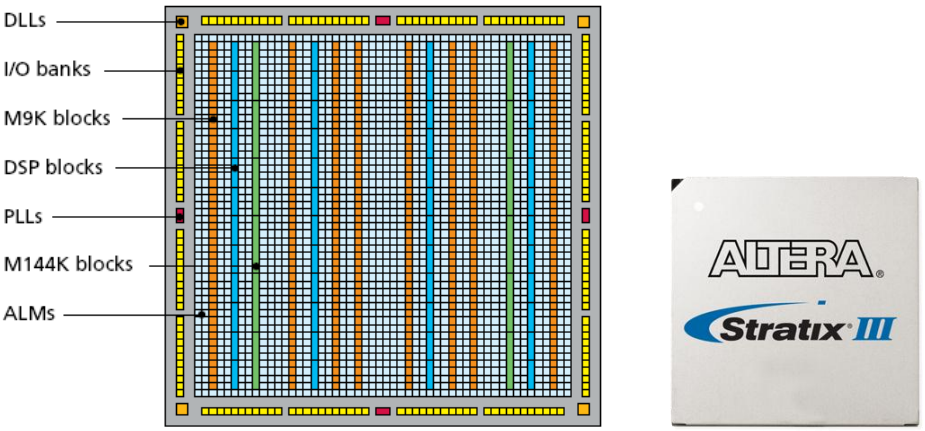


Figure 25 Stratix III Floorplan

Stratix III series FPGAs include large I/O banks, internal memory blocks, PLLs and ALMs. ALM is the main logic component which handles logical operations of any type. The memory inside the FPGA is implemented by M9K and M144K memory blocks where M9K represents 9K bits; M144K represents 144K bits of bulk memory. DSP operations are handled by high-performance DSP processors. DLLs and PLLs are used to manage clocking within the FPGA. Below, the detailed explanation of each sub-component of Stratix III FPGAs is given:

A.1.1 Adaptive Logic Modules (ALMs)

Stratix III devices are based on the innovative adaptive logic modules that Altera introduced in 2004 with Stratix II FPGAs. ALMs are based on patented eight-input look-up table (LUT) with two dedicated adders and two registers. ALMs can implement six-input functions, some seven-input functions, or multiple other combinations of functions with varying numbers of inputs. The schematic drawing of ALM is given below:

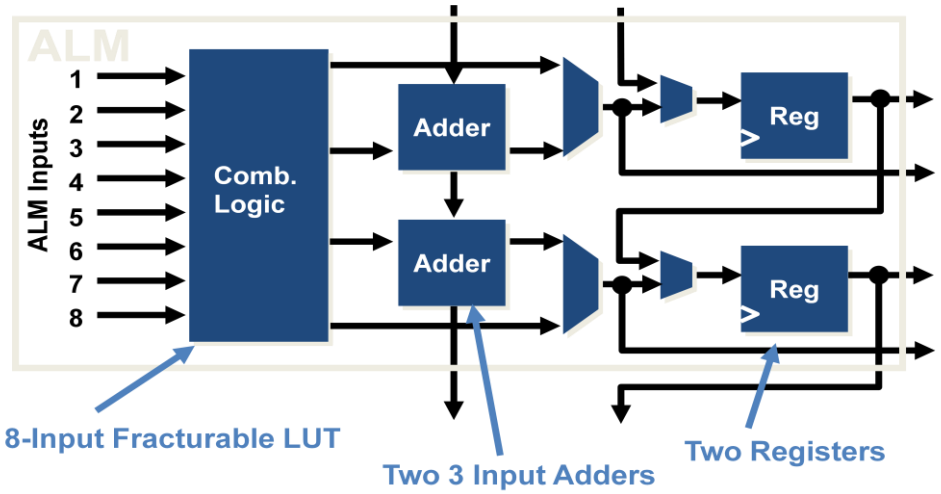


Figure 26 Infrastructure of ALM

A.1.2 Digital Signal Processing

For high-performance, silicon efficient programmable DSP functions, Stratix III FPGAs include DSP blocks with twice the multiplier capabilities of competing devices. Stratix III DSPs have variable bit-width multipliers, adders/accumulators, pipeline registers, and other arithmetic operations operating at up to 550 MHz. The internal structure of DSPs is given below:

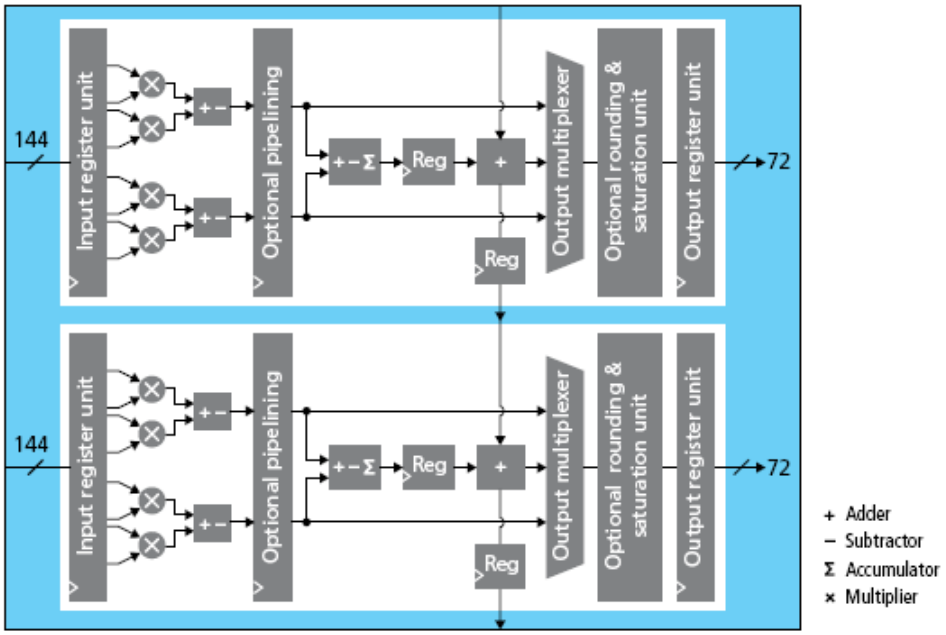


Figure 27 Architecture of DSP Block

A.1.3 TriMatrix Embedded Memory Blocks

TriMatrix embedded memory blocks provide three different sizes of embedded SRAM to efficiently address the needs of Stratix III FPGA designs. TriMatrix memory includes the following blocks:

- 640-bit MLAB blocks optimized to implement filter delay lines, small FIFO buffers and shift registers. MLABs are created by using 10 ALMs for greater memory bandwidth.
- 9-Kbit M9K blocks that can be used for general purpose memory applications.
- 144-Kbit M144K blocks that are used for processor code storage, packet and video frame buffering.

Each embedded memory block can be independently configured to be a single- or dual-port RAM, ROM, or shift register. Multiple blocks of the same type can also be

stitched together to produce larger memories. TriMatrix memory provides up to 16,272 Kbits of embedded SRAM at up to 600 MHz operation (39).

A.1.4 Clock Networks and PLLs

Stratix III delivers PLL resources with up to 12 PLLs per device and up to 10 outputs per PLL. Every output can be independently programmed creating a unique, customizable clock frequency with no fixed relation to any other input or output clock. Control over multiply, divide ratios and dynamic phase-shift reconfiguration help provide flexible designs. Stratix III device PLLs supports clock switchover, reconfigurable phase shift, PLL reconfiguration, and reconfigurable bandwidth. PLLs can be used for general-purpose clock management supporting multiplication, phase shifting, and programmable duty cycle. Stratix III PLLs also support external feedback mode and input clock tracking (39).

APPENDIX B

IMAGE FUSION SOFTWARE

B.1 Introduction

Image Fusion Software (IFS) is an image processing tool which can handle all the mentioned image fusion techniques throughout this thesis. It is used to evaluate and compare image fusion algorithms. It is developed on Matlab® and can run on version R2009b.

B.2 Features

- The IFS supports 18 different image fusion algorithm which are:
  i. This function implements the reference (32)
  ii. This function implements the reference (10)
  iii. This function implements the reference (12)
  iv. This function implements the reference (13)
  v. A Custom IF algorithm where the min value of the corresponding input pixels is assigned to channel R, the max value is assigned to channel G and the mean value is assigned to channel B.
  vi. This function implements the reference (40)
  vii. This function implements the *Proposed DCT Based IF*.
  viii. This function implements *Pixel Averaging Based IF*
  ix. This function implements *Select Maximum Based IF*
  x. This function implements *Select Minimum Based IF*
  xi. This function implements *Laplacian Pyramid Based IF*

xii. This function implements *FSD Pyramid Based IF*

xiii. This function implements *Ratio Pyramid Based IF*

xiv. This function implements *Contrast Pyramid Based IF*

xv. This function implements *Gradient Pyramid Based IF*

xvi. This function implements *Shift Variant Wavelet Based IF*

xvii. This function implements *Shift Invariant Wavelet Based IF*

xviii. This function implements *PCA Based IF*

xix. This function implements *Block-wise Shift Variant Based IF*

xx. This function implements *Jinshan Tang's IF*

xxi. This function implements *Zafar's IF*

- IFS can also save the output fused image to any location on hard disk for future use.
- The main GUI (Graphical User Interface) of IFS looks like:

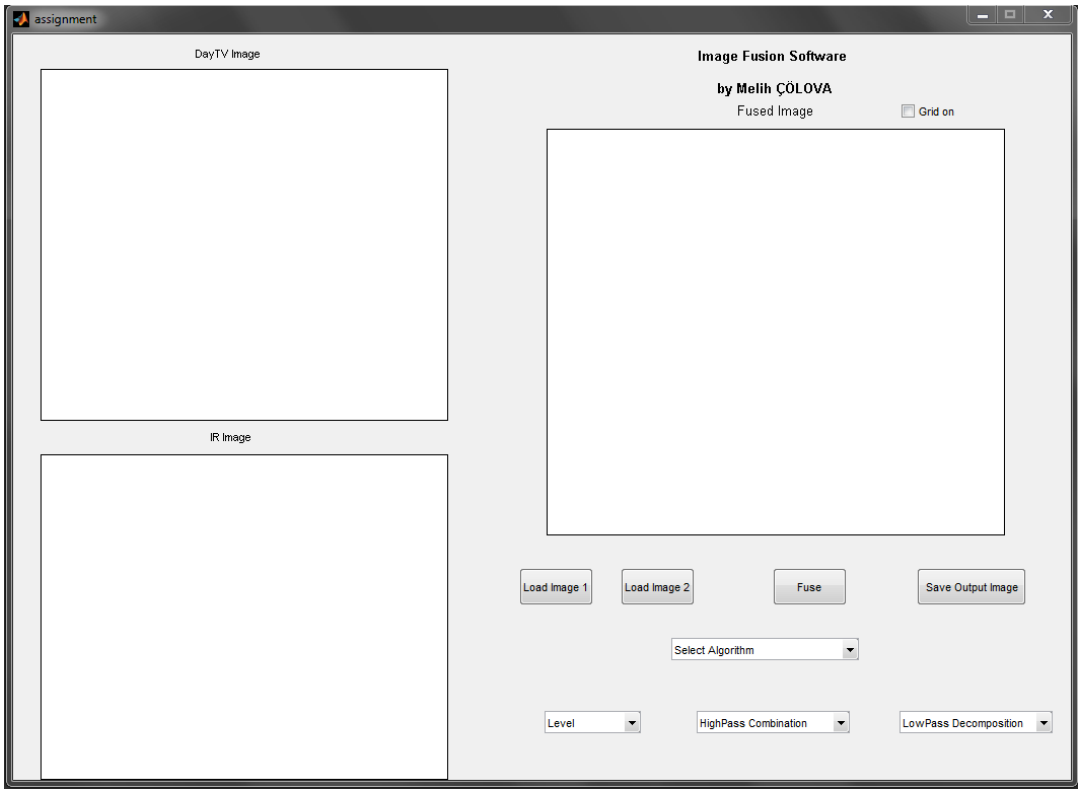


Figure 28 IFS GUI

B.3 Abilities

- IFS accepts two input images to be fused.
- Selectable algorithms out of 18 can be applied to the input images.
- When pyramid based or wavelet based algorithms enabled, the fusion level, high level fusion, low level fusion methods are become selectable.
- The fusion level option supports up to 7 layers.
- The high-pass combination has two options; the first one is *choose max* and the other one is *consistency check*. *Choose max* option simply chooses the max value among the corresponding high-pass layers of input images. *Consistency check* option uses a 3x3 window mask to suppress local discontinuities in the high-pass layer combinations.
- The low-pass combination has three options: The first option selects the first image as the basis image, the second option sets the second image as the base image and the third option sets the arithmetic average of two input images as the basis images. The first two options are useful when it is known that one of the input images has more information than the other. The last option performs well in most multi-layer image fusion algorithm without featuring or suppressing one of the input images.

B.4 Restrictions on Input Images

- The input images should be at the same size.
- In the sixth fusion algorithm, the first image must be RGB type whereas the second image is grayscale. This algorithm implements the paper (40) by Li et al and it is aimed for RGB type images.

APPENDIX C

VIDEO PROCESSING BOARD

The proposed DCT based algorithm is implemented on a VME form factor custom video processing board which has suitable interfaces for video applications and is a product of ASELSAN. The VPBD (Video Processing Board) host a Stratix III type FPGA with 80.000 logic elements. The front side picture of the VPBD is given below:

Figure 29 Video Processing Board

The schematic view of the board is drawn below:

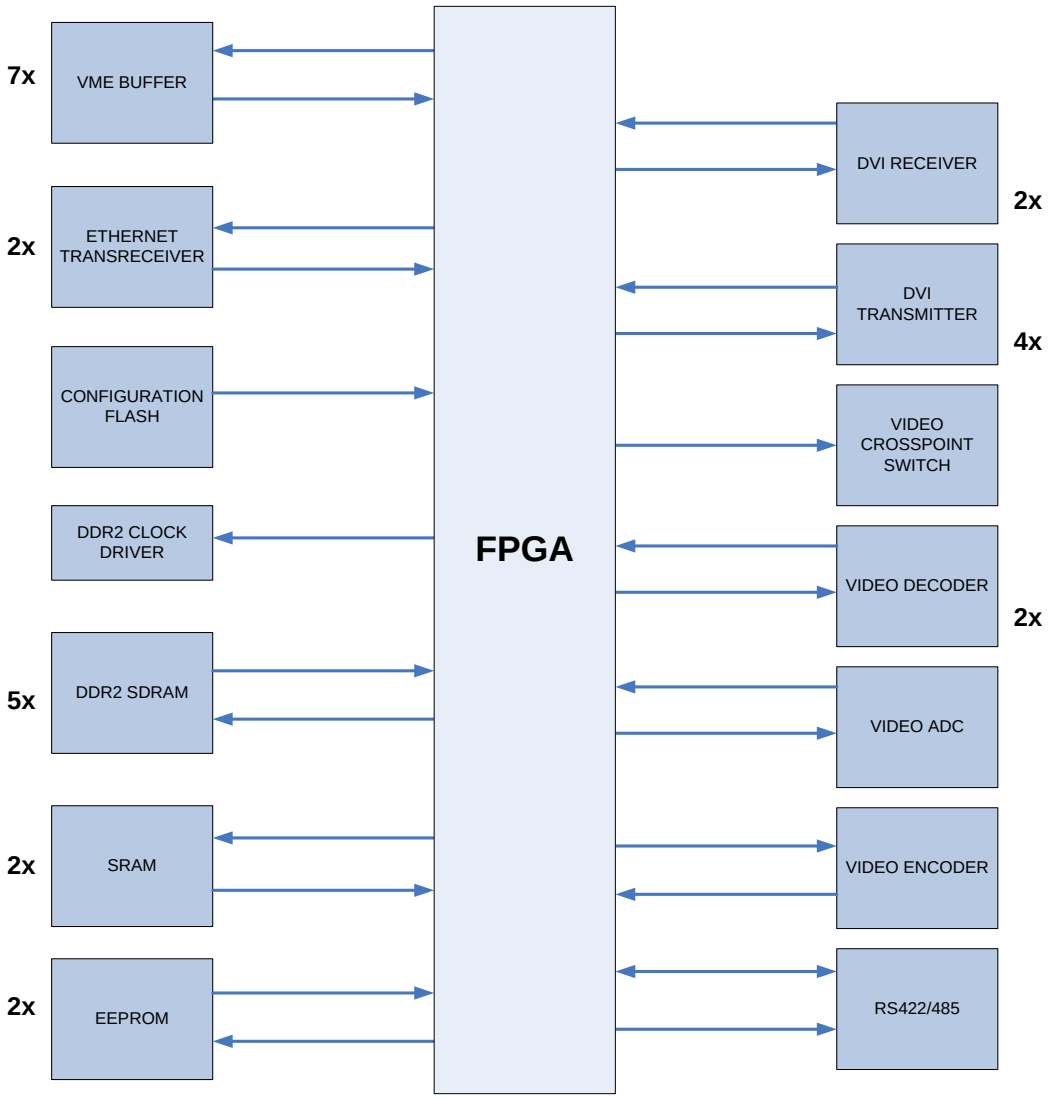


Figure 30 Schematic View of VPB

APPENDIX D

RESOURCE USAGE OF THE IMPLEMENTED DESIGN

Table 5 Resource Usage of the Implemented Design

Resource	Usage Ratio	Percentage
Combinational ALUTs	40,144 / 64,000	63 %
Memory ALUTs	0 / 32,000	0 %
Dedicated logic registers	20,834 / 64,000	33 %
Total registers	20858/N/A	N/A
Total pins	188 / 744	25 %
Total block memory bits	137,936 / 6,331,392	2 %
18-bit DSP blocks	528 / 672	79 %
Total PLLs	4 / 8	50 %
Total DLLs	0 / 4	0 %