2012

An Exploration of the Feasibility of FPGA Implementation of Face Recognition Using Eigenfaces

Vinod Anbalagan
University of Windsor

An Exploration of the Feasibility of FPGA Implementation of Face Recognition Using Eigenfaces

by

Vinod Anbalagan

A thesis

Submitted to the Faculty of Graduate Studies the Department of Electrical and Computer Engineering in Partial Fulfillment of the Requirements of the Degree of Master of Applied Science at the University of Windsor

Windsor, Ontario, Canada

2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Biometric identification has been a major force since 1990's. There are different types of approaches for it; one of the most significant approaches is face recognition. Over the past two decades, face recognition techniques have improved significantly, the main focus being the development of efficient algorithm. The state of art algorithms with good recognition rate are implemented using programming languages such as C++, JAVA and MATLAB, these requires a fast and computationally efficient hardware such as workstations.

If the face recognition algorithms could be written in a Hardware Description Language, they could be implemented in an FPGA. In this thesis we have choose the eigenfaces algorithm, since it is simple and very efficient, this algorithm is first solved analytically, and then the architecture is designed for FPGA implementation. We then develop the Verilog module for each of these modules and test their functionality using a Verilog Simulator and finally we discuss the feasibility of FPGA implementation.

Implementing the face recognition technology in an FPGA would mean that they would require relatively low power and the size is drastically reduced when compared to the workstations. They would also be much faster and efficient, since they are specifically designed for face recognition.

Acknowledgments

There are several people who deserve my sincere thanks for their generous contributions to this project. First and foremost I would like to express my deepest gratitude and appreciation to Dr. Jonathan Wu and Dr. Mohammad Khalid, my supervisors for their invaluable guidance, encouragement and support over the years. Their generosity, wisdom, empathy and professionalism helped me complete this thesis, would also like to thank Dr. Tepe and Dr. Zhang for their invaluable advice.

Secondly, I would like to express my gratitude to Dr. Nihar Biswas, who personally helped me overcome my personal difficulties. Also I am very thankful to Mrs. Andria Ballo, for always being there for me and for all the engineering souls in distress, assisting them with her guidance, support and friendly approach.

Next, I would like to thank my senior and friend Mohan Thangarajah, for constantly encouraging me and helping me whenever I was in need. His company through the university years would be cherished, especially our 7/11 and Tim Hortins coffee breaks whilst staying up all night for working on our projects in our office, lab and library.

Finally, I would like to thank my parents A. Premalatha and R. Anbalagan, my brother A.Vivek, for all their understanding, support and sacrifices over the past couple of years, and this note would be incomplete without thanking my family

Fifi Dyne, Lorena Marie, James O. Lepp, Steven O. Lepp, Jeni Robertson, Ricki Oltean, Kirsta and Chris Konrad, Gina, Chelsea and Gerry Lepp, your support, motivation and constant encouragement and has helped me immensely to complete this project.

To my family and friends for their unending love and support.

Contents

List of Figures

List of Tables

List of Abbreviations

The notation used in this thesis is as follows. In general, a scalar element of a vector is denoted as $v_i \in v$, Which is read as " the i^{th} element of vector v ", with indexing of vectors starting at i = 1 and ranging to the length of the vector. All vectors are assumed to be column vectors. Some commonly used operators, symbols and abbreviations are listed below.

Abbreviations	Definition
$(\bullet)^T$	Transpose Operator
$N \times N$	Height (N) and Width (N) of an Image
$\bar{x}$	Mean of x
I	Identity Matrix
D	Dimension
SD	Standard Deviation
V	Variance
Cov	Covariance
C	Covariance Matrix in Eigenfaces Algorithm
L	Dimensionality Reduced Covariance of Eigenfaces Algorithm
u	Eigenvector of Covariance Matrix 'C'
v	Eigenvector of Covariance Matrix 'L'
λ	Eigenvalue
Ω	Weight Vector
ε_{rec}	Smallest Euclidean Distance
θ_{rec}	Heuristically Chosen Value
Γ	An Image in the Database
Ψ	Average Face Vector

Φ	Difference face vector
M	Total Number of Images in the Database
K	Top Most Significant Eigenfaces
MATLAB	Matrix Laboratory
HDL	Hardware Description Language
FPGA	Field Programmable Gate Array
ID	Identification
E-Passport	Electronic Passport
DOD	Department of Defense
MIT	Massachusetts Institute of Technology
FERET	Face Recognition Technology
DARPA	Defense Advanced Research Product Agency
FVRT	Face Recognition Vendor Test
PCA	Principal Component Analysis
LDA	Linear Discriminant Analysis
ICA	Independent Component Analysis
FBG	Face Bunch Graphing
AAM	Active Appearance Model
HMM	Hidden Markov Model
FRGC	Face Recognition Grand Challenge
AT&T	American Telephone & Telegraph
CAS-PEAL	Chinese Academy of Science – Pose, Expression, Accessory and Lighting
PICS	Psychological Image Collections
RMA	Royal Military Academy
RAM	Random Access Memory
LSB	Least Significant Bit
MSB	Most Significant Bit
EA	Eigenfaces Algorithm
IC	Integrated Circuit

Chapter 1

Introduction

1.1 Face Recognition

Face recognition in humans has always been an enigma, for we can recognize a face with facial hair, occlusion, even after several years of separation and in worst possible lighting conditions. For two decades now researchers, not only from the field of computer vision but also psychologists and neurologists, have been trying to emulate this ability using machines, so far we only know that the temporal lobe in the brain is partly responsible for face recognition, damage to this region would result in a person losing his ability to recognize faces, and this condition is called prosopagnosia. Even after this condition has occurred, the perception of face remains unchanged because the human mind would use its hearing ability and cognitive ability to analyze the voice and gait of a person for recognition, so emulating such an ability will be an herculean task even with the latest available technology. This challenge is one reason why face recognition has caught the imagination of so many researchers from diverse fields.

1.2 Eigenfaces

The concept of eigenfaces is similar to Fourier decomposition, which is one of the most fundamental ideas in mathematics and signal processing. A Fourier series decomposes periodic signals into sum of (possibly infinite) set of simple oscillating functions namely sine and cosines.

Eigenfaces technique represents a face as linear composition of the base images also known as 'eigenfaces' or 'eigenpictures', these eigenfaces are basically a set of eigenvectors used in computer vision problem, they are generated by performing a mathematical procedure called 'Principal Component Analysis' [Joll 02] on a large set of human face images. In mathematical terms, we are finding the eigenvectors of the covariance matrix of a set of faces, where these faces are treated as a vector in a high dimensional space. The characteristic features from each face contributes to the eigenvectors, which can be represented as a ghostly face as show in Figure 1.1, we call this as an eigenface.

Figure 1.1: Eigenface

The approach of using eigenfaces for recognition was developed by Sirovich and Kirby (1987) at Brown University [Kirb 87], later was expanded and developed by Matthew Turk and Alex Pentland [Turk 91] [Pent 91], which was considered as the first successful model of automated face recognition technology.

1.3 Problem Statement

Face recognition techniques have improved significantly. The focus of face recognition has been to develop the most efficient algorithm; researchers have been striving to develop this elusive algorithm with highest recognition rate. Face recognition algorithms require computationally efficient and fast hardware with high storage capabilities such as mainframes, workstations and server desktop computers.

If we deploy face recognition technology; we would require the best and efficient workstations stationed at every entry points under human supervision, this could be very costly and is not feasible in places like a huge organizations, storage facilities, multistoried parking areas, residential complexes, warehouses, etc. The majority of the areas would go uncovered and vulnerable since they would have many entry points. In this thesis we explore the feasibility of implementing face recognition technology in a FPGA, which would drastically reduce the size and would only require relatively low power without compromising on recognition rate or speed. The reduction in size would imply that it could be used in places where we would normally hesitate to use a workstation or a server.

1.4 Objectives

The work presented in this thesis has the following objectives.

1. Investigate new and existing algorithms and choosing a computationally efficient, simple and accurate method.
2. Propose a feasible architecture for FPGA implementation based on the chosen algorithm.
3. Developing Verilog HDL module for each individual module and test its functionality using a Verilog simulator.

1.5 Motivation

Biometric signature is very distinct in verifying identities of an individual; they cannot be guessed, stolen, forged or lost.

Biometric identities are derived from physiological characteristics such as face, fingerprint, finger geometry, hand geometry, iris, palm, vein, retina and voice. Behavioral traits such as gait, signature and keystroke dynamics can also be used in establishing biometric identity.

Face recognition seems to offer several advantages over other biometric methods; also face recognition can be done passively, where the subject need not even raise their finger, but the face recognition technologies [Cogn 10] [Ayon 10] [Auro 08] that are available now, require computationally efficient workstations and servers, since the algorithm used are very complex and computationally intensive, along with the power requirements and lack of mobility are a huge drawbacks even though they offer good

recognition rate. This hurdle could be crossed if we could implement face recognition algorithm on an FPGA.

Currently face recognition algorithms are implemented programming languages such as C++, JAVA, MATLAB, Python and Mathematica. They are yet to be written in a Hardware Description Language (HDL). This thesis aims at exploring the feasibility of FPGA implementation of face recognition using the best and the most efficient algorithm implemented using Verilog HDL.

1.6 Thesis Organization

Developing from the introduction in Chapter 1, Chapter 2 summarizes the biometric systems, types of biometric system, applications and advantages of biometric system.

Chapter 3 covers face recognition technology's development though history, face detection, extraction and classification process, popular face recognition approaches, the difficulties involved in implementing face recognition technology and the available database for face recognition.

Chapter 4 of this thesis provides a thorough background, mathematical conceptualization and algorithm of principal component analysis and eigenfaces. The chapter continues with a discussion on the assumptions and the steps involved in the algorithm by presenting a detailed analysis of eigenfaces calculations using a small example.

Chapter 5 builds on the foundation laid in Chapter 4; it proposes a flexible architecture for implementing eigenfaces, each and every module from the architecture along with their functionality are discussed in detail.

Chapter 6 presents the ModelSim Simulation results of the functionality of the architectural blocks and discusses the feasibility of FPGA Implementation and related issues.

We conclude in Chapter 7 with a discussion of future work.

Chapter 2

Biometric Recognition

2.1 Introduction

Biometrics consists of methods for uniquely identifying individuals based on physical and behavioral traits. This Chapter begins by discussing the history of biometrics and the need for biometrics. We also discuss the different types of biometrics and finally the advantages and applications of biometrics.

2.2 What is Biometrics?

The term "Biometrics" is derived from the Greek word "Bio" (life) and "metrics" (to measure). Automated biometric system [Coun 06] has only been available over the last few decades due to significant advances in technology.

Biometric recognition, however fancy the name sounds, was conceived before thousands of years ago. In the earliest civilizations the cave walls were said to be adorned with paintings and alongside these paintings there were numerous handprints, which were believed to be tamper proof signature, by its creators. Face recognition was used by the early civilizations to categorize an individual between known and unknown. Human to human recognition kept evolving, the mind was unconsciously registering the behavior

patterns such as voice and gait. These physiological and behavioral patterns recognition were collectively known as biometric recognition.

2.3 Need for Biometric Systems

During the mid 1800s there was rapid growth of cities due to industrial revolution. As more and more people were migrating towards the cities, unrest and chaos was a common scene, this made it more and more difficult for the justice system, to keep track of the repeated offenders. They used a formal system that recorded the offences along with the identity trait of the offender; this marked the birth of the official biometric system.

Personal security is being treated with utmost importance these days and rightly so, since the identity thefts are on rise. Unconventional recognition techniques such as password and ID card are based on "what you know?" and "what you have?" In contrast Biometric technology is based on "who you are?" It is derived from physiological traits such as face, fingerprint, iris, palm-print, voice and behavioral traits such as signature, gait and keystroke. This technology is extremely difficult to duplicate, steal, copy, misplace or forge.

Since the advent of Internet, everything has gone online, including our day-to-day activities such as online banking, social networking, online shopping etc. All our online activities start with logging in and logging off the network with our user ID and password. These could be easily hacked, stolen and guessed. Once this information falls into the wrong hands, they could access your bank account; they could get your personal information such as residential address etc. This poses a very serious security threat for

anyone. Therefore to avoid such situations a good solution is to implement an effective biometric system.

2.4 Types of Biometric Recognition

There were two approaches to the early biometric system, the first approach was called as Bertillon System of Measuring Various Body Dimensions, and this originated in France. These measurements were written on cards that were sorted by height, weight, arm's length etc., this field was called anthropometrics. The second approach was the method of taking fingerprint from the index finger; this was introduced in Asia, Europe and South America by 1800s. This method was based on the ridges and the finger print pattern on the index finger.

The later half of the 20th century saw the much more advanced phase of biometric system, which was helped by development in Computer Systems and Technology, the biometric system could be categorized [Savv 11] into two groups, Physiological characteristic and Behavioral characteristics, different methodologies have been introduced based on these two categories shown in Figure 2.1

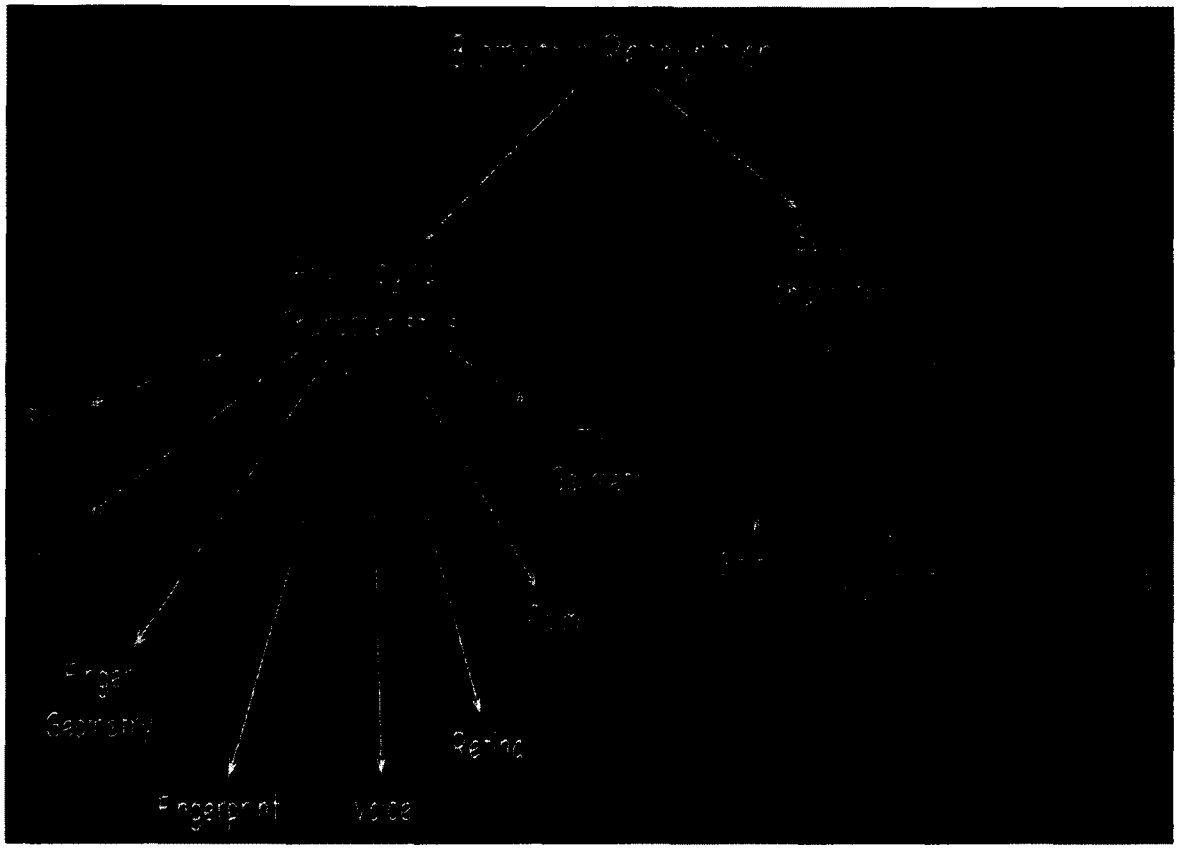

Figure 2.1: Types of Biometric Recognition

2.5 Applications of Biometric Systems

The following are some of the applications of biometric systems.

1. Biometric Fingerprint Identification Systems are widely used in forensics for criminal identification.
2. Biometrics are widely used for Physical Access Control
3. Logging into Computers
4. Welfare Disbursement
5. National ID cards and International Border Crossing
6. Keyless ignition in Automobiles

7. To verify customers during transactions via telephone and Internet.
8. E-passports are a work in progress for issue in near future, which has an embedded chip containing the holder's facial image and other traits.

2.6 Advantages of Biometric Systems

The following are some of the advantages of biometric systems

Uniqueness:

It is impossible for two people to share the same biometric data, so biometric systems are designed around an individual and unique characteristic.

Cannot be Lost:

A Biometric data could never be lost, unless the individual is involved in a terrible accident.

Cannot be Copied or Guessed:

Biometric data cannot be forged or shared or guessed, since the biometric data are physiological attributes.

The fact that biometric system needs "You" to authenticate that the subject is you is the advantage of this system.

Chapter 3

Face Recognition

3.1 Introduction

Face recognition is a form of biometric identification, which uses facial features as the basis for identification. This chapter covers face recognition, its development through history and the different areas of application. It also talks about the steps involved in face recognition; such as face detection, feature extraction and face classification. Finally, it describes the different types of approaches for face recognition, list of available face databases and the difficulties in face recognition.

3.2 Primary Tasks of Face Recognition

Face recognition is used for two primary tasks, which are as follows

Verification: (one to one matching)

When an individual presents an identity, the system verifies whether the individual is who he claims to be.

Identification: (one to many matching)

If an image of an unknown individual should be identified, the system verifies the image with other images in the database to establish the identity.

3.3 Disadvantages of Other Biometric Systems:

When using other biometric systems, there are a few disadvantages, which are as follows

1. Finger print of people working in chemical industry can be affected
2. Voice recognition system would fail when a person has a sore throat and also voice of a person would change with age, which would complicate the system.
3. For people with diabetes, the eyes would get affected resulting in failure to authenticate, also iris recognition is very costly to implement.
4. Digital Signatures could be modified or forged.

The above disadvantages could be overcome by using face recognition method with an effective algorithm.

3.4 Advantages of Face Recognition System:

Face recognition system is advantageous over other biometric systems; some of the advantages are as follows,

1. Almost all of the biometric systems require the user to perform an action like placing their hand or finger for finger print reading, speaking into the microphone for voice recognizer etc. However face recognition does not require any explicit action from the user.
2. Face recognition technology is cheaper when compared to other biometric systems.
3. It is non intrusive

4. Face recognition system does not cause any health risk to the user, whereas other biometric technology that requires multiple users to use the same equipment can potentially expose them to germs from previous users.

3.5 Evolution of Face Recognition

Face recognition technology has been extensively researched over the years. Researchers are still working on algorithms that can provide high accuracy and portability. Face recognition is a challenging task because of factors like change in expression, scale, location, occlusion, pose and lighting conditions.

Many algorithms have evolved over the years [Coun 06] The Earliest Work on Face Recognition was from the Field of Psychology during the 1950s. "The perception of people, a handbook of social psychology" [Tagi 54] by J.S Bruner and R. Tagiuri. Engineer's interest in face recognition resulted in the first semi automated face recognition system during the 1960s. Woodrow W. Bledose and other researchers developed the first semi automated face recognition system at Panoramic Research Inc., in Palo Alto, California [Bled 66]; the US Department of Defense (DOD) funded this work. This system required human interference to locate the feature points such as eyes, nose and mouth in photographs and the distance and ratios are calculated so that it can be later compared to test image for recognition.

During the 1970s face recognition moved forward from semi automation. A. J. Goldstein, Leon D. Harmon and Ann B. Lesk's [Gold 71] research in Bell Laboratories described a vector, containing 21 specific features such as lip thickness, ear protrusion, nose length etc., to recognize faces, nevertheless they were all manually measured and

compared. In 1973 face recognition with template matching was introduced. Martin A. Fischler and Robert A. Elschlager [Fisc 73] measured the similar features as in the earlier papers but they made it automatically, they described an algorithm that used local feature template matching approach. In the same year Kanade developed the First Fully Automated Face Recognition system [Kana 73]. Kanade used pattern classification technique to match test faces to a known set of faces, this was a purely statistical approach. Template matching technique was improved during the 1980s. Mark Nixon's [Agua 02] Eye spacing measurement improved template-matching approach by introducing 'deformable templates'.

The first semi-automatic facial recognition system was deployed during 1988. The Lakewood division of Los Angeles county sheriff's department began using composite drawings of suspects to conduct a Mug shot database search using this system. In the same year eigenfaces technique was developed for face recognition. L. Sirovich and M. Kirby [Kirb 90] applied Principal Component Analysis, a linear algebra technique on face images, they represented an image in a lower dimension as principal component vectors without losing much information, and then reconstruction them.

In 1991 face detection technique was mastered, making real time face Recognition was possible, Matthew Turk and Alex Pentland of MIT [Turk 91] [Pent 91] extended the work on eigenfaces technique and made this a state of art face recognition technique; this was the first successfully available industrial application, this paved way for a new era in Face recognition systems. In 1993 FacE REcognition Technology (FERET) program was initiated. The Defense Advanced Research Products Agency (DARPA) and Department of Defense (DoD) sponsored the FacE REcognition

Technology (FERET) [Phil 97] in an effort to develop face recognition algorithm and technology to commercialize the product. Face Detection based on Neural Networks was introduced in 1988. Henry A. Rowley, Shumeet Baluja and Takeo Kanade [Rowl 98] came up with face detection technique using Neural Networks.

Face Recognition using Elastic Bunch Graph Method was introduced in 1999, Laurenz Wiskott, Jean-Mark Fellous, Norbert Kruger, Christoph Von Der Malsburg [Wisk 97] presented a system for recognizing human faces using Gabor wavelets transform on images. A face graph is created from an image, it consists of sparse collection of jets at the edges where eyes, nose mouth are located, the face bunch graph has a stack like structure and combines graphs of individual sample faces. Comparing the similarities between the graphs can recognize a new face.

In 2000 First Face Recognition Vendor Test was held. Multiple US Government agencies sponsored the Face Recognition Vendor Test (FVRT) [Phil 03], this served as the first open large-scale technology evaluation of multiple commercially available biometric systems.

In the following decade face recognition systems has seen several changes and is being Sponsored and promoted by many government and private organizations.

3.6 Applications of Face Recognition

There are numerous areas where face recognition could be employed; a few are outlined below.

1. *Criminal Justice system* – Mug shot database, witness face reconstruction, video surveillance and Forensics reconstruction of face from remains.
2. *Network security* – User authentication, database access, e-commerce and online banking.
3. *National Security* – National IDs, Voter Registration, Border Crossing, etc.,
4. *Personal Security* – Home Video Surveillance, Driver Monitor system.
5. *Access Control* – Access control in areas like Warehouse, Seaports and Airports.
6. *Entertainment* – PlayStation, Digital cameras, etc.

3.7 Face Detection

Facial Recognition System [Grgi 07] is a whole package that consists of steps such as face detection, feature extraction and face classification. Figure 3.1 illustrates the steps involved a face recognition system

Figure 3.1: Steps in Face Recognition System

The first step in any face recognition system is the detection of faces in images, since the image might have multiple faces or structures similar to faces, but nowadays most face

recognition algorithm would not require to detect faces in images, since the images are normalized and fit to size according to the need of the algorithm, incase if the image is too complex and is not normalized then we might have to detect the face from the image, these images are mostly taken under uncontrolled environments, the following are the factors that challenge face detection

1. Pose variation
2. Facial Expression
3. Background Environment
4. Occlusion

3.7.1 Face Detection Scenarios

There are two basic scenarios in face detection, first is when an image is taken under controlled condition; the face is detected using edge detection technique. Second scenario is when an image is taken under uncontrolled condition [Leun 95], if it is a color image [Naka 96] then the skin color [Kend 96] could be used to identify the face, incase if it's a grey scale image then the position of features like eyes, nose and mouth could be identified in order to detect the face.

3.7.2 Face Detection Methods

Face detection methods [Ahuj 02] are detected in to four categories, which are as follows.

3.7.2.1 Knowledge Based Method

In knowledge-based method [Teka 98] we try to apply a set of rules, which are derived from our knowledge of faces, some of the rules are, a face usually has two symmetrical eyes, the distance between the eyes, the color difference between the cheeks and the area under the eyes, etc., while making these rules we have to make sure that they are not too vague (generalized), if they are then there would be many false positives, on the other hand false negatives would be generated if the rules are too fine (detailed). This method of face detection has its own limitations and the detection rate depends on the rules applied for this method.

3.7.2.2 Template Matching

Template matching [Pogg 92] method tries to define a face as a function; each feature such as eyes, nose, mouth and ears can be defined independently in a face. Face contour and relationship between different templates are identified as patterns, these standard patterns are compared to images to detect faces, this type of detection is very simple to implement, but these methods are limited to faces that are frontal and un-occluded, variation in shape and pose would result in poor recognition rate.

3.7.2.3 Appearance Based Method

Appearance based/ View based [Lew 96] [Pogg 98] method rely on techniques from statistical analysis and machine learning methods, Appearance based method is said to have a probabilistic nature, such that it finds if an image vector would belong to a face or not, it depends on the discerning ability to identify a face class from a non face class.

Comparing with the other two face detection methods, appearance based method has higher detection rate, few tools [Mitc 96] that are based on appearance-based methods [Turk 91] [Pent 91] [Sama 93] [Guo 00] [Phil 99] [Osun 97] [Sebe 02] are listed below

1. Eigenfaces
2. Neural Networks
3. Hidden Markov Models
4. Naive Bayes Classifier
5. Support vector machines

3.8 Feature Extraction

Facial features [Craw 06] [Yow 97] are the essence of a face, for they make a face distinct from one another, many face recognition algorithms incorporate feature extraction, this must be optimized so that it takes much less memory and has reduced computation time.

An input face image is reduced to a feature set; this feature set is reduced to a subset by discarding the non-relevant features and choosing the best of the feature set, the following are few methods [Rowl 98] of feature extraction.

1. Generic method based on lines, curves and edges
2. Principal Component Analysis
3. Template based method
4. Neural Network based method
5. Self Organizing Maps

The number of features that are to be extracted from a face should be carefully chosen, if it is too low, this might lead to loss in accuracy, if it is too high, then it might result in more false positives and might take more memory and processing time.

3.9 Face Classification

Face classification is the step that follows feature extraction; classifiers when used in combination with other classifiers outperform individual classifiers.

The basic classifier [Jain 00] is the one that classifies face based on similarity, it classifies the similar class from the non-similar class, and an example of such classifier is Euclidean Distance Classifier [Mitc 96], some feature extractors could also be used as a classifier, following are the list of few of the classifiers

1. Euclidean Distance Classifier
2. Vector Quantization
3. Self Organizing Maps
4. Template matching

Since a classifier could be used in combination with other classifiers [Roli 01] [Kitt 98], we can use a classifier to recognize the eyes, another classifier to recognize the nose and another classifier to recognize the mouth, all these classifiers could be combined [Tuly 08] [Wang 03] [Heis 03] for an effective classification, these combinations could be divided into three types

1. *Parallel* – All the classifiers are executed independently, and then they are finally combined together.
2. *Serial* – Classifiers run one after another, where each classifier would refine the previous classifier result.
3. *Hierarchical* – Classifiers are arranged in a tree like structure.

Choosing the best classifier impacts the processing speed and the accuracy, choosing a very simple classifier would produce less accurate but a quicker result, whereas choosing a complex classifier would produce a more accurate result but it takes more processing time, so its important to strike the right balance to choose the best classifier.

3.10 Different Approaches in Face Recognition

Face Recognition methods [Arab 09] [Rose 03] evolved over time, it can be seen as a process, which includes many steps. These steps could overlap or change their order to best suit the application. This makes it hard to definitively categorize the approaches of face recognition, but still they could be generally categorized [Lu 03] [Marq 10] [Tolb 06] as shown in Figure 3.2

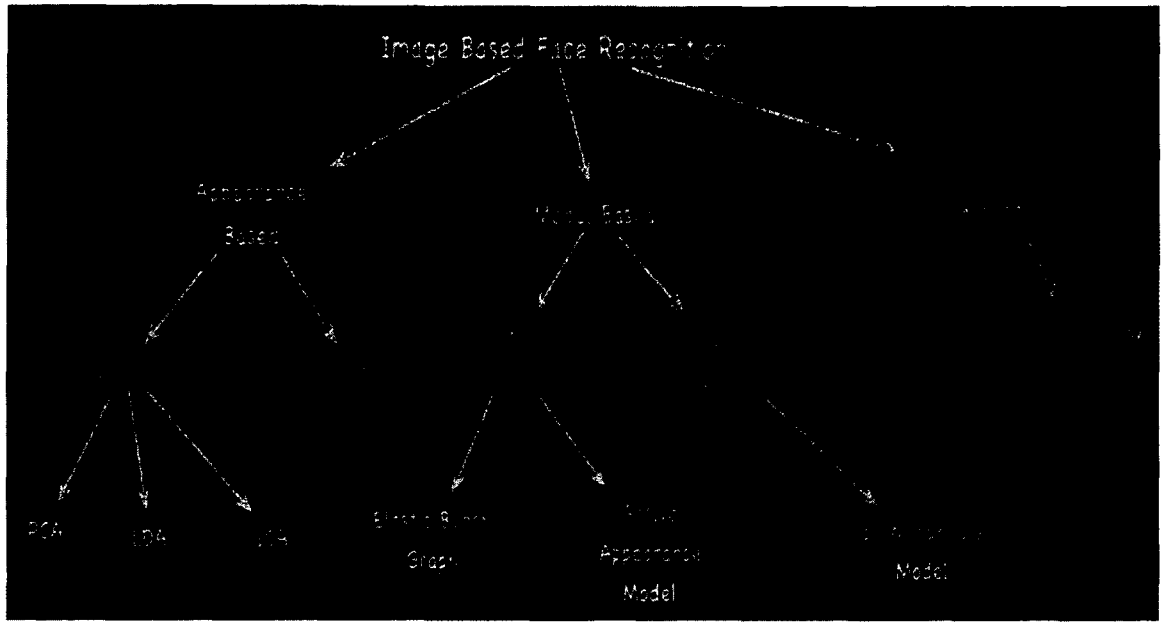

Figure 3.2: Face Recognition - Approaches

3.10.1 Appearance based Approach

An Image is considered to be a point in high dimensional vector space; an appearance based or view based approach uses statistical technique to analyze the distribution of image vector in the vector space and classifies the essential features for efficient recognition. The Appearance based method can be divided into Linear and Non-Linear Analysis.

3.10.1.1 Linear Analysis

Three of the widely used linear analysis classifications are

1. Principal Component Analysis (PCA)
2. Linear Discriminant Analysis (LDA)

3. Independent Component Analysis (ICA)

3.10.1.1.1 Principal Component Analysis

PCA is one of the most widely used classifiers; it is based on Karhunen-Loeve transformation [Kirb 87] [Pent 97] [Turk 01]. PCA performs a dimensionality reduction by extracting the principal components of high dimensional data, these principal component vectors defines the face space, which is a subspace in the image space.

The face images are projected onto the face space and their weights are identified, and the test image is projected onto the face space, the weight coefficient of the test face image is compared with the weights of the face images from the database, using a distance classifier will give us the closest possible match to the test face.

3.10.1.1.2 Linear Discriminant Analysis

Linear Discriminant Analysis [Belh 97] finds the vector in the underlying space that best discriminates the classes, for all samples of all classes the between-class scatter matrix and within-class scatter matrix are defined.

LDA is closely related to PCA, it explicitly attempts to model the difference between the classes of data; PCA on the other hand does not take into account any difference in class.

3.10.1.1.3 Independent Component Analysis

Independent Component Analysis [Bart 02] [Como 92] [Liu 99] minimizes both second order and higher order dependencies in the input data and attempts to find the basis along which the data are statistically independent. ICA is generalization of PCA.

3.10.1.2 Non Linear Analysis

The face manifold in subspace need not be linear; kernel methods are a generalization of linear methods.

Linear analysis are not very sensitive to relationships among multiple pixels in an image, to extract the non linear features of the image linear analysis methods was extended to non linear analysis [Yang 02] such as Kernel PCA [Schö 98] [Zhou 04], Kernel ICA [Jord 02] and Kernel LDA.

3.10.2 Model based Approach

A model of human face is constructed to capture the features, facial variations and texture of a face.

Prior knowledge of facial features are used to construct a model, a model based approach [Lani 95] derives distance and relative positions from the placement of facial elements such as eyes, nose, ears and mouth, a constructed model is often called as Morphable Model, a model based approach is divided into two types

1. 2D Approach
2. 3D Approach

3.10.2.1 2D Approach

Two-dimensional approach can be divided into two categories

1. Elastic Bunch Graphing
2. Active Appearance Model

3.10.2.1.1 Elastic Bunch Graphing

A face is represented as a graph, considering the fact that all human faces have the similar topographical structure, the graph is constructed with nodes positioned at eyes, nose edge, mouth, etc., these positions are called as fiducial points, the edges are labeled with a 2D distance vector, with these vectors a face graph is constructed.

The face bunch graph has a stack like structure and it combines graphs of individual sample faces, it is crucial that the individual graphs all have the same structure and that the nodes refer to the same fiducial points.

A jet is a condense and robust representation of a local grey value distribution, it is based on Gabor Wavelet Transform, which is a convolution with a family of complex gabor wavelets having the shape of plane waves restricted by Gaussian envelope function. All jets referring to the same fiducial points, for example, all the right eye jets are bundled together in a bunch, the right eye bunch might contain a male eye, a female eye, both closed and open etc., from which we can select any jet as an alternate description. To recognize a new face by elastic bunch graph matching [Wisk 97] [Kela 06], the fiducial points are positioned so as to extract a graph, after the nodes have been located on the new face, the face can be recognized by comparing the similarities

between the graph of this face and the graph of every face stored in Face Bunch Graph (FBG).

3.10.2.1.2 Active Appearance Model

An Active Appearance Model (AAM) [Tayl 01] [Walk 00] is an integrated statistical model, which combines a model of shape variation with a model of the appearance variation in a shape-normalized frame.

The Active Appearance Model is constructed based on a set of labeled images, where landmark points are marked on each example face at key positions to describe the facial features, models are combined together by using linear analysis method such as PCA. Matching to an image involves finding model parameters; AAM fitting is applied to seek a set of model parameters, which minimize the differences between the image and a synthesized model example projected into the image.

3.10.2.2 3D Approach

Human face is a surface lying in the 3D space, thus a 3D model is more suitable for representing faces. Once such method based on 3D approach [Zhan 09] [Bron 04] is 3D Morphable Model.

3.10.2.2.1 3D Morphable Models

3D models have stronger ability to minimize the problems of head, pose and illumination, a 3D Morphable model [Vett 03] is extended from 2D Morphable Model.

The Morphable face model is based on a vector space representation of faces, which is constructed such that any convex combination of shape and texture vectors of a set of examples describes a realistic human face.

3.10.3 Piecemeal/Holistic Approach

Faces can be identified with minimal information; some algorithms would require only the independent information for face recognition unlike other algorithms that uses the whole face or the relationship between individual features and the face. Early researchers tried to use very little but relevant features [Mals 92] for face recognition. Although feature processing is important, relation between features is also important. This is one of the reasons why most face recognition follow holistic approach, one such model that is based on holistic approach [Nixo 85] is Hidden Markov Model (HMM)

3.10.3.1 Hidden Markov Model

Hidden Markov Models [Sama 93] [Nefi 98] [Raja 98] are a set of statistical models used to characterize the statistical properties of a signal. Faces are intuitively divided into regions such as eyes, nose, mouth etc., these regions can be associated with the states of a Hidden Markov Model, since HMMs require a one dimensional observation sequence and images are two dimensional, the images should be converted into one dimension before associating the states.

3.11 Face Recognition Database

Face recognition algorithms keeps on evolving, the best way to test and benchmark an algorithm is to use a standard test data set, there are many standard databases [Dela 11] [Grou 97] available, we choose the one that suits our application, in our thesis we have faces from AT&T database [Camb 02] for the figures, a list of available face database is as follows

1. The FacE REcognition Technology Database (FERET)
2. Face Recognition Grand Challenger Database (FRGC)
3. AT&T Database of Faces
4. The Yale Face Database
5. CAS –PEAL Face Database
6. BioID Face Database
7. Psychological Image Collection at Stirling (PICS)
8. 3D_RMA Database
9. Texas 3D Face Recognition Database
10. Natural Visible and Infrared Facial Expression Database

3.12 Difficulties in Face Recognition

Face Recognition involves more than one dimension, and there could be many faces in an image and there is also the structures that resemble faces, along with this we have to take the external conditions into account, the external conditions account for noise when we project the image in an low dimensional space, all these conditions makes face recognition more difficult, a list of roadblocks for face recognition is listed below.

1. *Lighting* – Difference in lighting conditions could cause error in recognition. This could be avoided to an extent by using a standard grey scale image. But this might not be of any help for algorithms that works with color images.
2. *Pose & Expression* – The orientation of the head and the expression can affect the recognition rate, this could be avoided by having multiple images for a single person with different poses and expression
3. *Occlusion* – Facial hair, glasses, headgear could occlude the face resulting in poor recognition rate.
4. *Ageing Problem* – A face would undergo major changes with time, especially during the age group of 10-25 years and also during 40-50 years, this affects the accuracy of the algorithm, and this could be avoided by constantly updating the database with the latest face images.
5. *Image Quality* – The images used for the database should be of a good quality. The best result could be obtained, if the background of the image could be cropped and the image is fit to size as per the requirements of the algorithm.

Chapter 4

Eigenfaces

4.1 Introduction

This chapter discusses Principal Component Analysis (PCA) and the related mathematical concepts. It then proceeds with an example calculation that clearly illustrates the concept of PCA and its role in face recognition. It also thoroughly examines eigenfaces approach and its procedure. Lastly, face space is defined and different possible cases of where an image could lie in the face space are discussed.

4.2 Principal Component Analysis

Invented by Karl Pearson in 1901, PCA is a powerful tool for analyzing data. It is considered as one of the most valuable tools used in mathematics and computer vision. PCA is widely used as a tool in exploratory data analysis and for making predictive models. It is very simple and has a non-parametric method of extracting relevant information from complex datasets.

PCA is the simplest of the true eigenvector based tools for multivariate analysis. It has the ability to reveal the internal structure of the data in a way that best explains the variance of the data. When a dataset with multiple variables with co-ordinates in multi

dimensional space is given to PCA, it can show us the equivalent lower dimensional picture that is easier to understand.

PCA is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original values, so we can say that PCA is a statistical method for reducing the dimensionality of a dataset while retaining the majority of the variations present in the dataset [Joll 02].

4.3 Background Mathematics

To understand PCA [Smit 02] [Shle 05] [Rorr 04] better, we use a small example dataset. We begin with some definitions.

4.3.1 Standard Deviation

Standard deviation (SD) is a widely used measure of variability, which shows how much variation exists from the average, it tells us how spread out the data is.

For a uni dimensional data set, SD is given by Eq. (4.1)

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{(n-1)}} \qquad (4.1)$$

4.3.2 Variance

Variance is a measure of how far the data set is spread out; it is almost identical to standard deviation.

$$V = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{(n-1)} \qquad (4.2)$$

Variance is the square of standard deviation

4.3.3 Covariance

Covariance is the measure of how much two random variables change together. With variance we can measure one-dimensional dataset, but if we have two or more dimensions, we use covariance. This tells us whether there is any relationship between the dimensions.

The covariance between x and y is given by Eq. (4.3)

$$Cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1} \qquad (4.3)$$

If covariance is positive, it signifies that both the dimensions increase together. If covariance is negative, then as one-dimension increases, other dimension decreases. If the covariance is zero, it indicates that the two dimensions are independent of each other.

If we have a dataset with more than two dimensions, for example (x, y, z), then we calculate Cov (x, y), Cov (y, z) and Cov (z, x). The best way to represent this is to put it into a matrix as shown in Eq. (4.4)

$$Cov(x,y,z)=\begin{pmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{pmatrix}_{3x3} \qquad (4.4)$$

In the above matrix, we can notice that along the diagonal, the covariance value is between one dimension and itself; this gives the variance of that dimension. The covariance matrix is symmetrical about the main diagonal, since

$$Cov(a, b) = Cov(b, a) \qquad (4.5)$$

4.3.4 Eigenvectors and Eigenvalues

The Eigenvectors of a square matrix are the non-zero vectors that after being multiplied by a matrix, remains parallel to its original vector.

For each eigenvector the corresponding eigenvalue is the factor by which the eigenvector is scaled when multiplied by the matrix, the mathematical expression of this idea is as follows.

If 'A' is a square matrix, a non-zero vector 'v' is an eigenvector of 'A' if there is a scalar λ, such that

$$Av = \lambda v \qquad (4.6)$$

The scalar λ is said to be the eigenvalue of 'A' corresponding to 'v'

The following are some of the properties of eigenvectors

1. Eigenvectors can only be found for square matrix and not every square matrix has eigenvectors.
2. If a given $n \times n$ matrix does have eigenvectors, then there are 'n' of them.
3. All eigenvectors are perpendicular.
4. Eigenvectors and Eigenvalues always come in pairs.

In order to keep the eigenvectors standard, we scale all the eigenvectors to a length of 1.

4.4 PCA Example Calculation

The above-discussed mathematical concepts are enough to understand PCA. The following example calculation and graphs will help us to understand PCA even better.

Consider the data shown in Table 4.1.

Table 4.1: Original Data

X	Y
10	20
35	60
40	80
11	10
5	90
6	4
50	15
22	45
36	70

The above data is plotted in a graph, which is shown in Figure 4.1. This graph does not convey any relationship between the elements in the data set.

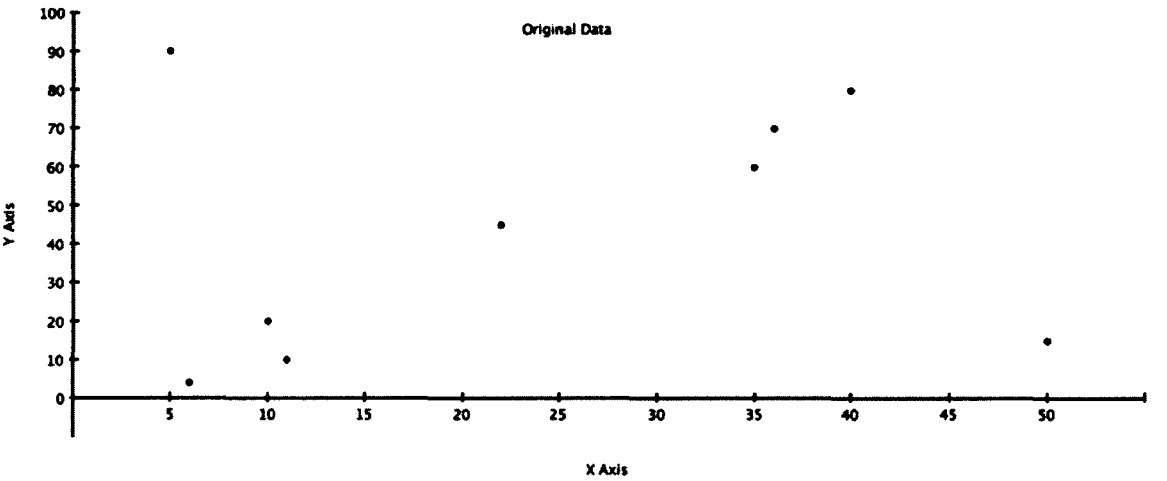

Figure 4.1: Plot of Original Data

The mean of the variable x and variable y are found and they are represented by $\bar{x}$ *and* $\bar{y}$ respectively. Then we subtract the mean value from the original value, and the result is shown in Table 4.2.

Table 4.2: Mean Adjusted Data

$X-\bar{X}$	$Y-\bar{Y}$
-13.88	-23.77
11.11	16.22
16.11	36.22
-12.88	-33.77
-18.88	46.22
-17.88	-39.77
26.11	-28.77
-1.88	1.22
12.11	26.22

The above mean adjusted data is plotted in a graph shown in Figure 4.2.

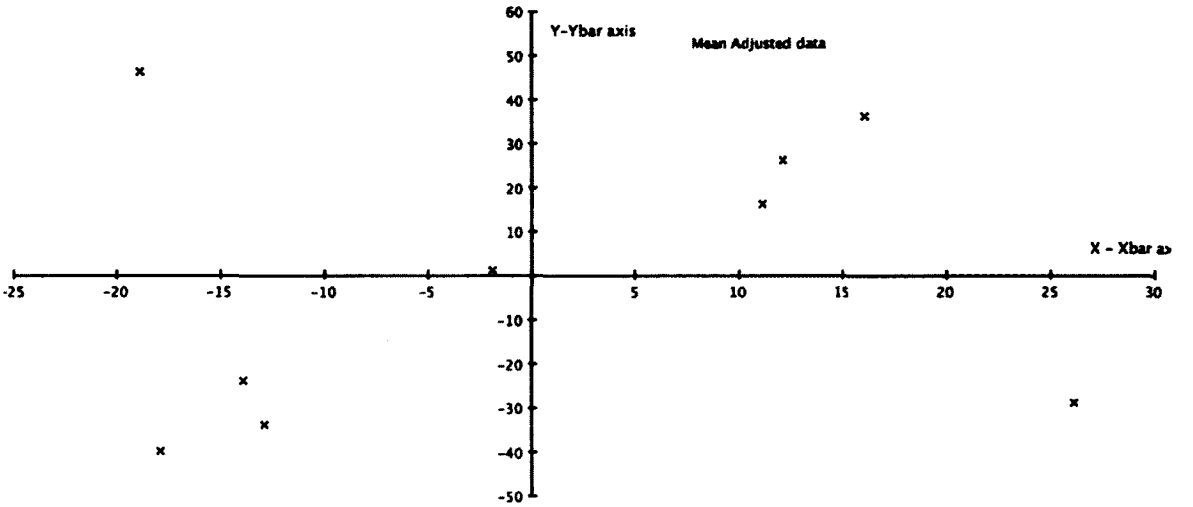

Figure 4.2: Mean Adjusted Data

The covariance matrix of x and y is found using the formula from Eq. (4.3)

$$Cov(x,y) = \begin{pmatrix} 250.09 & 103.53 \\ 103.53 & 946.39 \end{pmatrix} \tag{4.7}$$

The Eigenvectors and Eigenvalues are calculated from the given covariance matrix.

Eigenvalues are: 235.02, 961.45

Eigenvectors are: $v_1 = \begin{bmatrix} -0.99 \\ 0.14 \end{bmatrix}, v_2 = \begin{bmatrix} -0.14 \\ -0.99 \end{bmatrix}$ (4.8)

The mean adjusted data is plotted along with the eigenvectors, which are represented by dotted lines, as shown in Figure 4.3.

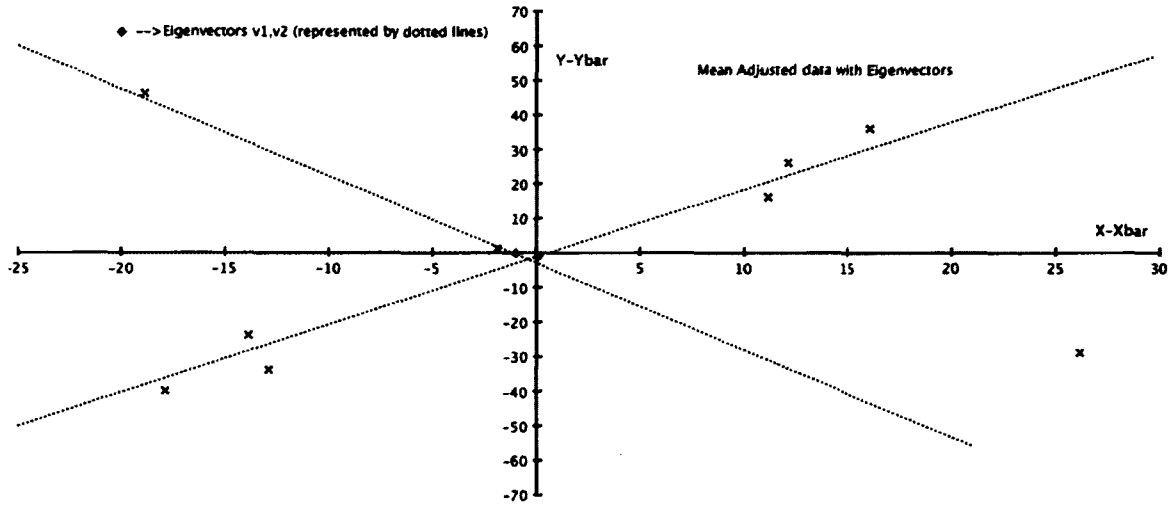

Figure 4.3: Mean Adjusted Data with Eigenvectors

A feature vector is formed from the obtained eigenvectors. The eigenvectors are concatenated together based on their eigenvalues, the eigenvector with the highest eigenvalue is added first, then the eigenvector with second highest value and so on.

$$Feature\ vector = (eigenvector_1, eigenvector_2 \cdots\cdots eigenvector_n) \quad (4.9)$$

$$Feature\ vector = \begin{pmatrix} -0.99 & -0.14 \\ 0.14 & -0.99 \end{pmatrix} \quad (4.10)$$

The final data is obtained using Eq. (4.11)

$$Final\ data =$$

$$(Transpose\ of\ feature\ vector) \times (Transpose\ of\ Mean\ adjusted\ data) \quad (4.11)$$

Table 4.3: Final Data

Final data X	Final data Y
25.53	10.31
-17.66	-8.67
-38.17	-10.73
35.27	7.88
-43.03	25.34
41.93	11.97
24.71	-29.99
-0.95	2.04
-27.7	-8.22

The final data shown in Table 4.3 is plotted in a graph as shown in Figure 4.4. The eigenvectors are represented as black dots in Figure 4.4.

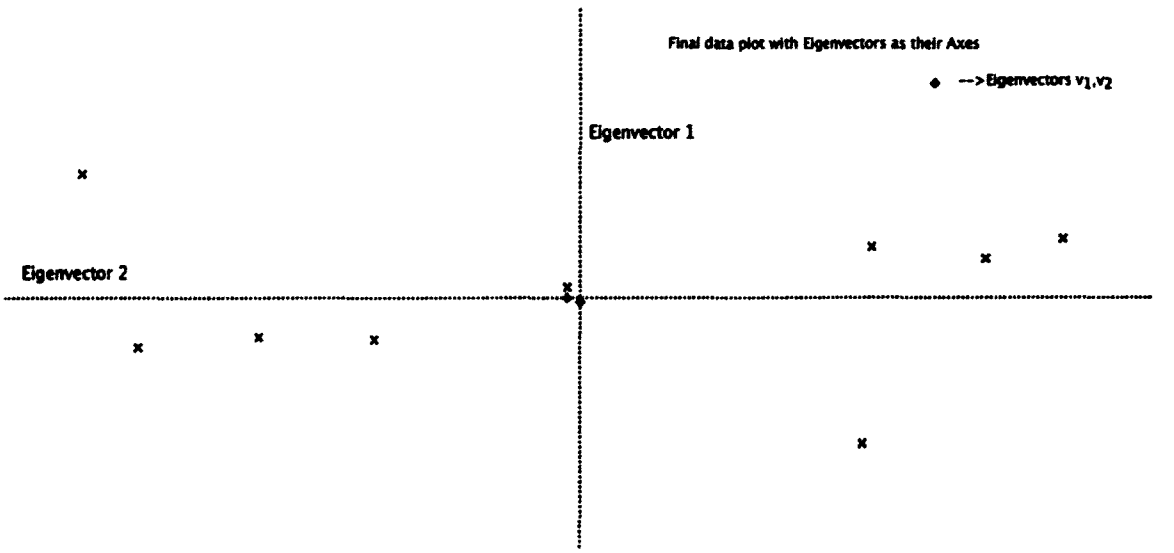

Figure 4.4: Final Data with Eigenvectors

The Eigenvectors form the axes for the final data as shown in Figure 4.4. Incase we have more eigenvectors; we would have more than two axes. The axes of eigenvectors are always perpendicular which makes it more efficient to express the data set.

Fundamentally we have transformed our data set so that it is expressed in terms of patterns between them. The patterns are the lines that can efficiently describe the relationship between the data.

Comparing the original dataset with the final data set, as shown in Figure 4.5, gives us an idea about the meaningful result produced by PCA for a random data set.

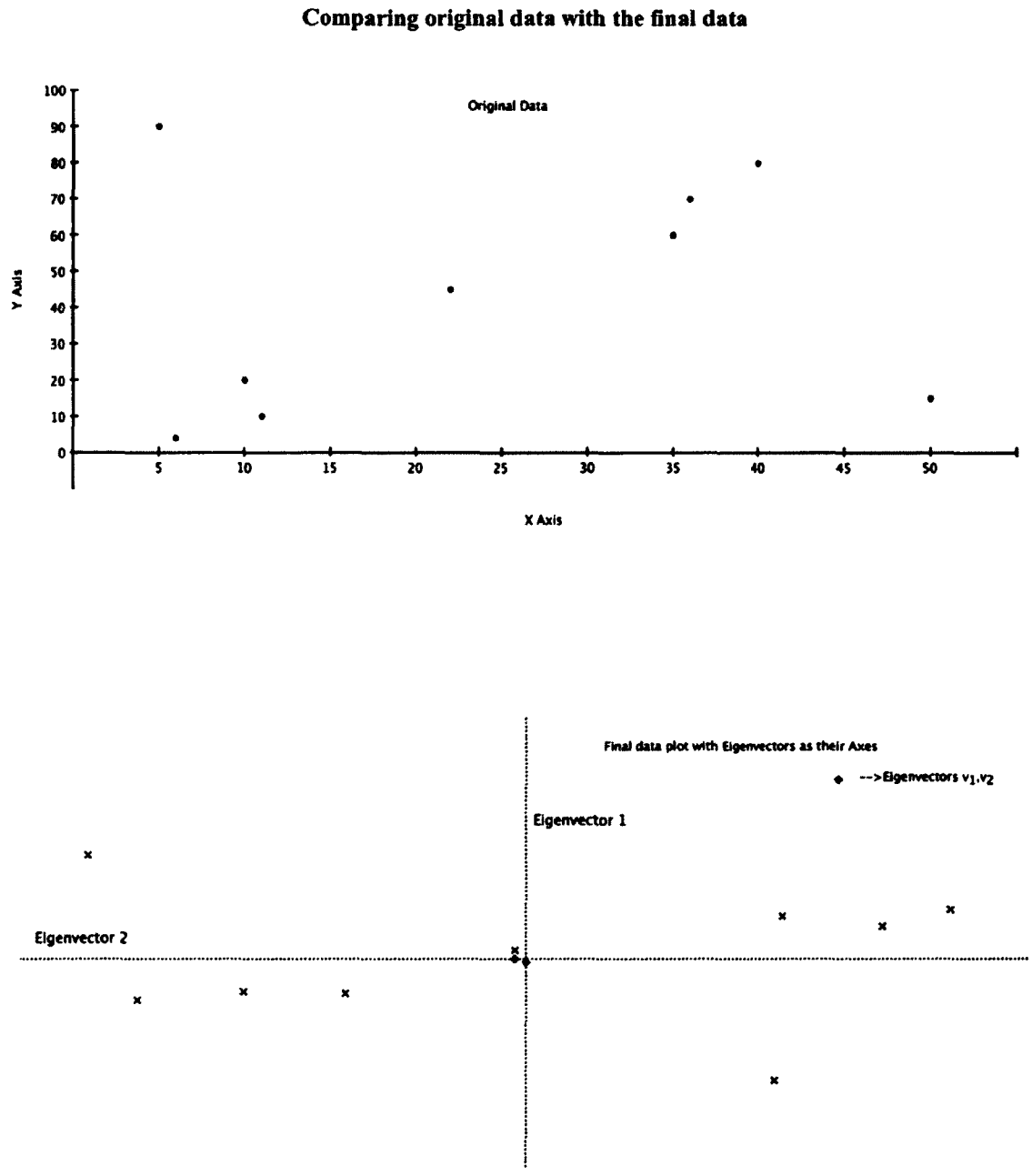

Figure 4.5: Comparison between Original Data and Final Data

We can see that representing the data set in terms of their eigenvectors can efficiently describe the relationship between the elements in the dataset. It clearly describes the pattern in the dataset, whereas the original dataset can only represent the scattered

elements that do not give us any indication about the relationship between the elements. This type of data analysis technique from PCA is used in eigenfaces method to classify the image vectors.

4.5 PCA in Face Recognition

PCA is a statistical dimensionality reduction tool, Kirby and Sirovich (1990) [Kirb 87] applied PCA for representing faces, Turk and Pentland (1991) [Pent 91] extended PCA to recognize faces. To understand the role of PCA in face recognition, we should first consider the representation of images.

Images are represented as a matrix of pixels. Consider an image of dimension $N \times N$; this can be represented as N^2 dimensional vectors by concatenating all the rows into a single column. Similarly for 5 different images, each of dimension $N \times N$, we will have 5 different image vectors. Then we concatenate all these vectors together to get a matrix. We then apply PCA on this matrix, which gives us the original data in terms of eigenvectors. Once we get the test image, we project the test image on the image space. Then we find the difference between the test image and the images in the database using a distance classifier. This effectively discriminates the images in the database that resemble the test image. Figure 4.6 illustrates the role of PCA and distance classifier in face recognition.

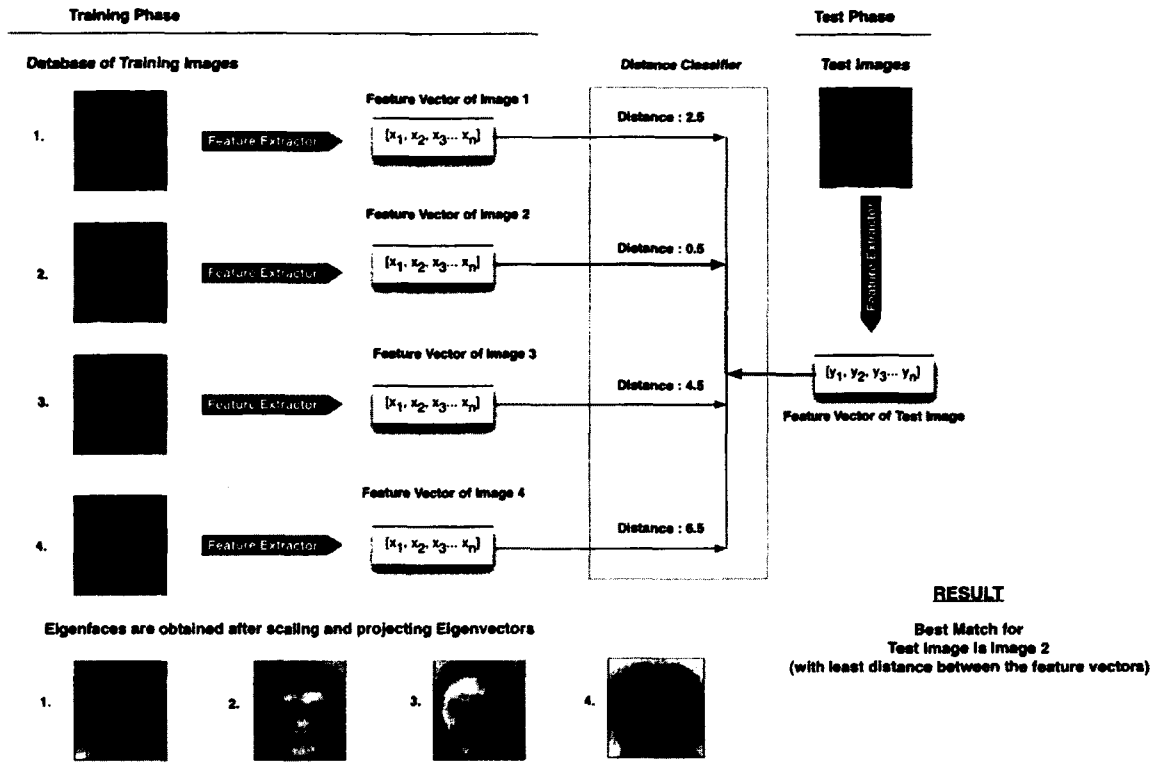

Figure 4.6: PCA in Face Recognition

PCA is a statistical analysis tool. To personalize PCA for face recognition, we need a new algorithm. The EA developed by Turk and Pentland [Turk 91] [Pent 91] is one such algorithm.

4.6 Eigenfaces

An Input image consists of many characteristic features. PCA is a mathematical tool that we use to highlight and differentiate these features. Once we have these features for a set of images, we find their feature vectors or eigenvectors. These eigenvectors are also called as eigenfaces when they are projected into the image space.

Once we have a set of eigenfaces for the database of images, we find the weights that are required for the eigenfaces to reconstruct each image in the database. When we are presented with a test image, we find the weight vector of the test face by projecting it into the eigenfaces. We then compare the weight of the test face with the weight of the images in the database using a distance classifier. This tells us how closely each particular image in the database resembles the test image. This procedure is an extension of PCA called as eigenfaces.

Sirovich and Kirby's efficient representation of faces using PCA was the motivation for the concept of eigenfaces. Turk and Pentland extended PCA and arrived at a method "that would build up the characteristic features by experience over time and recognize a particular face by comparing the feature weights needed to (approximately) reconstruct them with the weights associated with the known individuals" [Turk 91] [Pent 91].

With EA the individual images could be represented compactly as eigenfaces based on their features. From these eigenfaces we can also reconstruct an image from the database, since all we need are only the eigenfaces and since it is very compact, eigenfaces would use very less memory.

Since the publication of eigenfaces many new algorithms have been proposed. However, even today eigenfaces remains the benchmark for face recognition algorithms.

4.7 Eigenface Approach

The steps used in EA [Turk 91] [Pent 91] [Triv 09] [Carm 09] for face recognition are as follows:

1. *Initialization* – The Images that constitute the database are assimilated
2. *Calculation* – The eigenfaces are calculated from the images in the database. The M eigenvectors that correspond to the highest eigenvalues are kept. They constitute the face space, and this is constantly updated as we obtain more images for the database.
3. *Finding Weights* – The weight vectors of the known images are found by projecting them into the face space. These weight vectors can be used to reconstruct a face in the database using the eigenfaces.

The process mentioned above is done offline (back end process); we are required to calculate the weights only when the database needs to be updated. The following are the steps for recognition process; they used to be done online (front end process) whenever the test image is produced.

1. When the test image is produced for identification, the weight vectors associated with the test image are found by projecting them on the face space.
2. Once we have the weight vector of the test image, we compare it with the weight vectors of the known images in the database, so that we can ascertain whether the test face is a known face or an unknown face.
3. If the weight vector lies with in the face space, we can conclude that the given image is a face, and then we find if there is any closest neighbor to the test vector.

Once we find the closest neighbor and if it satisfies the threshold condition, we can say that the face is a know face from the database and its identity can be established.

4. The eigenfaces and weight pattern are updated once we get new images for the database. If there is an unknown face that is seen constantly, it could be labeled as a known face and added to the database.

The eigenfaces method [Piss 02] is illustrated using a flowchart as shown in figure 4.7.

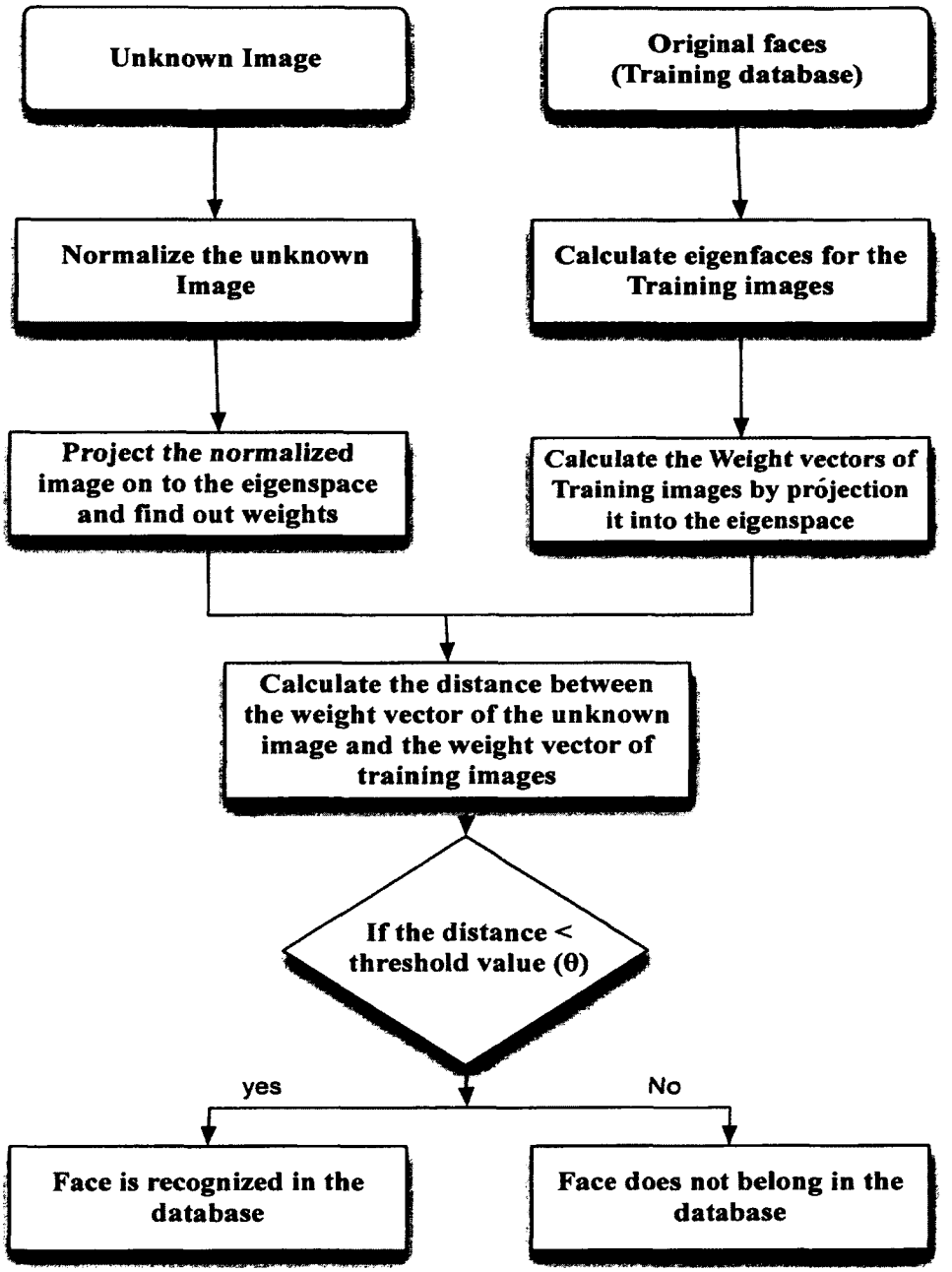

Figure 4.7: Flowchart – Eigenfaces

4.8 Assumptions in EA

The following are the assumptions that are made in eigenfaces procedure

1. There are M images in the training database
2. There are 'k' most significant eigenfaces, using which we can satisfactorily approximate a face, where (k < M)
3. All images are $N \times N$ matrices, which can be represented as $N^2 \times 1$ dimensional vectors. Same logic applies for images with unequal length and breadth.

4.9 EA

The figure 4.8 illustrates the concept of eigenfaces.

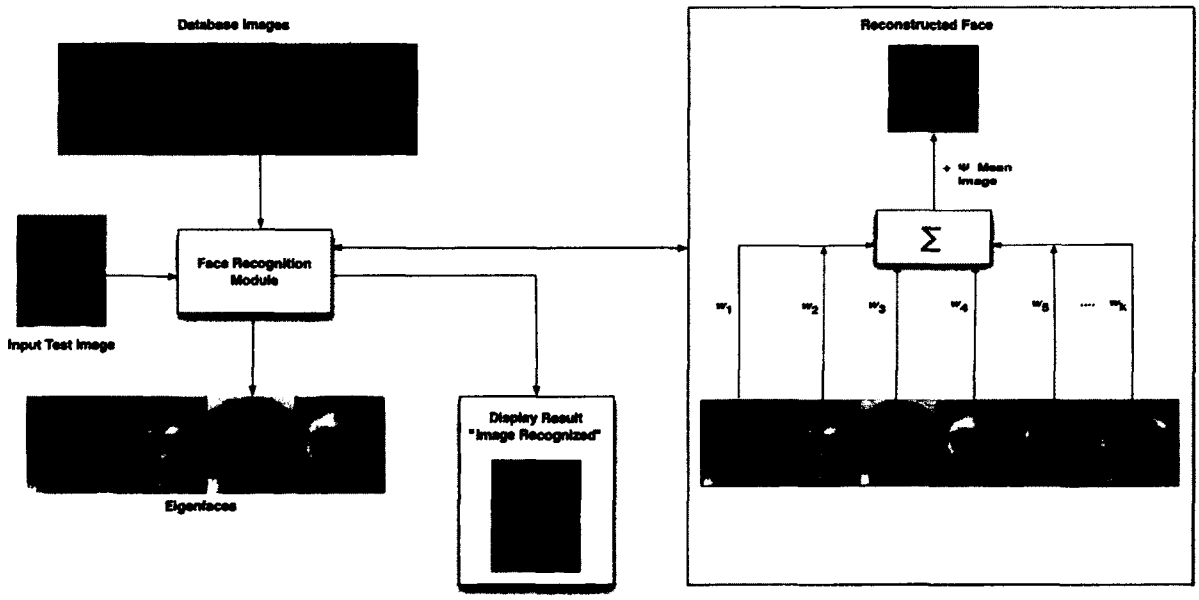

Figure 4.8: Eigenfaces Concept

The first step of EA [Zaba 09] is to obtain the training set; this consists of M grey-scale face images $I_1, I_2 \cdots I_M$ they must be face centered images of same scale. Figure 4.9 illustrates few of the faces obtained from AT&T database

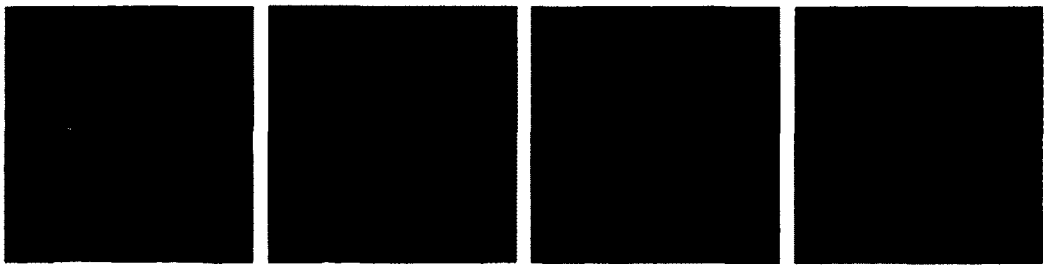

Figure 4.9: Face Images from AT&T Database

An image I_i can be represented as

$$I_i = \begin{bmatrix} a_{11} & a_{12} & \square & a_{1N} \\ a_{21} & a_{22} & \square & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \square & a_{NN} \end{bmatrix}_{N\times N} \xrightarrow{Concatenation} \begin{bmatrix} a_{11} \\ \vdots \\ a_{1N} \\ a_{21} \\ \vdots \\ a_{2N} \\ \vdots \\ a_{N1} \\ \vdots \\ a_{NN} \end{bmatrix}_{N^2\times 1} = \Gamma_i \qquad (4.12)$$

The values in the matrix above are the pixel values ranging from 0-255, once we have the pixel values; we change the $N \times N$ matrix to $N^2 \times 1$ but concatenating the rows into a single column, this makes the image as a vector in N^2 dimension. The characteristic features of the images are of prime focus, so we have to subtract all the common

elements between them. So we begin by finding the average face and then we subtract each face from the average face so we can find the difference face.

The average face is give by

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \qquad (4.13)$$

Figure 4.10 represents an average face, which was obtained from the image set in AT&T database.

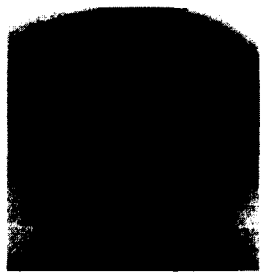

Figure 4.10: Average Face

Now, each face differs from the average face by the vector

$$\Phi_i = \Gamma_i - \Psi \qquad (4.14)$$

Figure 4.11 illustrates few the difference faces obtained using EA

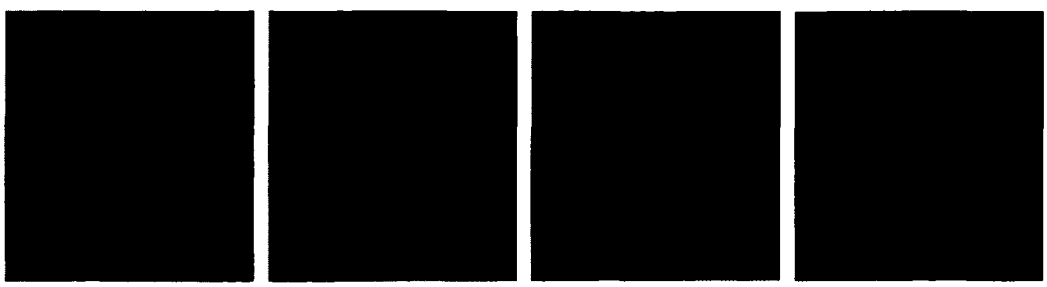

Figure 4.11: Difference Faces

A set of orthonormal vectors are to be found, these vectors best describe the distribution of the data, we begin with finding the covariance matrix

$$C=\frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = A\,A^T$$

where $A=[\Phi_1,\Phi_2,\ldots\ \Phi_M]$ (4.15), (4.16)

The next step is to find the eigenvectors u_k and eigenvalues λ_k of C

However, we can see that A is of the size $N^2 \times M$, hence the matrix C is of the size $N^2 \times N^2$, to assess things, assume the image of size 120×120, then the size of the resulting matrix would be 14400×14400. Determining eigenvectors and eigenvalues for a matrix of this size would be extremely difficult, but this hurdle could be easily crossed by a simple mathematical trick.

Lets take $L = A^T A$, then the size of L matrix would be $M \times M$, we then solve the L matrix to find the eigenvectors v_i *where* $i = 1 \ldots M$ *of* L

Now $Lv_i = \lambda_i v_i$ *then* multiplying A on both sides we get

$$\Rightarrow A\,L\,v_i = \lambda_i A v_i$$
$$\Rightarrow A\,A^T A\,v_i = \lambda_i A v_i$$
$$\Rightarrow C\,A\,v_i = \lambda_i A v_i \qquad (4.17), (4.18), (4.19)$$

Hence $u_i = Av_i$ and λ_i are respectively the M eigenvectors and eigenvalues of C

When we get the eigenvectors they are generally normalized to 1, such that $\|u_i\| = 1$

These eigenvectors u_i are called as eigenfaces, when we scale these vectors by 255 and project them on the face space, we will get a ghostly face called eigenfaces, as shown in the figure 4.12.

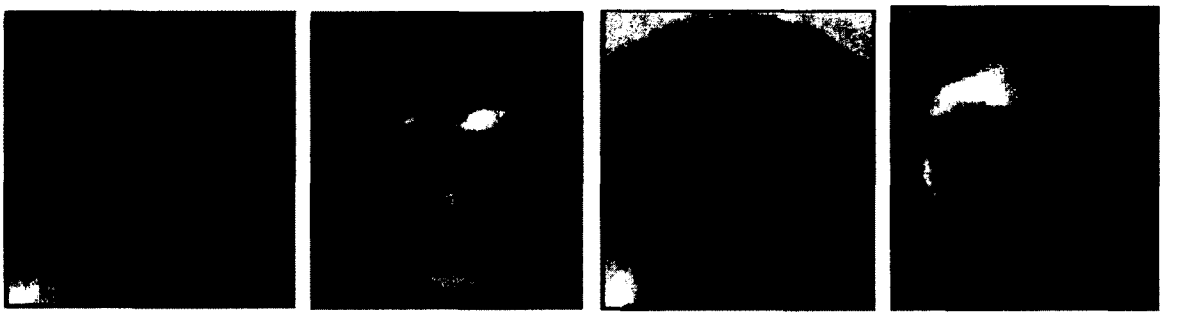

Figure 4.12: Eigenfaces

The eigenvectors with smallest eigenfaces can be excluded, so we are only left with k eigenvectors, where ($k \leq M$). The reason for excluding the eigenvectors with smallest eigenfaces is that, the eigenvectors with highest eigenvalues are the ones that contribute most to the eigenfaces.

These eigenvectors are grouped together as $U = [u_1, u_2 \cdots u_k]_{N^2 \times k}$ (4.20)

Once the basis vector (U) for the face space has been constructed, all that remains is to project all the images in the training set onto the 'face space'. This can be done by the following operation

$$\Omega = U^T(\Gamma - \Psi) = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_k \end{bmatrix}_{k \times 1} \quad (4.21)$$

The weights $\omega_i \in \Omega$ describes the contribution of each eigenface in representing the input face image, once we have the weights of the known images $(\Omega_1, \Omega_2 \cdots, \Omega_M)$and the test images (Ω), we can find the smallest Euclidean distance ϵ_{rec} between the test face and training face weight vectors from the following expression:

$$\epsilon_{rec} = \min\|\Omega - \Omega_i\| \quad \text{Where } i = 1 \ldots \text{M} \qquad (4.22)$$

If $\epsilon_{rec} < \theta_{rec}$, then we can say that the test face is identified as the image which gives the lowest score, where θ_{rec} is chosen heuristically.

If $\epsilon_{rec} > \theta_{rec}$, then we can say that the face is not identified in the database.

4.10 Face Space

The space in which the image vectors could be mapped is known as Image space, the space in which the face vectors could be mapped is known as the face space, the image space contains the face space. Figure 4.13 illustrates the idea of image space and face space [Triv 09].

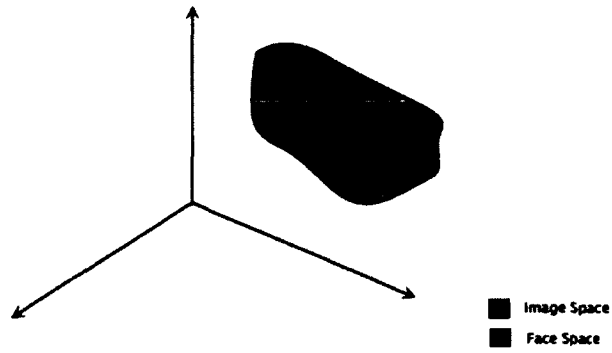

Figure 4.13: Image Space and Face Space

When we project the weight vector of the test image onto the face space, there are four possibilities on where an image could lie

1. *Near a face class and near a face space* – This case happens when the test image is of a known individual from the database.
2. *Near a face space but away from face class* – This case happens when the test image consists of a face that is not present in the database.
3. *Distant from face space near face class* – This case happens when the test image is not a face, however it sill resembles a particular face class stored in the database
4. *Distant from both face space and face space* – when the probe is not a face image, i.e., away from the face space and is nothing like any face class that is stored in the database.

Out of these four cases, case 3 is responsible for most false positives, but still the false recognition might be detected since there is significant distance between the weights of

the test image and the face image from the database. Figure 4.14 illustrates all the four cases.

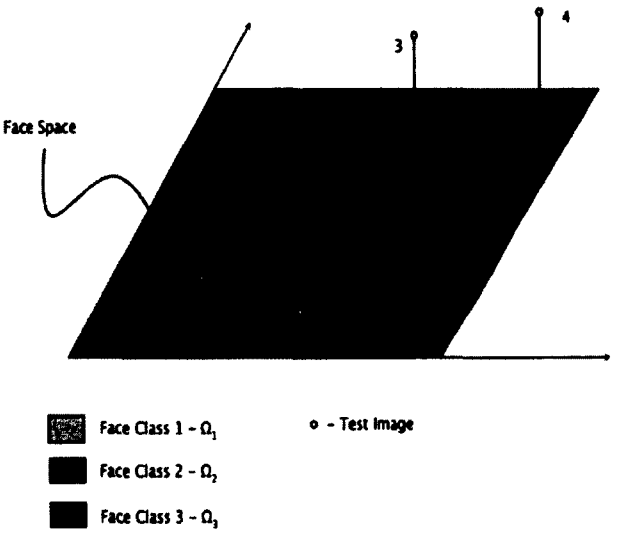

Figure 4.14: Different Possibilities of Face Space & Face Class

The concept of PCA and its role in face recognition were clearly illustrated. We then examined the EA in detail. Architecture for FPGA implementation of EA is proposed in the following chapter.

Chapter 5

Proposed Architecture for FPGA Implementation of EA

5.1 Introduction

In this chapter we describe an architecture for EA. This chapter begins with a discussion of the data representation used in the architecture and the assumptions in the EA. It then proceeds to describe the architecture for implementing the EA and divides it into two phases. Finally it elucidates the input, output and functionality of each and every module of Phase I and Phase II with the help of a model calculation based on eigenfaces.

FPGA is a digital Integrated Circuit (IC) that contains configurable blocks of logic along with configurable interconnects between these blocks. Design Engineers can program such a device to perform a variety of tasks.

The proposed architecture is designed flexibly around the face images, which are used to build our database. These images are taken under controlled conditions, so that they are of same scale and have similar lighting conditions.

5.2 Data Representation

A grey-scale image for representing a face is sufficient, since eigenfaces method does not depend on the color of the image. This is advantageous, since it would require less memory. The pixel values of the grey scale images are in the range 0-255, where 0 is black and 255 is white, and the rest are different shades of grey [Work 10]. Each pixel of the grey-scale image is represented by an 8 bit signed binary value using 2's complement representation.

In EA arithmetic calculations are widely used, so the number of bits to represent the data will keep on changing from one module to other, therefore to compensate, we increase or decrease the number of bits as per need, so the data width keeps changing from one module to another [Cory 03] [Cory 05].

5.3 Assumptions

Solving EA mathematically helps in understanding the building blocks of the architecture. The dimensions of the image are 92×112. Solving a matrix of size 92×112 would be improbable, so a smaller example matrix of size 3×3 is assumed to be an image. The mathematical calculations of EA are applied on these example matrices. Analyzing the model mathematical calculations helps in designing the modules and their functionalities using Verilog HDL.

While applying EA on the example matrix, we would obtain values with fractional component, but in the proposed deign we have excluded any fractional component, since excluding the fractional part [Fish 04] would not have any considerable

effect other than increasing or decreasing the brightness of the grey scale image, and also including the fractional part would complicate the architecture for FPGA.

In our proposed architecture, we have 3 images that make up the database; these images are of the size (92×112), so we would have 10304 pixels (since $92 \times 112 = 10304$). We are also introducing an unknown image during the EA; this is the test image that is to be recognized. The test image is of the same size as the other images from the database, so we will be working with 41216 pixels ($10304 \times 4 = 41216$). As per the EA the images are represented as follows

1. Known Image 1 - Γ_1
2. Known Image 2 – Γ_2
3. Known Image 3 - Γ_3
4. Unknown Image - Γ

As we proceed we will simultaneously discuss both the 'model calculations' and their corresponding 'architecture' step by step.

5.4 Proposed Architecture

The proposed architecture in Figure 5.1 can divided in to two phases

1. Phase I
2. Phase II

Phase I

Begins by uploading the known images into the RAM. It continues with the mathematical operations on the image vectors and ends when the eigenvectors and eigenvalues are obtained.

Phase II

Begins from the eigenvectors and eigenvalues module, proceeding to the weight vector module, where the test face is introduced and it ends when the identity of the test face has been established as a known or an unknown face.

Figure 5.1 illustrates the proposed architecture for FPGA Implementation of EA

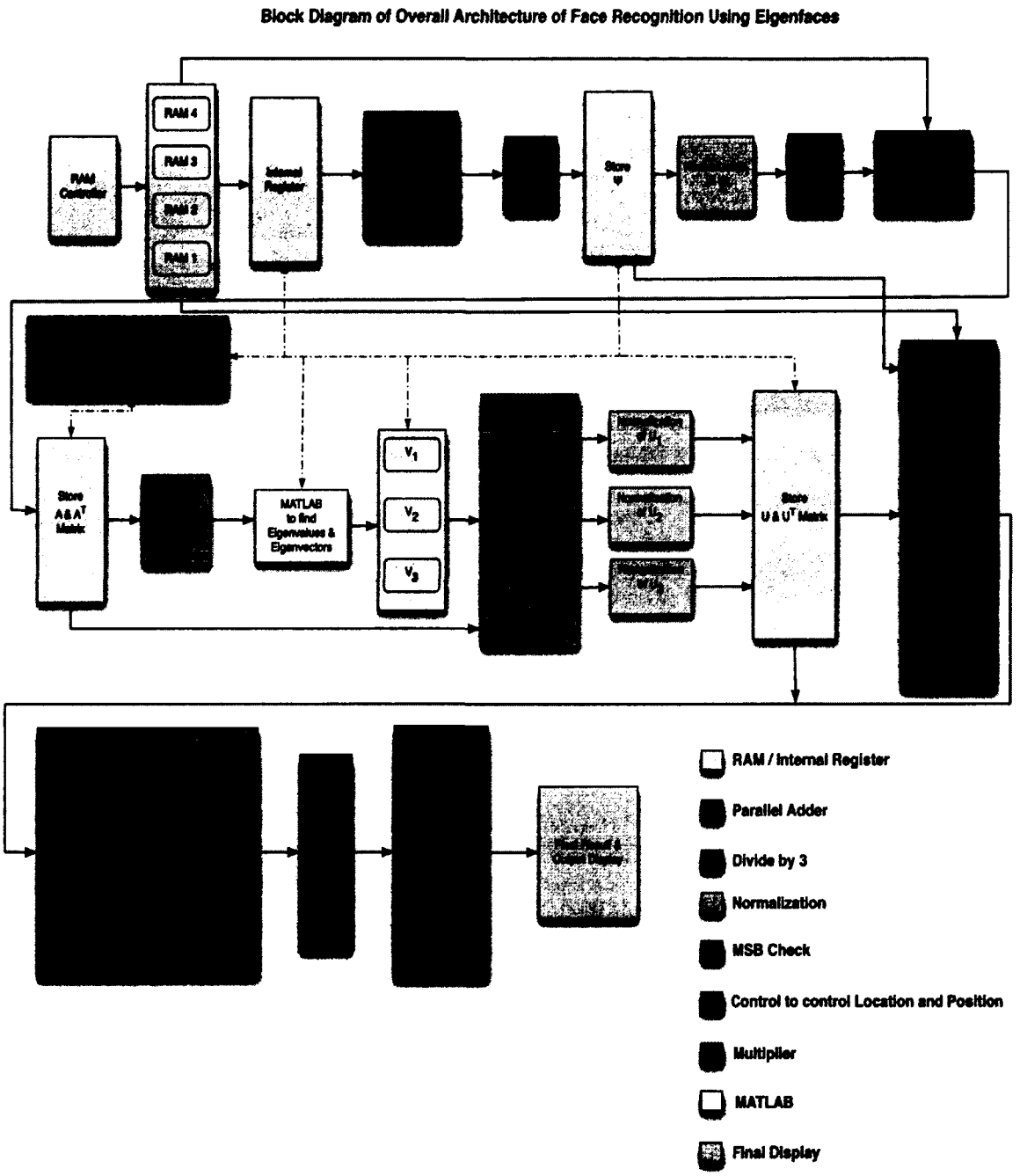

Figure 5.1: Proposed Architecture for FPGA Implementation of EA

5.4.1 Phase I

Figure 5.2 illustrates Phase I of the architecture.

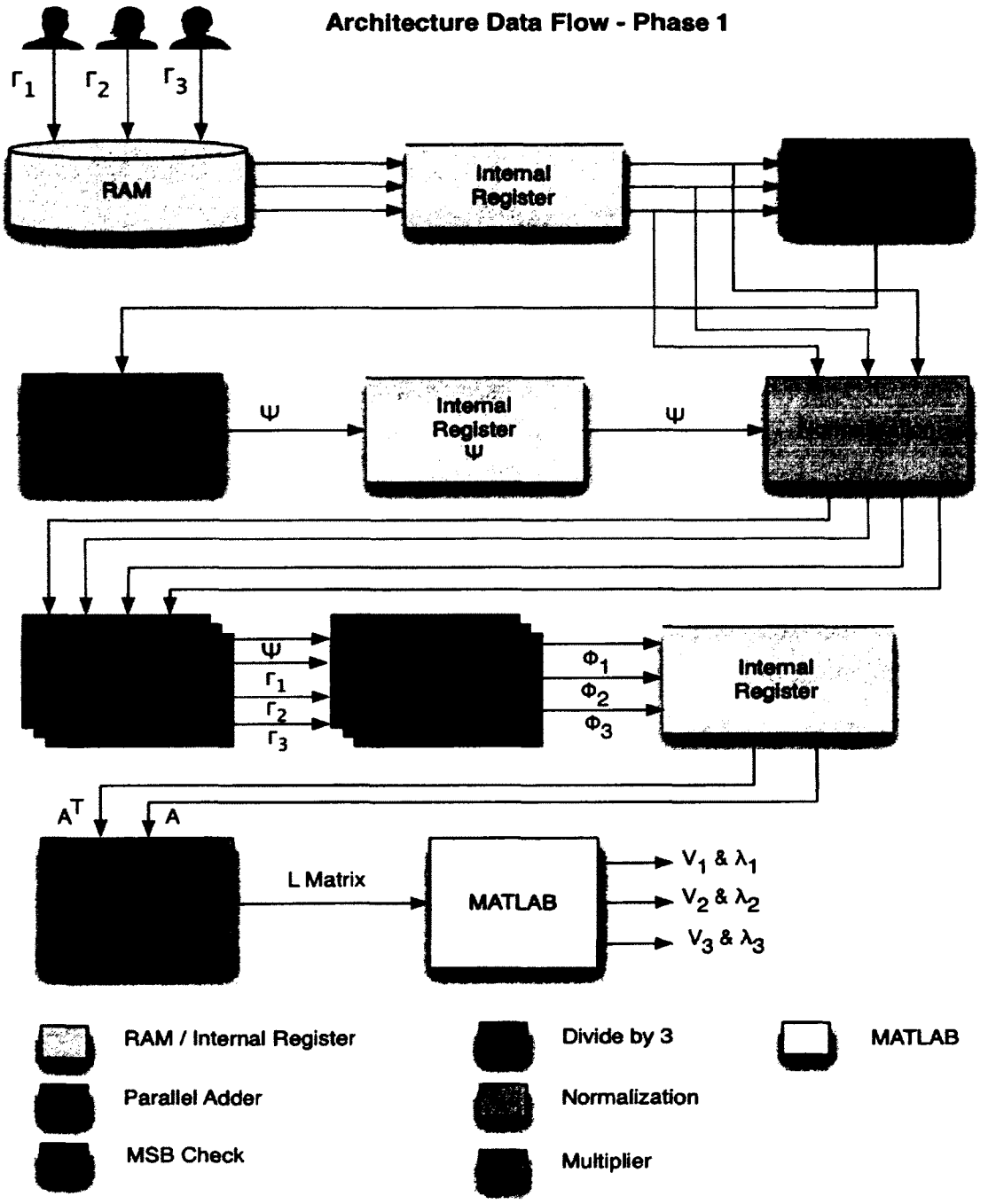

Figure 5.2: Architecture Dataflow - Phase I

Model calculation

Step 1: Let the assumed smaller matrix of known faces be Γ_1, Γ_2 *and* Γ_3

$$\Gamma_1 = \begin{bmatrix} 10 & 35 & 40 \\ 11 & 5 & 6 \\ 50 & 22 & 36 \end{bmatrix}_{3\times3} \quad \Gamma_2 = \begin{bmatrix} 20 & 60 & 80 \\ 10 & 90 & 4 \\ 15 & 45 & 70 \end{bmatrix}_{3\times3} \quad \Gamma_3 = \begin{bmatrix} 40 & 25 & 92 \\ 3 & 80 & 42 \\ 22 & 5 & 10 \end{bmatrix}_{3\times3}$$

(5.1), (5.2), (5.3)

Step 2: To apply eigenfaces method, we should first convert the above matrices into vectors

$$\Gamma_1 = \begin{bmatrix} 10 \\ 35 \\ 40 \\ 11 \\ 5 \\ 6 \\ 50 \\ 22 \\ 36 \end{bmatrix}_{9\times1} \quad \Gamma_2 = \begin{bmatrix} 20 \\ 60 \\ 80 \\ 10 \\ 90 \\ 4 \\ 15 \\ 45 \\ 70 \end{bmatrix}_{9\times1} \quad \Gamma_3 = \begin{bmatrix} 40 \\ 25 \\ 92 \\ 3 \\ 80 \\ 42 \\ 22 \\ 5 \\ 10 \end{bmatrix}_{9\times1} \qquad (5.4), (5.5), (5.6)$$

Architecture design for Step 1 and Step 2

We start off with four grey scale images of equal dimension (92 × 112), three of which are the known images that constitute the database, the fourth is used as the unknown test image, which would be introduced once the eigen faces are found.

A RAM block is required to store these images; we have 4 images (3known and 1 test image), so we are dividing the RAM into 4 blocks namely RAM1, RAM2, RAM3 and RAM4, where each block corresponds to Image1, 2, 3 and test image respectively.

The dimensions of the image is (92×112), each pixel is represented as 8 bit signed binary value. The total number of bits in one image:

$= (92 \times 112 \times 8) = 82432\ bits => 84KB$

The data width would increase or decrease as we move on from one module to another [Cory 03] [Cory 05]. The data widths for the following modules are indicated in their corresponding block diagram.

5.4.1.1 RAM Controller

A RAM Controller is required to read, write and control all the RAM Blocks. This controller would send in 736 bits [735:0] per cycle for 112 cycles ($736 \times 112 = 82432\ bits$) to successfully upload the pixels of one image into the RAM block. Table 5.1 illustrates the HDL port names of the RAM controller, its direction and description. Figure 5.3 illustrates the RAM controller block diagram.

Table 5.1: RAM Controller - Port Names and Description

HDL Port Names	Direction	Description
din_1 [735:0]	Output	Output data
din_valid_1	Output	Data valid signal
Wa [6:0]	Output	Write address
Ce_n_1	Output	Chip select for RAM1
ra [6:0]	Output	Read address
re1	Output	Read enable

row_n[6:0]	Output	Row address selection
we1	Output	Write enable
clk	Input	Clock signal
reset	Input	Asynchronous master reset

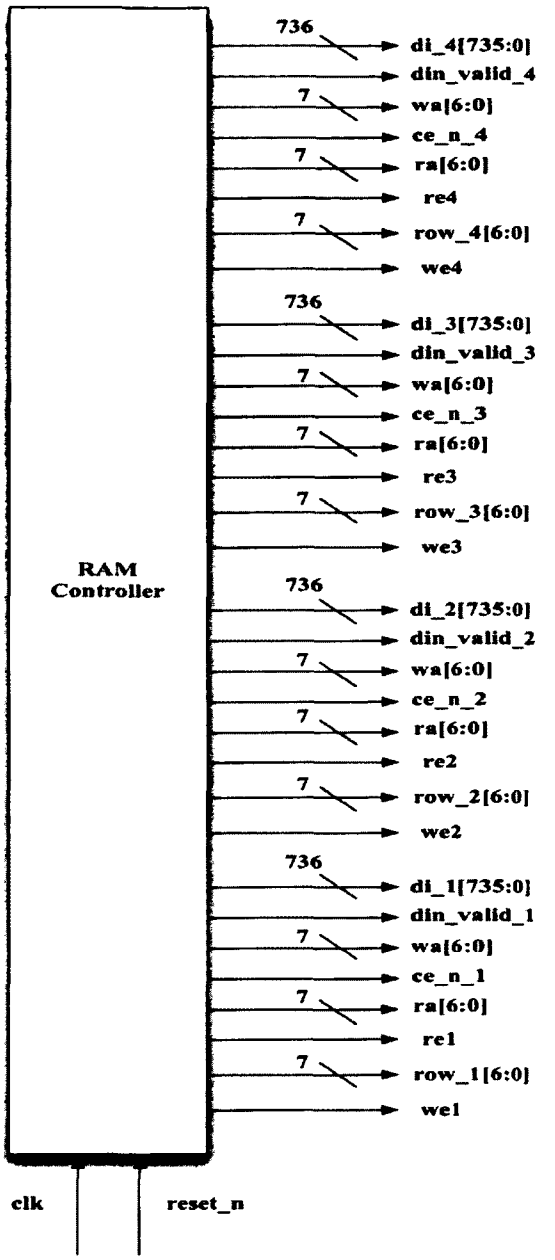

Figure 5.3: RAM Controller

5.4.1.2 RAM Blocks

Each RAM stores 82Kb of information, which is the size of the Images in the database; these images are transferred to the RAM blocks from the RAM controller. There they are stored as an array, which can be easily represented by a matrix. Figure 5.4 illustrates the block diagram of the individual RAM blocks.

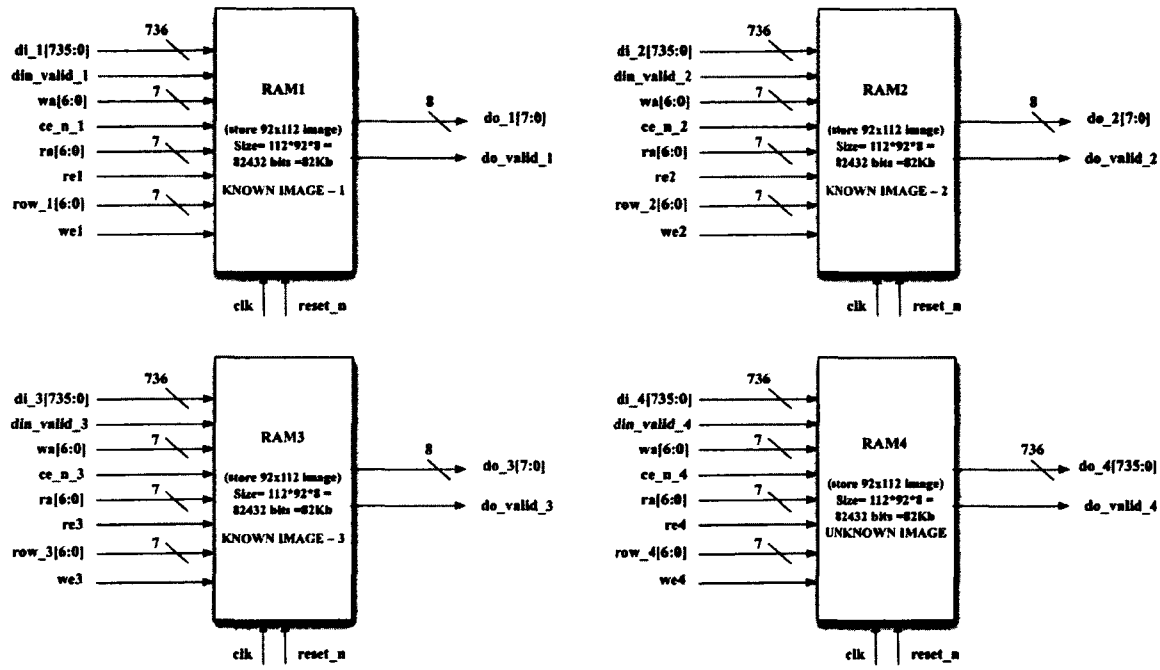

Figure 5.4: RAM Blocks

Most of the calculations in eigenfaces method are based on vectors, so it is important to convert the image matrix into a vector, this is done by concatenating the rows of the matrix into a single column, this vector is stored in the internal register, we will have one vector per image, so we will have 3 vectors in the internal register.

Each vector is of the size (10304 × 1), the image from RAM1 is stored pixel-by-pixel in the internal register as a single column, since we have 3 images in the database,

we store all the three matrix as 3 different vectors side by side, each pixel is 8 bit wide, so in a single row, we will have 3 pixels or 24 bits, therefore the size of the internal register would be [10303:0] mem [0:23]. Table 5.2 illustrates the HDL port names of the internal register, its direction and description. Figure 5.5 illustrates the block diagram of the internal register.

Table 5.2: Internal Register - Port Name and Description

HDL Port Names	Direction	Description
img_n [7:0]	Output	Data from internal register
add_n	Input	Read address
loc_add [13:0]	Input	Read address row location
mem_n [2:0]	Input	Write address
loc [13:0]	Input	Write address row location
do_n	Input	Input data from RAM
do_valid_n	Input	Valid signal from RAM

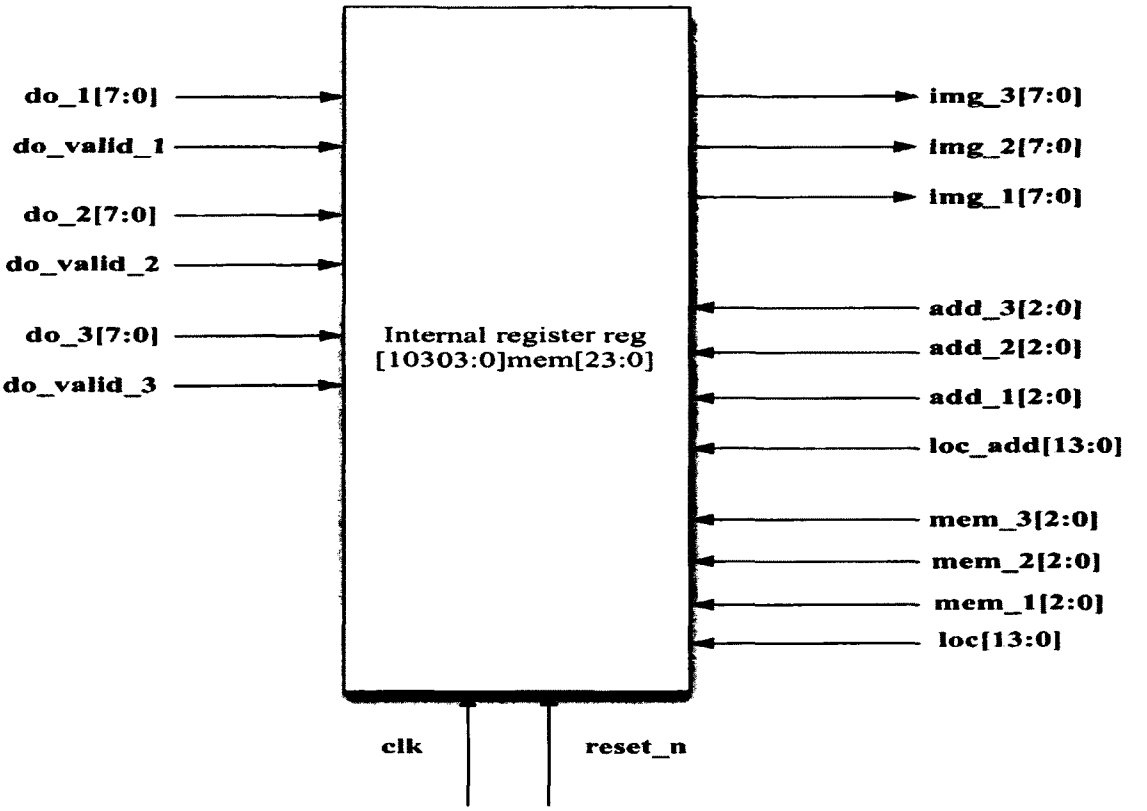

Figure 5.5: Internal Register

5.4.1.3 Controller to Control Address Location and Position

Similar to a RAM controller which controls the address for reading and writing in the RAM block, we need a controller to control the address location and position for internal register, the signal 'mem_n' controls where the data should be written in the memory location, 'add_n' controls the read address memory and the signals 'loc' and 'loc_add' are for write address row location and read address row location respectively. Figure 5.6 illustrates the block diagram of the controller to control the location and position selection.

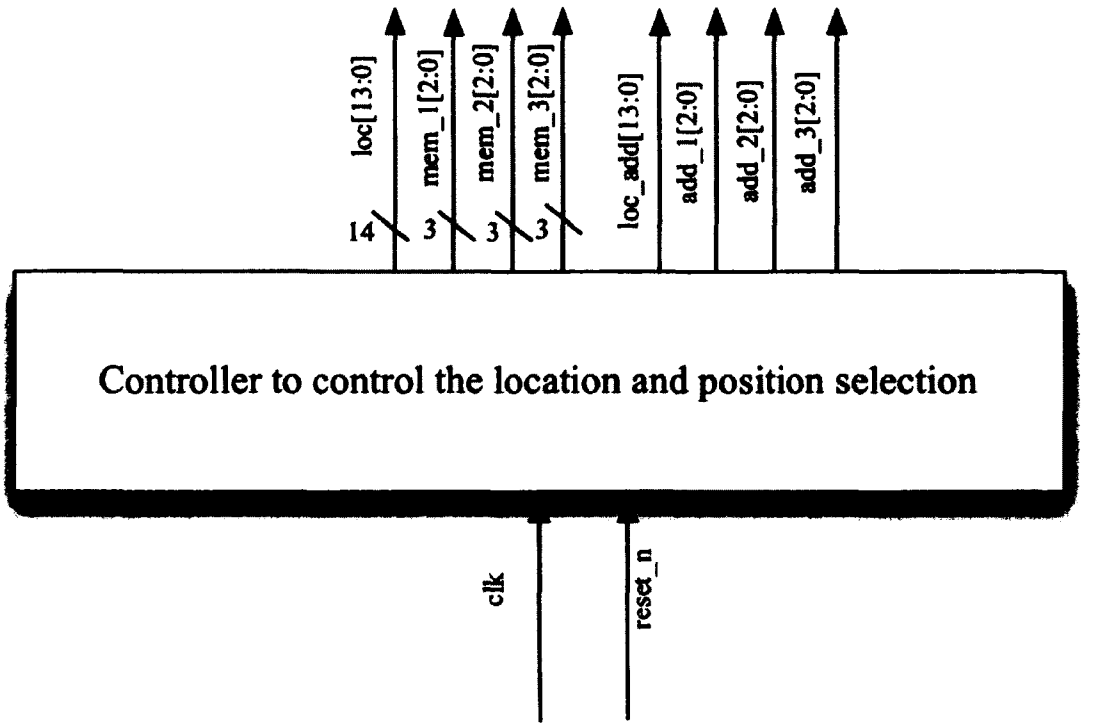

Figure5.6: Controller to Control the Address Location and position

Model Calculation Continued

Step 3: The Average face vector is given by Eq. (4.13)

$$\Psi = \frac{1}{3}\begin{bmatrix} 70 \\ 120 \\ 212 \\ 24 \\ 175 \\ 52 \\ 87 \\ 72 \\ 116 \end{bmatrix}_{9\times 1} \Rightarrow \begin{bmatrix} 23.33 \\ 40 \\ 70.66 \\ 8 \\ 58.33 \\ 17.33 \\ 29 \\ 24 \\ 38.66 \end{bmatrix}_{9\times 1} \quad \therefore \Psi = \begin{bmatrix} 23 \\ 40 \\ 71 \\ 8 \\ 58 \\ 17 \\ 29 \\ 24 \\ 39 \end{bmatrix}_{9\times 1} \qquad (5.7), (5.8)$$

Architecture Design for Step 3

To find the average face, we have to add the face vectors and divide them by the total number of faces.

Once the image vectors are stored in the internal register, we are using a parallel-pipelined adder to add all 3 image vectors pixel by pixel. Conventional methods are not sufficient for computationally intensive circuits, so using a parallel-pipelined adder instead of a normal adder would greatly reduce delay of the adder.

5.4.1.4 Pipelining

In traditional approach processes such as add, subtract and multiply etc., are treated as a single process, which may take considerable amount of time for processing.

In pipelining we have data flowing though combinational logic and registers driven by system clock.

Pipelining approach basically divides an entire process into small and equal sub-processes, such that the total processing time is substantially reduced due to concurrent execution of sub-processes. This provides much faster speed and throughput.

Lets consider a process of adding two 12bit numbers, this will be a time consuming process if the addition is carried out on 12bits, since the bit-wise carry needs to propagate though all the bits. A better way of doing this is to divide it into four, this will be very efficient than adding 12 bits at one go. This can be effectively carried out by pipelining, the LSBs of the two numbers are added first and stored in a pipeline register

along with the generated carry at the rising edge of the clock, in the next rising edge of the clock, the next 3 bits of the two numbers are added along with the carry generated while adding the LSBs. In this manner we process entire data width. Figure 5.7 illustrates the addition of two 12bit numbers using pipeline approach.

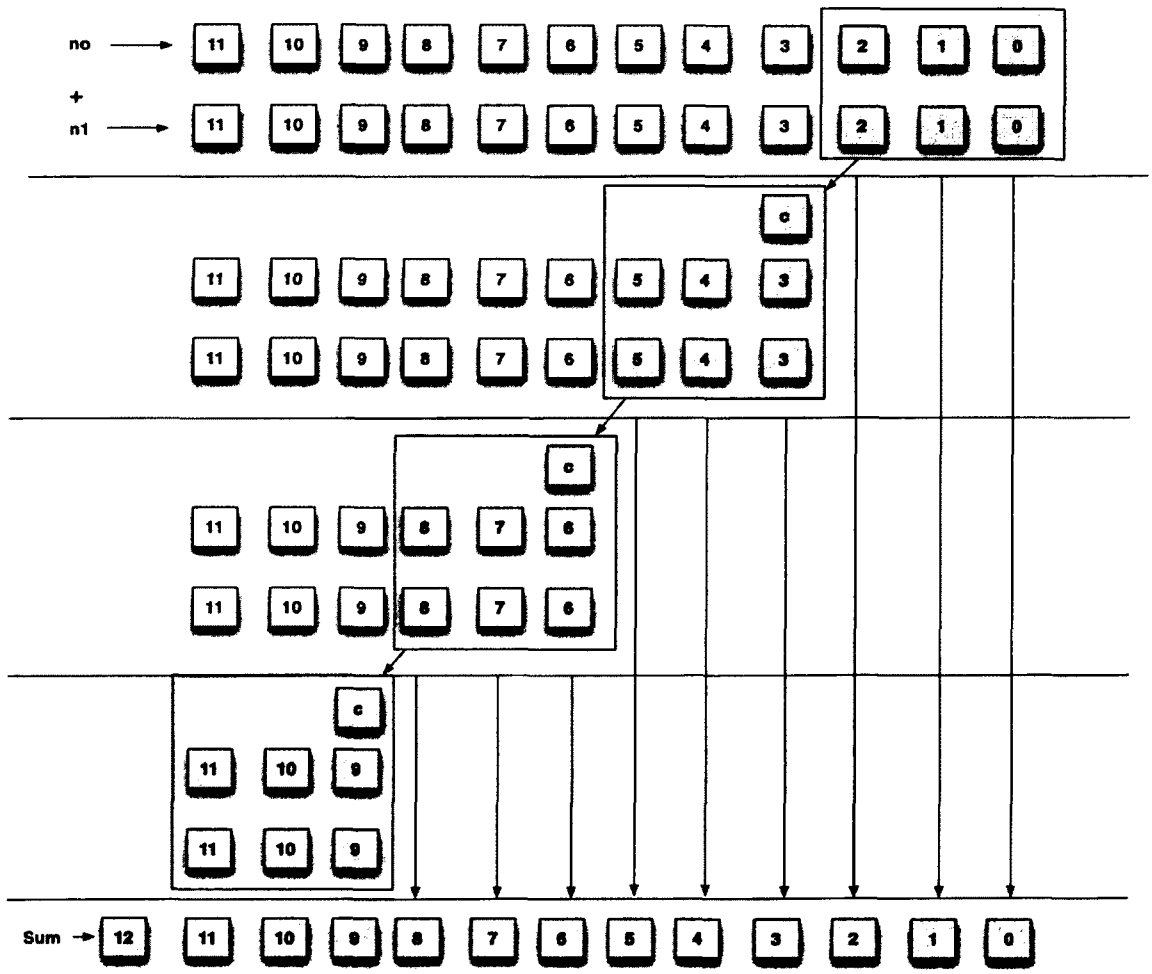

Figure 5.7: Pipelined Addition of 'Two' 12bit Data

5.4.1.5 Parallel-Pipelined Adder

The data width of the output data coming from the internal register is 8 bits. Since we replicate the MSB to avoid overflow, in our case we replicate the MSB 4 times, so now the input coming into the parallel adder [Piur 96] would be a 12 bit data. So we need a signed parallel adder that adds eight signed input numbers each of width 12 bits and delivers the sum of these numbers as output. Even thought in our case we only need to add three numbers each of width 12 bits (since we only have 3 images in the database), designing a parallel adder for 8 numbers, makes it more flexible, in case if we decide to add more images to our database our parallel adder would remain flexible and we can add up to a maximum of 8 images to our database. For now we only need to use the inputs img_1, img_2 and img_3 of the internal register (since our database has only 3 images), the rest (img_4 to img_8) are not used. Figure 5.8 illustrates the parallel-pipelined adder design.

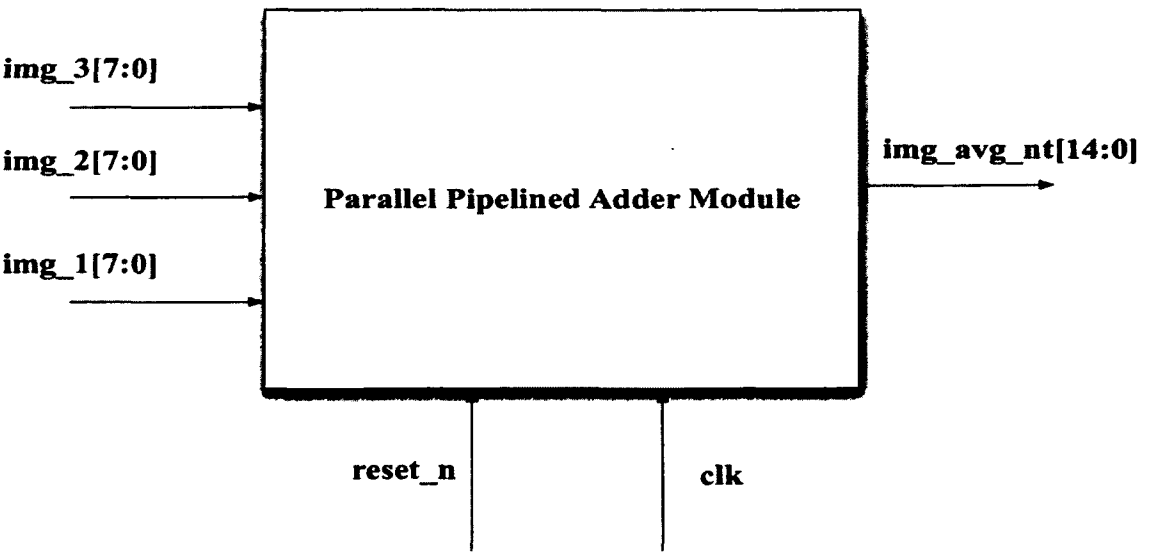

Figure 5.8: Parallel-Pipelined Adder

This signed addition can be realized with seven two input adders and 14 pipeline stages, in the first stage we have Eight numbers of 12bit two's complement adder to add all the eight numbers, they work concurrently, there by speeding up the process, they have pipeline registers internally, the clock input is marked as clk1, clk2, etc., corresponding to the internal pipelined register, we will add the LSBs at the first clock pulse and the MSBs at the next clock pulse along with the carry generated from the LSB, in the second stage we will add the four outputs, each of size 13bits generated from the first stage, In the second stage two numbers of two input adders are used, First the LSBs and then the MSBs are added subsequently with the arrival of each rising edge clock pulse, finally in the last stage, we will add the two inputs of size 14bits, which was obtained from the second stage, and then in the same fashion we add the MSBs along with the carry generated from the LSB addition, finally to produce a 15 bit final sum, we have 3 major stages in our design, each stage adds one bit growth, so we finally end up with 15 bits.

The Three image vectors are added pixel by pixel, each pixel is 8 bit wide.

$$Img1[7:0] + Img[7:0] + Img[7:0] = img\_avg\_nt[14:0] \qquad (5.9)$$

Table 5.3 illustrates the HDL port names of the parallel-pipelined adder, its direction and description.

Table 5.3: Parallel-Pipelined Adder - Port Name and Description

HDL Port Names	Direction	Description
img_n [7:0]	Input	Data from internal register
img_avg_nt [14:0]	Output	Output Sum

5.4.1.6 Divider Module

After finding the output of the parallel adder, we need to divide the result by 3 (M = 3, since we have 3 known images in our database) to obtain the average.

The output img_avg_nt [14:0] from the parallel-pipelined adder is sent to the divide by 3-module; the output obtained from this module is the average vector img_avg[14:0], the start and end of the average signal is signaled by img_start and img_end respectively, the img_valid signal validates the average output from this module. Table 5.4 illustrates the HDL port names of the divide by-3 module, its direction and description.

Table 5.4: Divide by 3-Module - Port Name and Description

HDL Port Names	Direction	Description
img_avg_nt [14:0]	Input	Sum of image vectors
img_avg[14:0]	Output	Average of image vectors
img_valid	Output	Validates the signal
img_start	Output	Indicates start of the signal
img_end	Output	Indicates end of the signal

Figure 5.9 illustrates the block diagram of the Divide by-3 module.

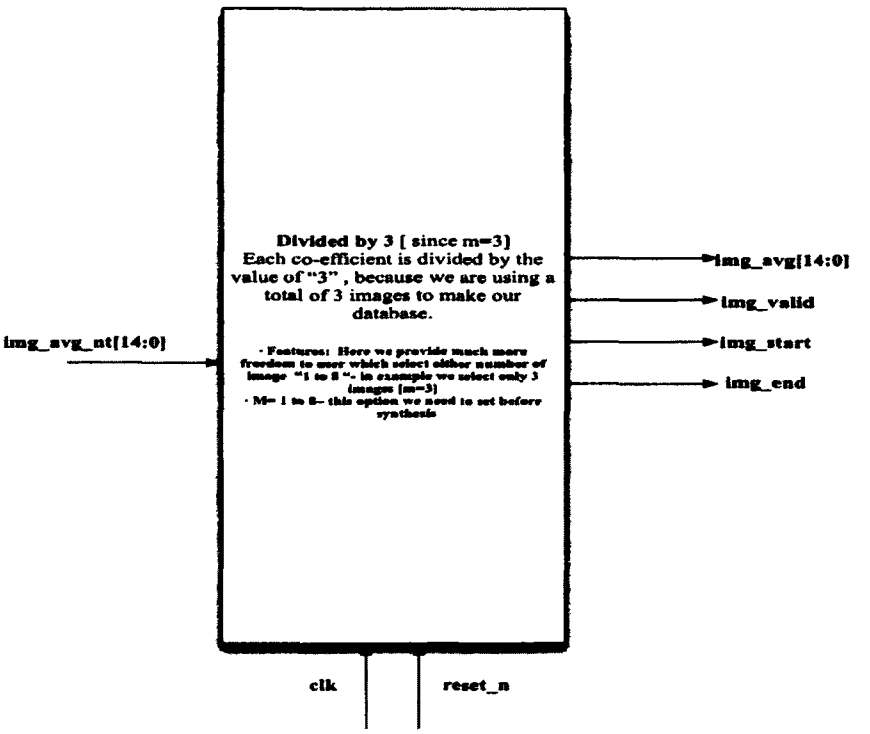

Figure 5.9: Divide by 3-Module

The average image resulting vector (img_avg) is stored in an internal register, the average vector Ψ is of dimension 10304×1, here we again use the controller to control the address location and position to read and write data.

Figure 5.10 illustrates the block diagram of the average information module.

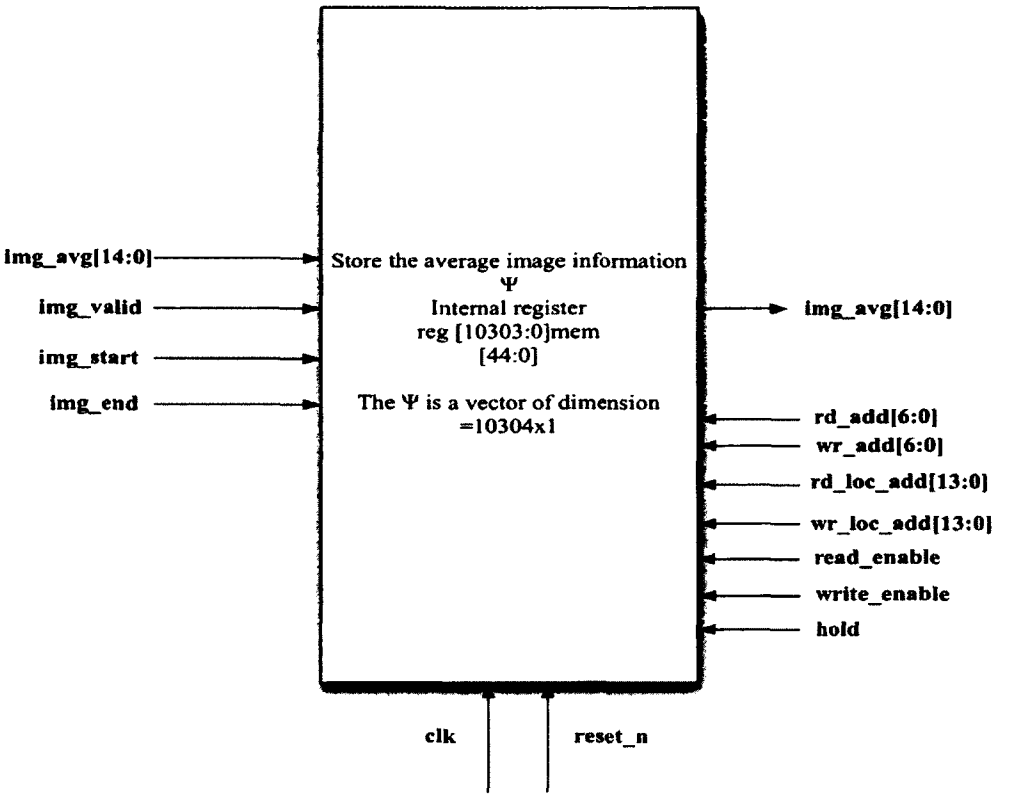

Figure 5.10: Average Information Module

The average information (img_avg) from the internal register (Ψ) has a data width of 15 bits; because of the bit growth most of the MSBs would be zero, and because division is involved there is a possibility of fractional result, since we are only using the integer values, we are rounding off each coefficient to its nearest integer value represented by 10bits.

The output signals from RAM1, RAM2 and RAM3 are do_1, do_2 and do_3 each are of 8 bits, since we would use these signals along with the average information signal, we are normalizing all these signals to 10bits.

Figure 5.11 illustrates the block diagram of the Normalization module.

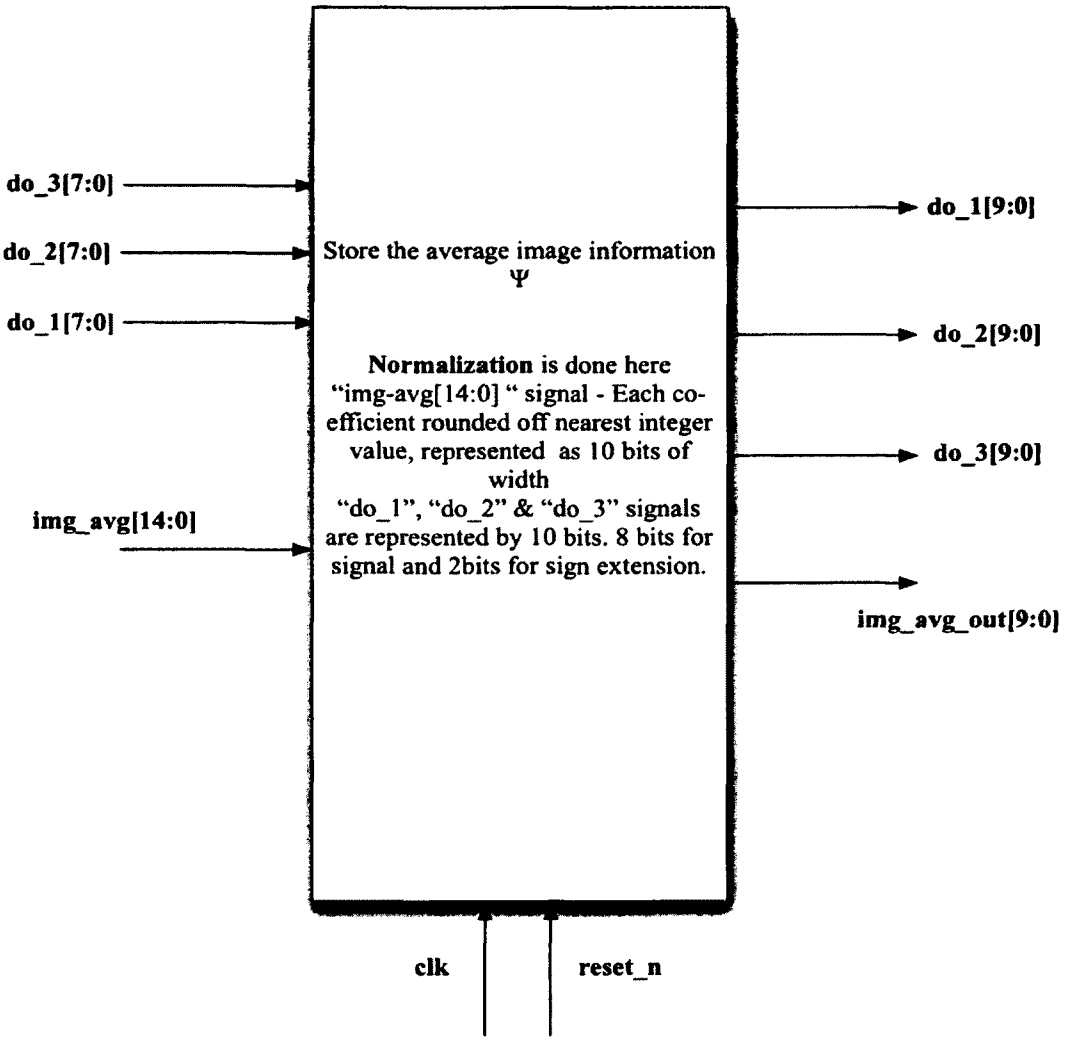

Figure 5.11: Normalization Module

Figure 5.12 illustrates the block diagram of the Controller to control the location and position selection for the average information module.

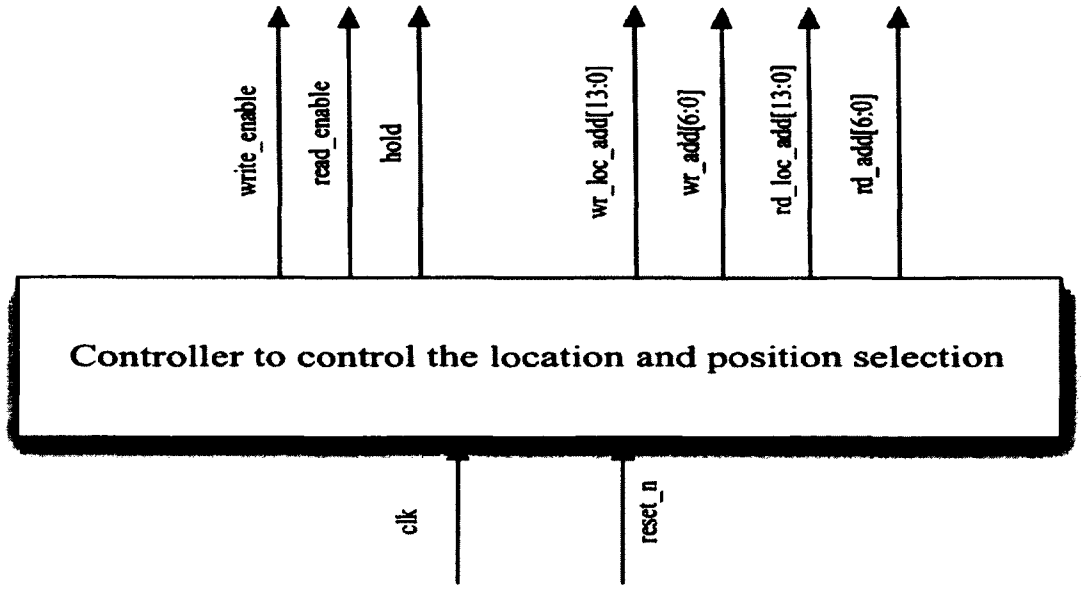

Figure 5.12: Controller to Control the Address Location and Position

Table 5.5 illustrates the HDL port names of the controller to control the address location and position, its direction and description.

Table 5.5: Controller for Address Location and Position - Port Names and Description

HDL Port Names	Direction	Description
write_enable	Output	Enables write signal
read_enable	Output	Enables read signal
hold	Output	Holds the data transfer
wr_add	Output	Write address
wr_loc_add	Output	Write address row location
rd_add	Output	Read address
rd_loc_add	Output	Read address row location

Model Calculation Continued

Step 4: Find the difference face

Each face differs from the average face, which is called as the difference face.

$$\Phi_i = \Gamma_i - \Psi \quad where\ i = 0, 1 \square\ M \tag{5.10}$$

M = 3, since we have 3 known images in our database.

$$\Phi_1 = \Gamma_1 - \Psi \tag{5.11}$$

$$\Phi_2 = \Gamma_2 - \Psi \tag{5.12}$$

$$\Phi_3 = \Gamma_3 - \Psi \tag{5.13}$$

Subtracting the mean face from the known face images 1, 2 and 3, we get the following difference faces.

$$\Phi_1 = \begin{bmatrix} -13 \\ -5 \\ -31 \\ 3 \\ -53 \\ -11 \\ 21 \\ -2 \\ -3 \end{bmatrix}_{9\times1} \quad \Phi_2 = \begin{bmatrix} -3 \\ 20 \\ 9 \\ 2 \\ 32 \\ -13 \\ -14 \\ 21 \\ 31 \end{bmatrix}_{9\times1} \quad \Phi_3 = \begin{bmatrix} 17 \\ -15 \\ 21 \\ -5 \\ 22 \\ 25 \\ -7 \\ -19 \\ -29 \end{bmatrix}_{9\times1} \tag{5.14), (5.15), (5.16}$$

$$A = [\Phi_1, \Phi_2 \square\ \Phi_M] \quad here\ M = 3$$

$$\therefore A = [\Phi_1, \Phi_2, \Phi_3] \tag{5.17), (5.18}$$

The difference face vectors are concatenated to obtain the 'A' matrix of dimension 9 × 3

$$A = \begin{bmatrix} -13 & -3 & 17 \\ -5 & 20 & -15 \\ -31 & 9 & 21 \\ 3 & 2 & -5 \\ -53 & 32 & 22 \\ -11 & -13 & 25 \\ 21 & -14 & -7 \\ -2 & 21 & -19 \\ -3 & 31 & -29 \end{bmatrix}_{9\times3} \quad (5.19)$$

Architecture Design for Step 4

Difference face is the difference between the average face and the known face, since we have 3 known faces in the database, we would get 3 difference-face vectors, namely Φ_1, Φ_2 *and* Φ_3.

As seen in the above model calculation, we would encounter negative values during eigenfaces process, taking this into consideration while designing the architecture, we would use a bit check module to convert the average face data and individual face data into its equivalent two's complement form and then we add the individual face and the average face together using a parallel-pipelined adder.

5.4.1.7 Bit check Module

In the bit check Module, we convert the incoming data into its equivalent two's complement form. Then we finally send this data to the parallel adder, to avoid any overflow in the parallel adder stage, we would add a two-bit sign extension in the bit

check module, and the inputs for the bit check module are the average face vector and the individual face vector. Figure 5.13 illustrates the block diagram for the bit check module.

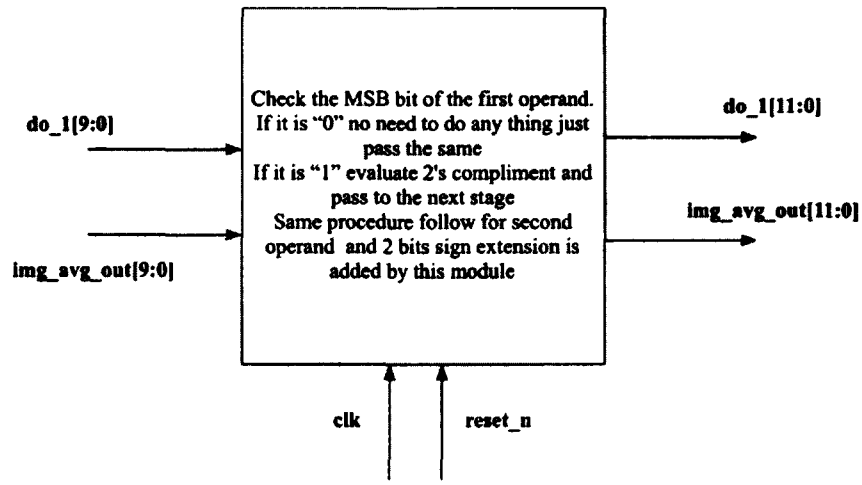

Figure 5.13: Bit Check Module

5.4.1.8 Pipelined adder

Once we have the outputs from the bit check module, all we have to do is to add them using a pipelined adder, the parallel adder that were using here is similar to the one that we have used earlier in this architecture, but we would require only one stage, and in that stage we are adding the two's complement of the known image 1 (do_1) and the average image (img_avg_out) to obtain the difference face 1 - Φ_1 (diff_face_1), similarly we can obtain difference face 2 - Φ_2 (diff_face_2) and difference face 3 - Φ_3 (diff_face_3) by using known image 2 and 3 respectively.

Figure 5.14 illustrates the block diagram for the pipelined adder module.

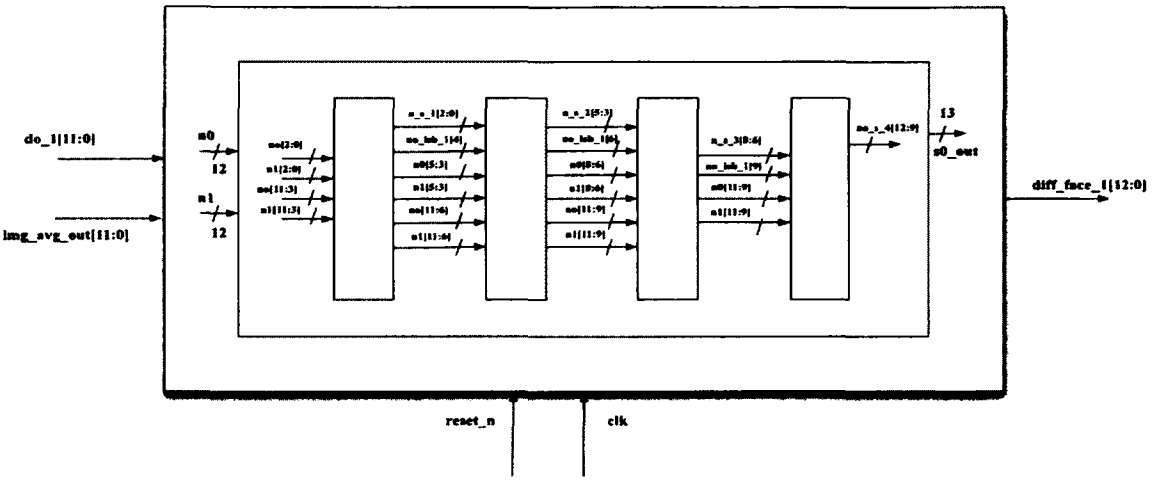

Figure 5.14: Pipelined Adder

Figure 5.15 illustrates the block diagram for the difference face module.

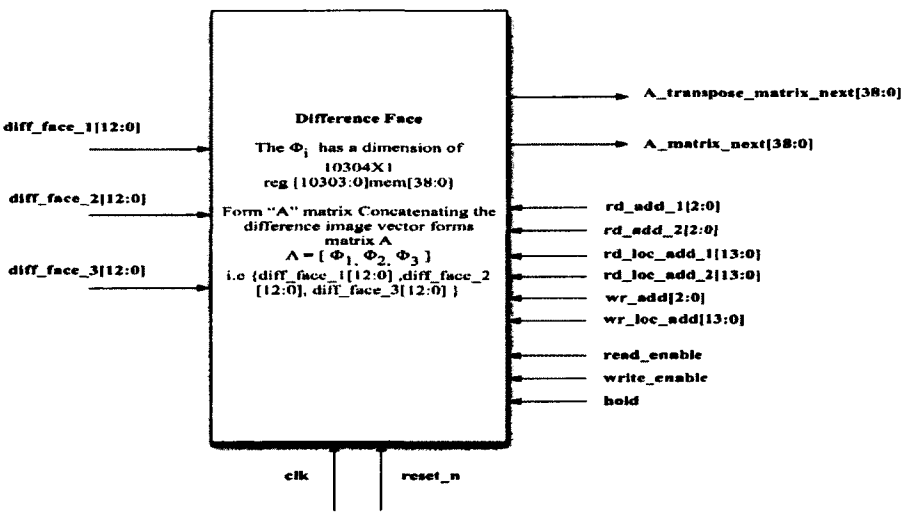

Figure 5.15: Difference Face

The difference faces are stored in an internal register; each difference face is a vector of size 10304×1, since we have three images in our database, we would obtain three difference faces, concatenating all 3 vectors would form a matrix of size 10304×3, this matrix is represented by 'A', the controller to control the location and position would provide the address to read and write the data to and from the internal register.

Model Calculation Continued

Step 5: Obtain the Covariance matrix C

$$C = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = A\,A^T \tag{5.20}$$

$$Where\ A = [\Phi_1, \Phi_2 \square\ \Phi_M] \tag{5.21}$$

The covariance matrix C is solved to obtain the eigenvectors u_i and eigenvalues λ_i. In our model calculation, the dimension of 'A' matrix is 9×3, so multiplying $A * A^T$ would give us the covariance matrix of dimension 9×9, which would yield 9 eigenvalues and 9 eigenvectors, instead we can use a mathematical backdoor to reduce he matrices as follows, lets take

$$L = A^T A \tag{5.22}$$

$$=> L = A^T_{3\times9}\ A_{9\times3} \tag{5.23}$$

$$=> L_{3\times3} \tag{5.24}$$

Now we have reduced the matrix down from 9×9 to 3×3, this step is very important, because looking at the bigger picture, we would have huge values, for example if we have 10 images of dimension 92×112, then we will end up with the following data.

$$= A_{10304\times10} * A^{T}_{10\times10304}$$
$$=> C_{10304\times10304} \qquad (5.25)$$

Solving the covariance matrix C with dimension 10304 × 10304 would yield 10304 eigenvectors and 10304 eigenvalues; this is a very huge value and could possibly crash a processor, so it is essential to solve the matrix 'L' instead of matrix C.

$$L = A^{T}A \qquad (5.26)$$

$$L = \begin{bmatrix} 4548 & -2316 & -2275 \\ -2316 & 3285 & -993 \\ -2275 & -993 & 3340 \end{bmatrix}_{3\times3} \qquad (5.27)$$

Architecture design for Step 5

Once we have our difference faces, we concatenate them to obtain the 'A' matrix; this matrix is stored in an internal register, to obtain 'A^T' we should read the rows of 'A' matrix as column and the columns as rows, then we multiply A^T and A.

5.4.1.9 Multiplier

The multiplier we use here is a normal multiplier rather than a pipeline multiplier, since the multiplier and the multiplicand are of 39 bits each, this would require excessive pipelining, which would increase latency and also would require more resources (circuit elements) thereby increasing the chip area. Figure 5.16 illustrates the block diagram for the multiplier module.

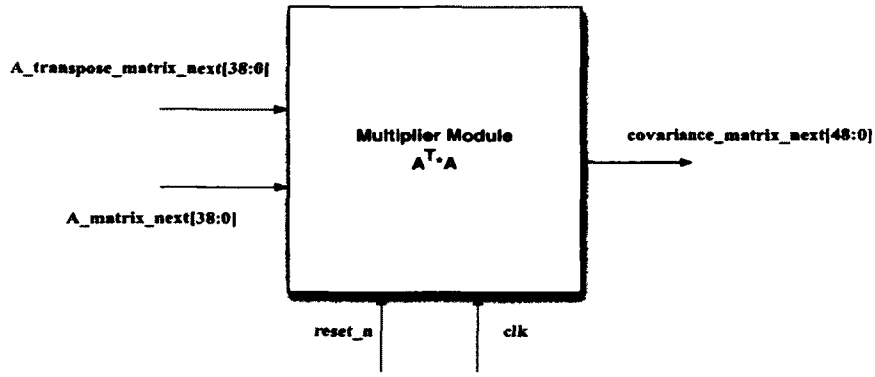

Figure 5.16: Multiplier Module - ('A' matrix and 'A^T' matrix)

In the multiplier module, multiplying A^T and A would give us L matrix, once we have the 'L' matrix, each and every element of the matrix is normalized using a normalization module. Figure 5.17 illustrates the block diagram of the Normalization module for L-Matrix.

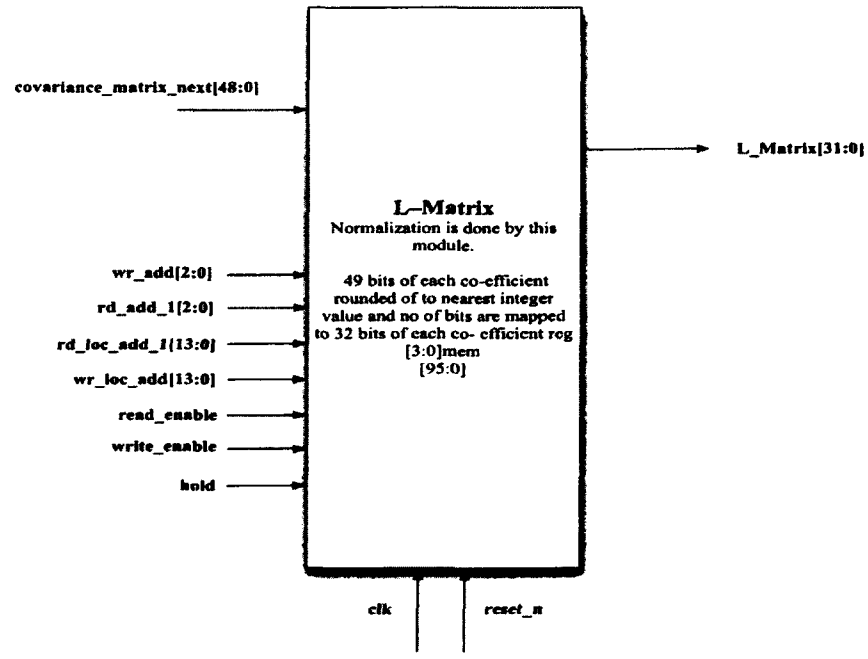

Figure 5.17: L Matrix - Normalization Module

Model Calculation Continued

Step 6: Find the eigenvectors u_i and eigenvalues λ_i of C

Now if, $Lv_i = \lambda_i v_i$ (5.28)

where v_i is the eigenvectors of L and λ_i are eigenvalues of L, then

Multiplying A on both sides of the above equation gives us

$$=> A L v_i = \lambda_i A v_i \quad (5.29)$$

$$=> A A^T A v_i = \lambda_i A v_i \quad (5.30)$$

$$=> C A v_i = \lambda_i A v_i \quad (5.31)$$

Hence $u_i = Av_i$ and λ_i are respectively the M eigenvectors and eigenvalues of C.

To find the eigenvalues and eigenvectors of C, we should first find the eigenvectors v_i of L Matrix.

$$L = \begin{bmatrix} 4548 & -2316 & -2275 \\ -2316 & 3285 & -993 \\ -2275 & -993 & 3340 \end{bmatrix}_{3\times3} \quad (5.32)$$

To find the eigenvalues and eigenvectors [Hami 1990] of L matrix, we should introduce λ and an Identity matrix 'I' and then we must find the determinant for $[L - \lambda . I]$, this would give us a cubic equation in terms of λ,

$$\lambda^3 + 2077\lambda^2 + 29576870\lambda - 1.06\times10^{10} = 0 \quad (5.33)$$

Solving the above equation for λ would give us 3 eigenvalues,

$$\lambda_1 = 6866, \lambda_2 = 4306, \lambda_3 = 1 \quad (5.34)$$

Then we have to substitute each eigenvalue in $[L - \lambda . I]$ and multiply it with vector v,

$(L - \lambda_1 . I)v_1$.

Where v_1 is $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$, solving $(L - \lambda_1 . I)v$, would give us three simultaneous equations in terms of x, y and z, solving them would give the eigenvector of λ_1, similarly solving the matrix and the resulting simultaneous equations for $(L - \lambda_2 . I)v_2$ and $(L - \lambda_3 . I)v_3$ would give us the eigenvectors of $\lambda_2$and λ_3 respectively.

Finally the eigenvalues and their corresponding eigenvalues are as follow

$$\lambda_1 = 6866,\ v_1 = \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix} \qquad (5.35)$$

$$\lambda_2 = 4306,\ v_2 = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \qquad (5.36)$$

$$\lambda_3 = 1,\ v_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad (5.37)$$

Architecture design for Step 6

5.4.1.10 MATLAB Module

The eigenvalues and eigenvectors are to be found from the 'L' Matrix. Solving this matrix for eigenvalues and eigenvectors using Verilog HDL for FPGA implementation would be a difficult task. The above model calculations clearly explain the intricate steps involved in solving them. This is currently being researched [Brav 08] [Brav 06]. Due to lack of time, we could not complete its implementation, so we have used MATLAB to

solve the covariance matrix to obtain the eigenvalues and the eigenvectors. Figure 5.18 illustrates the block diagram of the MATLAB module.

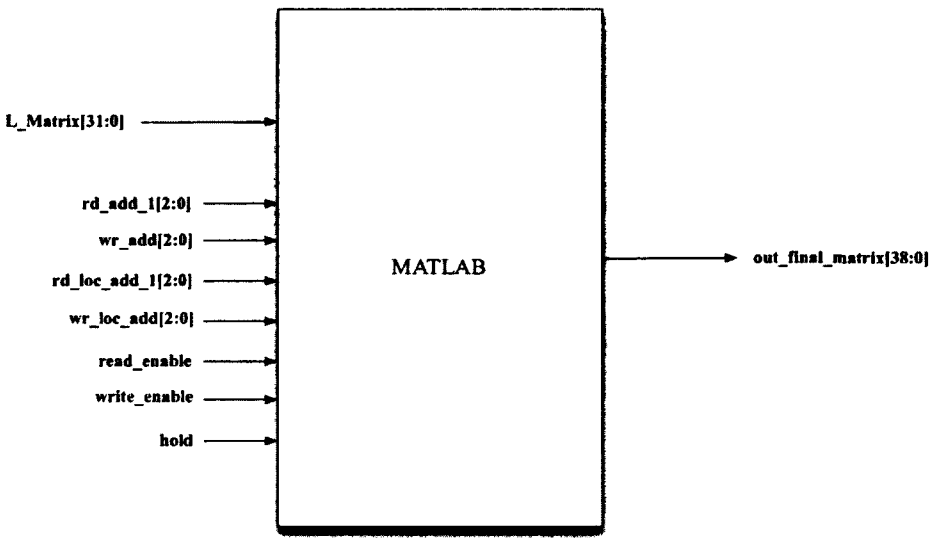

Figure 5.18: MATLAB Module

The eigenvalues λ_i and eigenvectors v_i are found using MATLAB. This concludes the architecture for Phase I.

5.4.2 Phase II

Figure 5.19 illustrates Phase II of Architecture

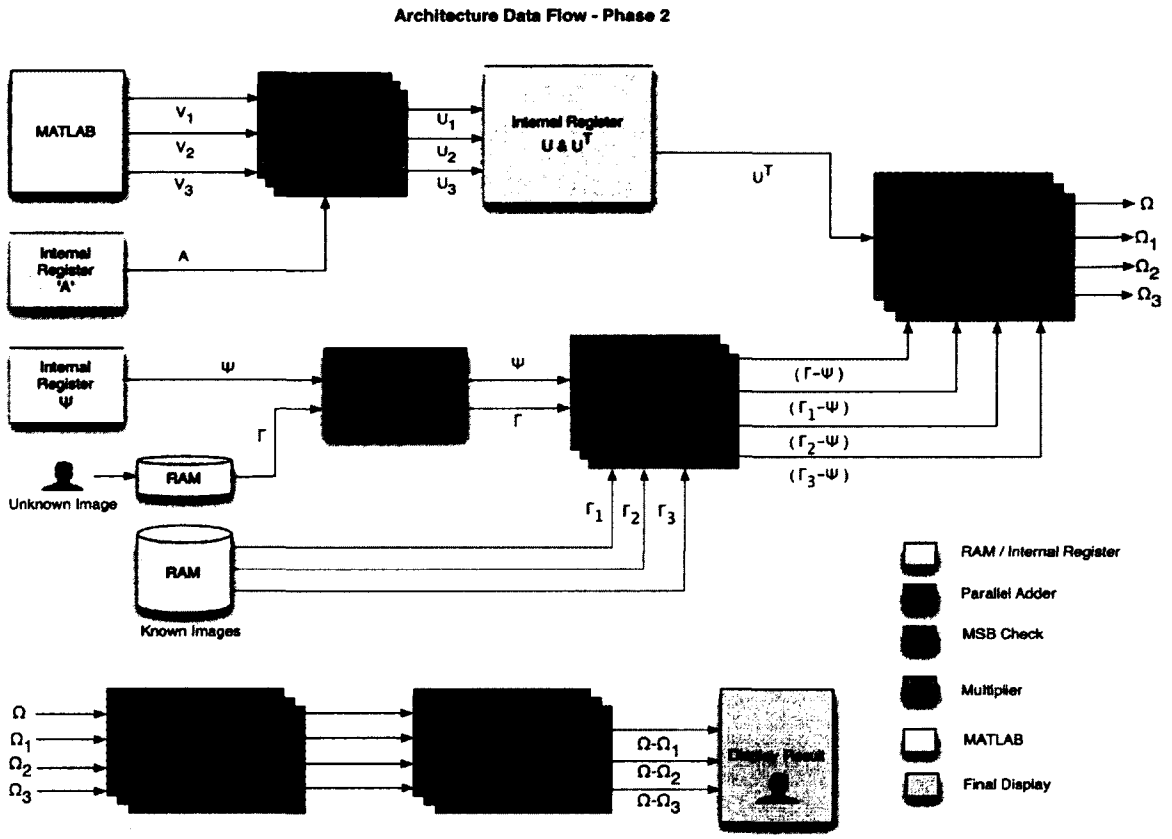

Figure 5.19: Architecture Date Flow - Phase II

Model Calculation Continued

Step 7: Find the eigenvector u_i of C

$$u_i = A.v_i \qquad (5.38)$$

Multiplying the eigenvectors v_i of L with A, would give us the eigenvectors u_i of C, generally the eigenvectors with the smallest eigenvalues could be neglected, since they

don't contribute to too much to the eigenface. But in our case since we only have 3 eigenvectors, there is no need to neglect the eigenvector with lowest eigenvalue, each of these eigenvectors are normalized, scaled by 255 and then projected on the image space, they would render a ghostly face image called as 'Eigenface', since we have 3 eigenvectors, we would have 3 eigenfaces. The following eigenvectors u_i given below are normalized and are scaled by 255.

$$u_1 = \begin{bmatrix} 48 \\ 18 \\ 115 \\ -10 \\ 199 \\ 41 \\ -79 \\ 5 \\ 8 \end{bmatrix}_{9\times1} \quad u_2 = \begin{bmatrix} 53 \\ -94 \\ 31 \\ -18 \\ -25 \\ 102 \\ 18 \\ -110 \\ -163 \end{bmatrix}_{9\times1} \quad u_3 = \begin{bmatrix} 51 \\ 0 \\ -51 \\ 0 \\ 51 \\ 51 \\ 0 \\ 0 \\ -51 \end{bmatrix}_{9\times1} \qquad (5.39), (5.40), (5.41)$$

$$u = [u_1, u_2 \ldots u_{M'}]_{N^2\times M'} \quad where\ M' \leq M \qquad (5.42)$$

$$u = \begin{bmatrix} 48 & 53 & 51 \\ 18 & -94 & 0 \\ 115 & 31 & -51 \\ -10 & -18 & 0 \\ 119 & -25 & 51 \\ 41 & 102 & 51 \\ -79 & 18 & 0 \\ 5 & -110 & 0 \\ 8 & -163 & -51 \end{bmatrix}_{9\times3} \qquad (5.43)$$

Architecture design for Step 7

5.4.2.1 Multiplier

We need to multiply 'A' matrix with the eigenvectors v_1, v_2 and v_3, we are using the same multiplier module as used in Phase I of this Architecture. Figure 5.20 illustrates the block diagram of the multiplier module.

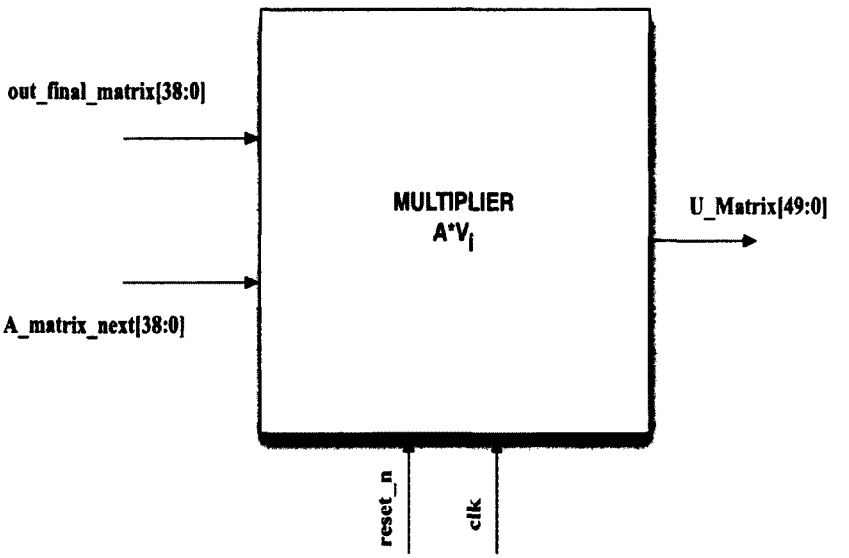

Figure 5.20: Multiplier Module - ('A' Matrix and Eigenvectors V_1, V_2 and V_3)

The resulting $'U'$ matrix is then stored in an internal register and since we would be only be using with $'U^T'$ matrix, it is better to take transpose of $'U'$ matrix and save it in the internal register, the controller to control the location and position provides the address for reading from and writing to the internal register. Figure 5.21 illustrates the block diagram of the Internal Register for U and U^T Matrix.

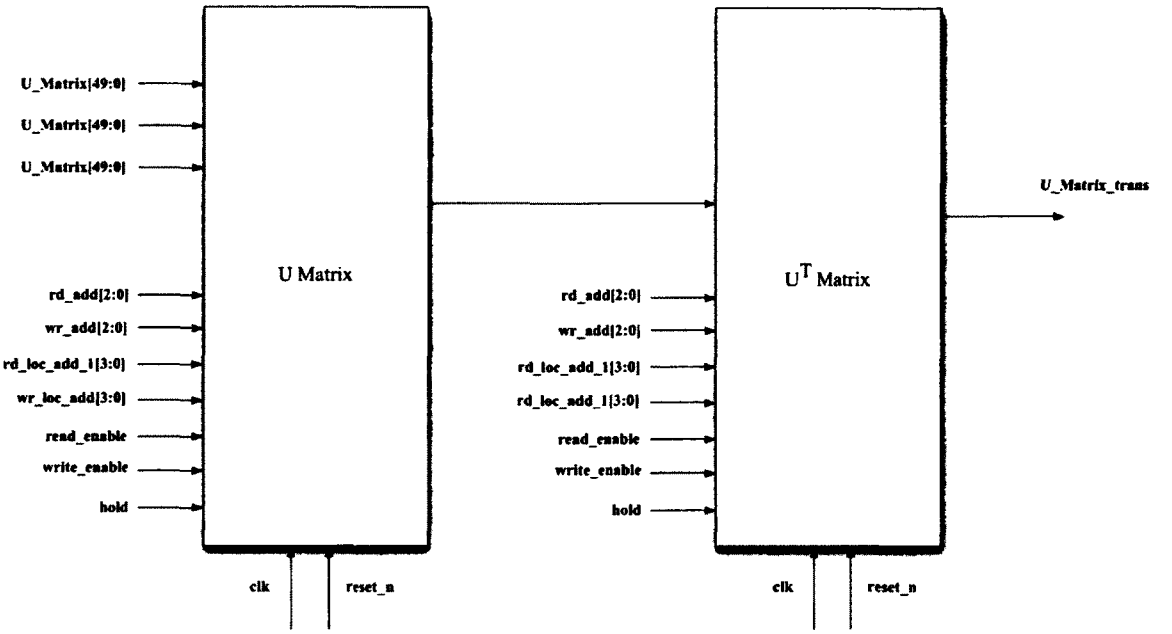

Figure 5.21: Internal Register ('U' and 'U^T' Matrix)

Model Calculation Continued

Step 8: Face classification

Once the eigenfaces were created, a new face image Γ can be transformed into its eigenface component by a simple operation.

$$\Omega = U^T(\Gamma - \Psi) = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}_{k\times 1} \qquad (5.44)$$

where $k = 1,2 \ldots M'$

The weight $w_i \in \Omega$ describes the contribution of each eigenface in representing the input face image, once the weight vector (Ω) of the test image Γ is found; we also find the weight vector of the known images from the database, since we have 3 known images,

we will get (Ω_1, Ω_2, Ω_3) and then we use the Euclidean distance classifier to find the smallest distance between the weight vectors of the test face and the known faces in the database.

$$\epsilon_{rec} = \min\|\Omega - \Omega_i\| \qquad (5.45)$$

If $\epsilon_{rec} < \theta_{rec}$, where θ_{rec} is chosen heuristically, then we can say that the input test image is recognized as the image with which it gives the lowest score.

Since in our model calculation we have assumed smaller matrices to be our known images, here we are assuming our test image to be a similar matrix, for convenience, we are assuming the test image to be the same as the matrix of known image1.

$$\Gamma = \begin{bmatrix} 10 \\ 35 \\ 40 \\ 11 \\ 5 \\ 6 \\ 50 \\ 22 \\ 36 \end{bmatrix}_{9\times1} \quad \Psi = \begin{bmatrix} 23 \\ 40 \\ 71 \\ 8 \\ 58 \\ 17 \\ 29 \\ 24 \\ 39 \end{bmatrix}_{9\times1} \quad (\Gamma - \Psi) = \begin{bmatrix} -13 \\ -5 \\ -31 \\ 3 \\ -53 \\ -11 \\ 21 \\ -2 \\ -3 \end{bmatrix}_{9\times1} \qquad (5.46), (5.47), (5.48)$$

Multiplying U^T matrix with $(\Gamma - \Psi)$ will give us Ω

$$\Omega = \begin{bmatrix} -17000 \\ 56 \\ -2193 \end{bmatrix}_{3\times1} \qquad (5.49)$$

Multiplying U^T matrix with $(\Gamma_1 - \Psi)$ will give us Ω_1

$$\Omega_1 = \begin{bmatrix} -17000 \\ 56 \\ -2193 \end{bmatrix}_{3\times1} \quad (5.50)$$

The weight vectors Ω and Ω_1 are similar, since we have assumed the test image and the known image 1 to be the same.

Multiplying U^T matrix with $(\Gamma_2 - \Psi)$ will give us Ω_2

$$\Omega_2 = \begin{bmatrix} 8525 \\ -11537 \\ -1224 \end{bmatrix}_{3\times1} \quad (5.51)$$

Multiplying U^T matrix with $(\Gamma_3 - \Psi)$ will give us Ω_3

$$\Omega_3 = \begin{bmatrix} 8640 \\ 11743 \\ 3672 \end{bmatrix}_{3\times1} \quad (5.52)$$

The Euclidean distance classifier is applied to find the distance between the weight vectors $\epsilon_{rec} = \min\|\Omega - \Omega_i\|$, we would get

$$\|\Omega - \Omega_1\| => \epsilon_{rec} = 0 \quad (5.53)$$

$$\|\Omega - \Omega_2\| => \epsilon_{rec} = 28051 \quad (5.54)$$

$$\|\Omega - \Omega_3\| => \epsilon_{rec} = 28781 \quad (5.55)$$

The minimum distance is '0' for known image 1; hence the test image(Γ) is recognized as known image 1 (Γ_1).

Architecture design for Step 8

The new face image Γ can be transformed into its eigenface component by a simple operation

$$\Omega = U^T(\Gamma - \Psi) \qquad (5.56)$$

The test face that is stored in RAM 4 and the average face vector from the internal register are the inputs for the bit check module, once the two's complement representation of test face and average face vector are obtained using the bit check module, we then add these data using a parallel-pipelined adder module, then the result obtained is multiplied with U^T matrix to obtain Ω. Figure 5.22 illustrates the block diagram of the bit check module.

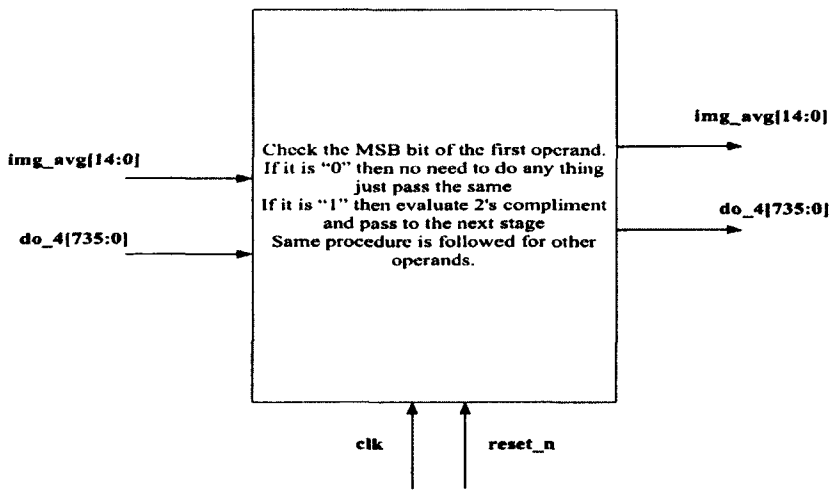

Figure 5.22: Bit Check Module

Figure 5.23 illustrates the block diagram of the Parallel-pipelined adder module.

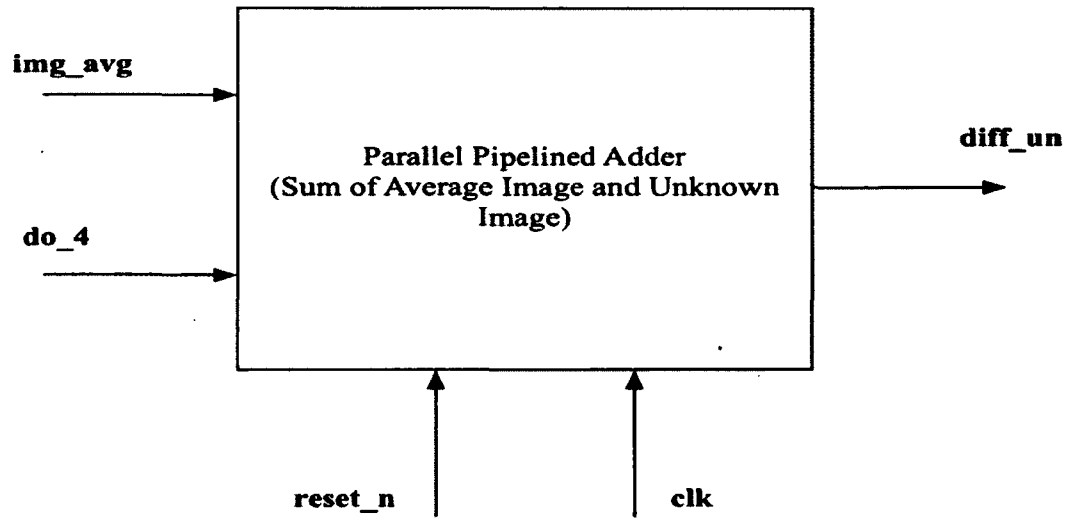

Figure 5.23: Parallel-Pipelined Adder - Average Face and Unknown Test Face

Once we have found the weight vector for the test image, we proceed to find the weight vectors of the known images, we start off by finding the two's complement for the average face vector (Ψ) and the known faces (Γ_1), (Γ_2) and (Γ_3) using the bit check module, and then using the parallel adder module to find the ($\Gamma_1 - \Psi$), ($\Gamma_2 - \Psi$) and ($\Gamma_3 - \Psi$). Figure 5.24 illustrates the block diagram of the Parallel-pipelined adder module.

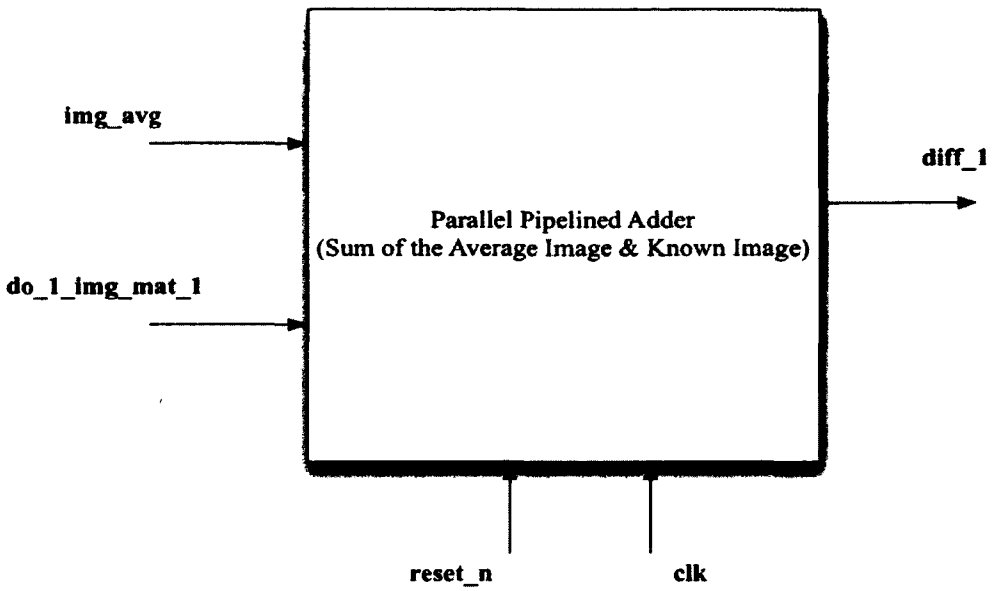

Figure 5.24: Parallel-Pipelined Adder - Average Face and Known Face

We use the multiplier module to multiply the result from the parallel-pipelined adder with U^T, which would give us the weight vectors of the known face Ω_1, Ω_2 and Ω_3. Figure 5.25 illustrates the block diagram of the multiplier module.

Once we have the weight vectors of the test image and the known images, we then find the Euclidean distance between the weight vectors to identify the weight vector that likes closest to the test weight vector, so we subtract the weight vectors of the known face with the weight vector of the test face, $(\Omega - \Omega_1)$, $(\Omega - \Omega_2)$, and $(\Omega - \Omega_3)$

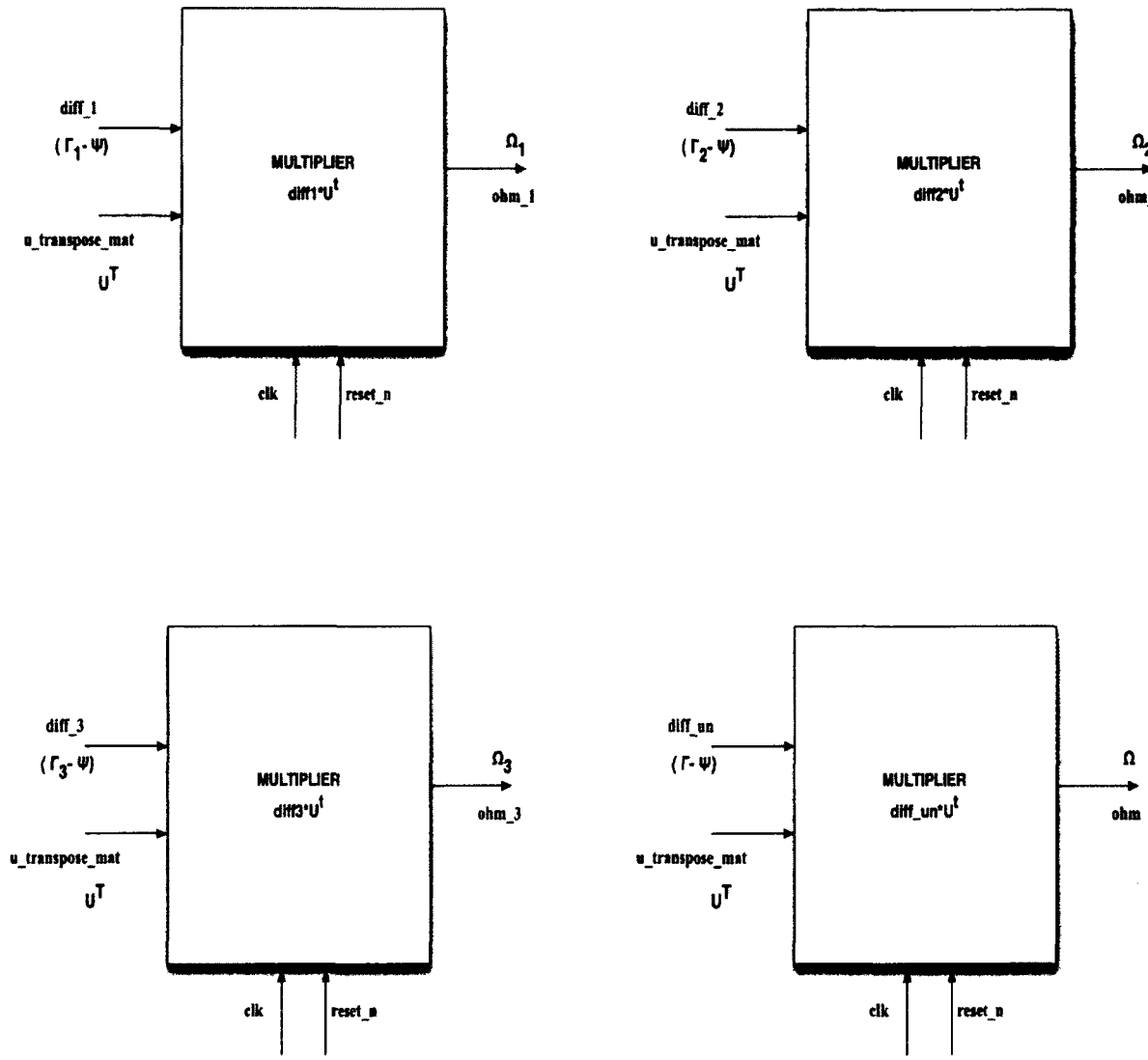

Figure 5.25: Multiplier Module ($U^T * (\Gamma_1 - \Psi)$)

The bit check module is used to convert the weight vectors into their 2's complement representation, and then the pipelined adder module is used to add the weight vector of the test face and the weight vector of the known face. Figure 5.26 illustrates the bit check module for the weight vectors and the Figure 5.27 illustrates the pipelined adder for the weight vectors.

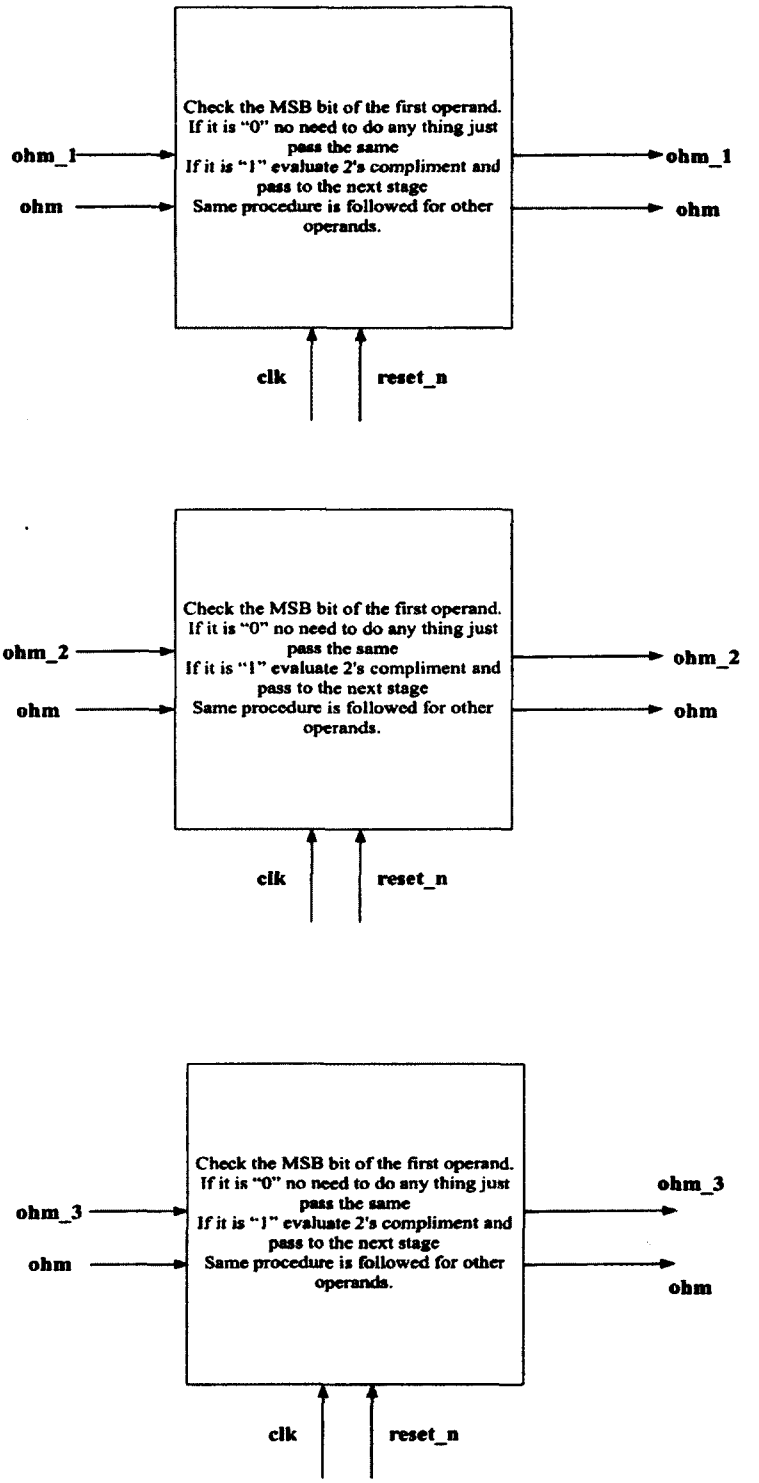

Figure 5.26: Bit Check Module for Weight Vectors

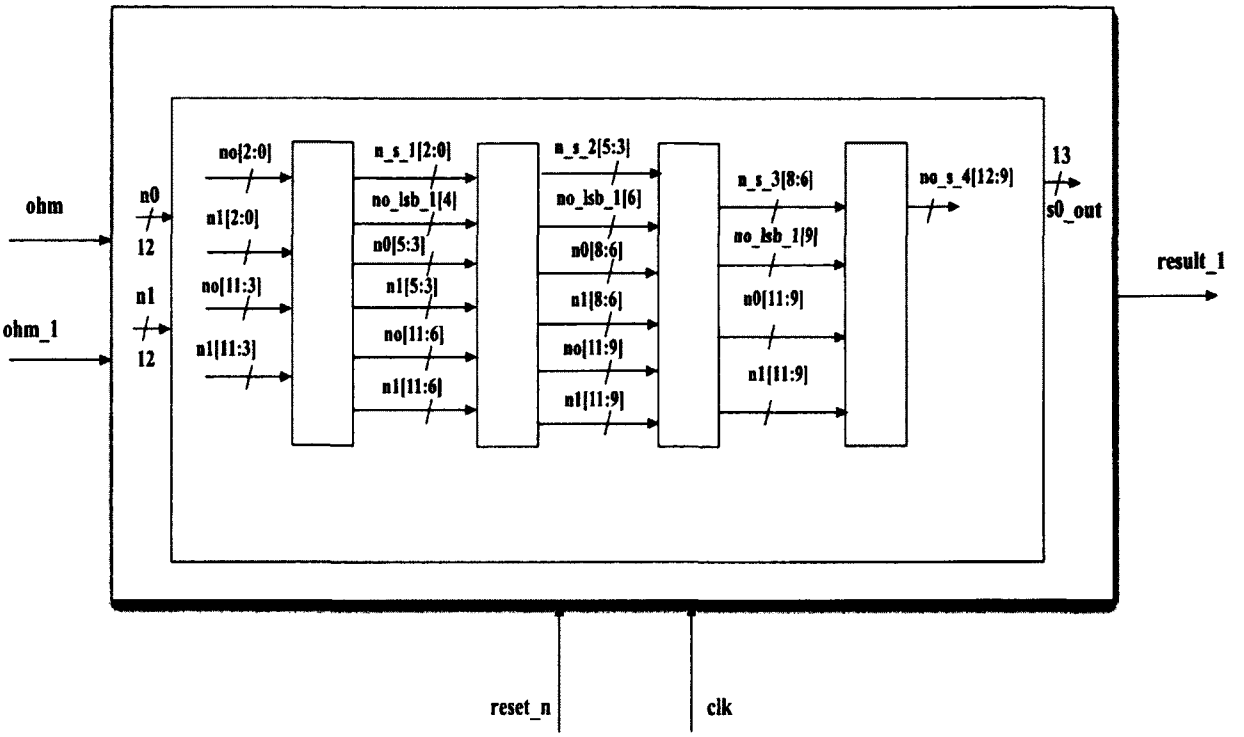

Figure 5.27: Pipelined Adder -for Weight Vectors

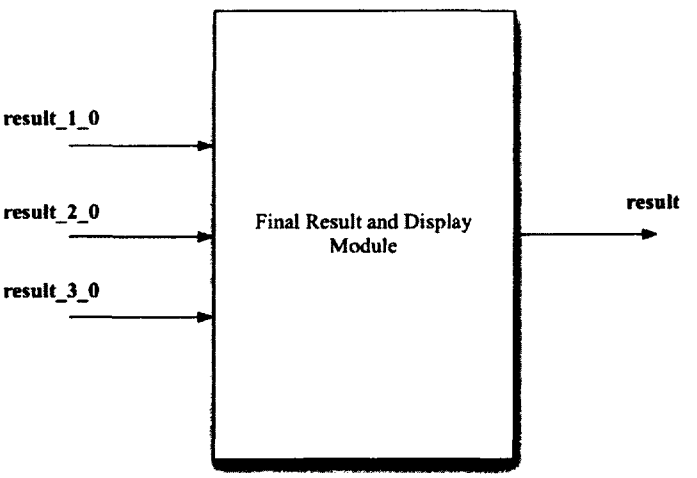

Figure 5.28: Compare and Display module

The weight vector from the known image database that has the lowest distance to the weight vector of the test face is identified in the compare and display module, this minimum distance is represented as ϵ_{rec}, when $\epsilon_{rec} < \theta_{rec}$, where θ_{rec} is chosen heuristically, then we can say that the input test image is recognized as the image with which it gives the lowest score, then this recognition result is displayed in the display module. Figure 5.28 illustrates the block diagram for the compare and display module.

This concludes the phase II of the architecture, combining the architecture of Phase I and Phase II gives us the complete architecture for implementing EA.

Architecture for FPGA implementation of EA was proposed and the individual modules were discussed. The simulation results of the basic modules are presented and the FPGA implementation issues of EA are discussed in the following chapter.

Chapter 6

Simulation and FPGA Implementation Issues

6.1 Introduction

This chapter presents the simulation results of the main modules of EA using ModelSim HDL Simulator and proceeds to discuss the feasibility of FPGA implementation. It concludes with discussion of issues related to FPGA implementation of eigenfaces architecture.

6.2 Simulation

Following the discussion of EA in chapter 4, and its architecture description in chapter 5, we discuss the verification of the functionality of main architecture modules using ModelSim HDL Simulator.

ModelSim SE PLUS 6.2 C version [Soft 11] was used to compile and simulate the Verilog modules and its corresponding test benches for the main architecture modules.

6.2.1 Basic Simulation Flow

To simulate the Verilog modules, we developed the required test benches. The basic simulation flow [Alte 11] [Mode 05] using ModelSim is shown in Figure 6.1

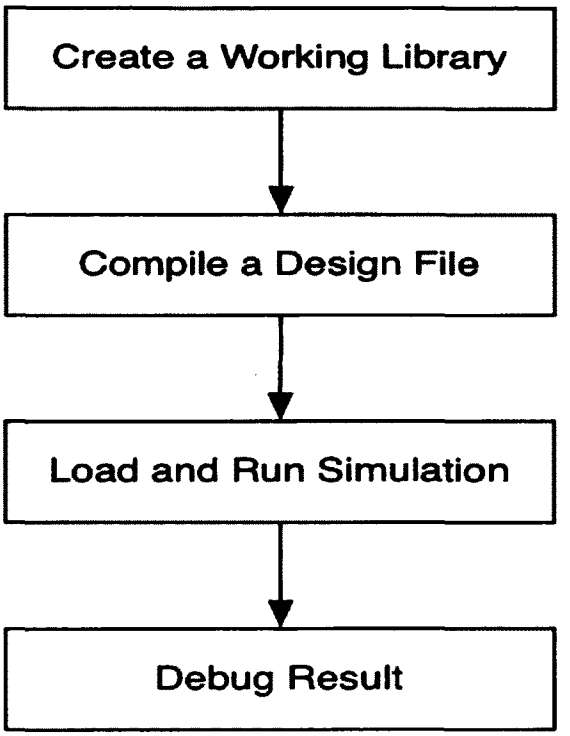

Figure 6.1: Basic Simulation Flow

- All the designs in ModelSim are compiled into the library, all the design simulation are under the 'work' library, this is the default system library.
- Once the work library is created, the design in the library is compiled, if there are any no errors, the compilation would be successful.
- After compiling the design, the simulator is loaded by invoking the simulator on the top-level module (Verilog), once the simulator is loaded successfully, the simulation is run.
- Incase the expected result is not obtained; the ModelSim debugging environment is used to track down the problem.

6.2.2 Simulation Results

All the modules in the eigenfaces architecture excluding the MATLAB module were developed in Verilog HDL. We now briefly describe the various models and their simulation. The list of modules developed is given below.

1. Parallel-Pipelined Adder Module
2. Divide by 3-Module
3. Read and Write Address Generator Module
4. Read and Write Data to the RAM Module

6.2.2.1 Parallel-Pipelined Adder Module

The simulation in this section is for parallel-pipelined adder, which was discussed in section 5.4.1.4. Figure 6.2 illustrates the interface diagram for the parallel-pipelined adder module

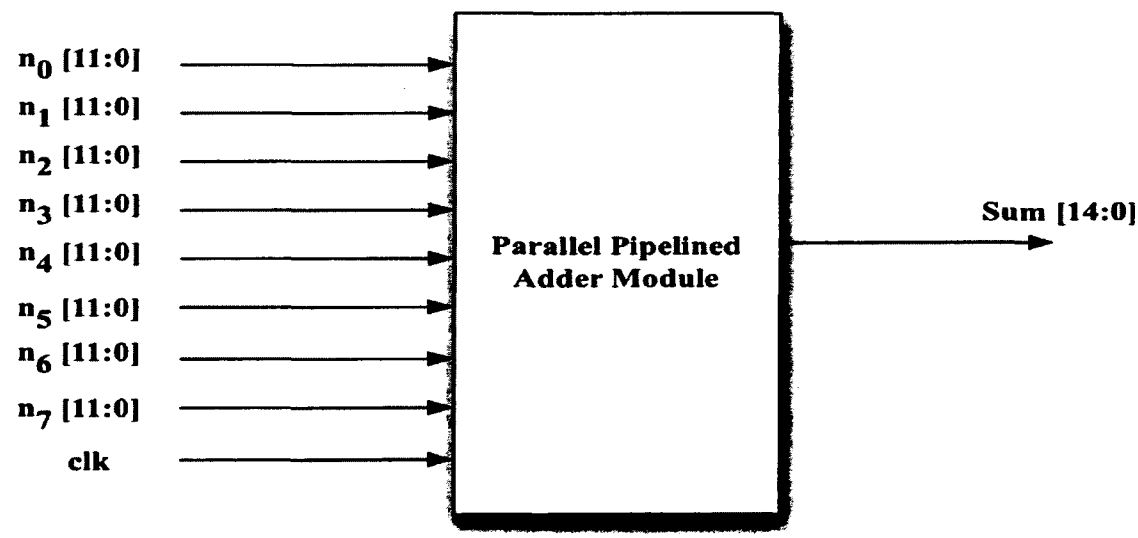

Figure 6.2: Parallel-Pipelined Adder Interface Module

The parallel-pipelined adder module is designed to add 8 different inputs (n_0-n_7), 12 bits each. The output is a 15-bit Sum. This module was tested with random 12 bit input values, and the simulation was successful.

6.2.2.2 Divide by 3-Module

The simulation in this section is for divide by 3-module, which was discussed in section 5.4.1.6. Figure 6.3 illustrates the interface diagram for divide by 3-module.

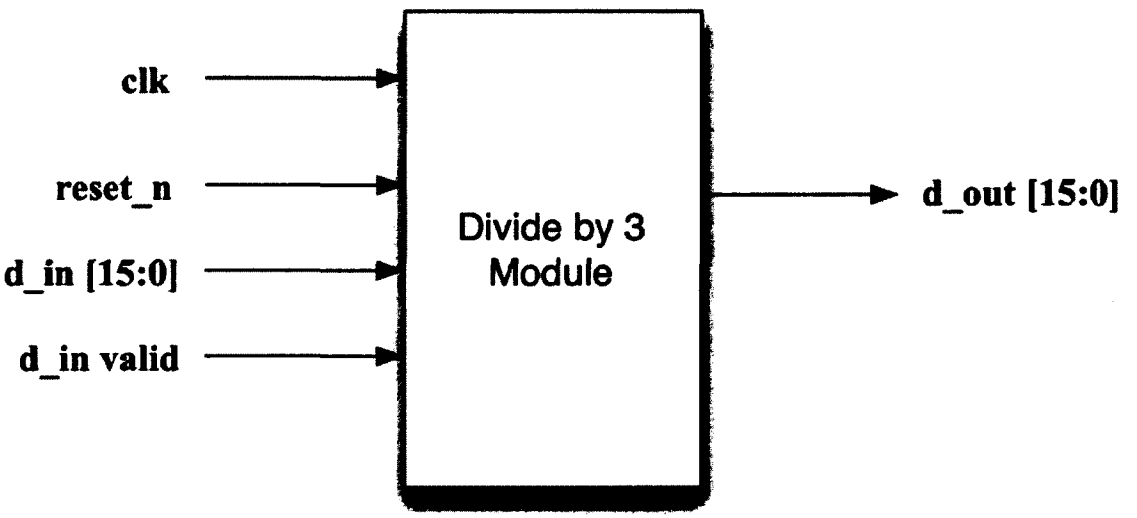

Figure 6.3: Divide by 3-Interface Module

The divide by 3-module is designed to divide the input din by 3. To find the average face it is required to divide the sum obtained from the parallel-pipelined adder by the total number of faces in the database. In our architecture we are using the divide by 3-module since we have used a total of 3 images in out database. The divide by 3-module was tested with different 15bit values for d_in, and the simulation was successful.

6.2.2.3 Read and Write Address Generator Module

The simulation in this section is for read and write address generator module, which was discussed in section 5.4.1.3. Figure 6.4 illustrates the interface diagram for read and write address generator module.

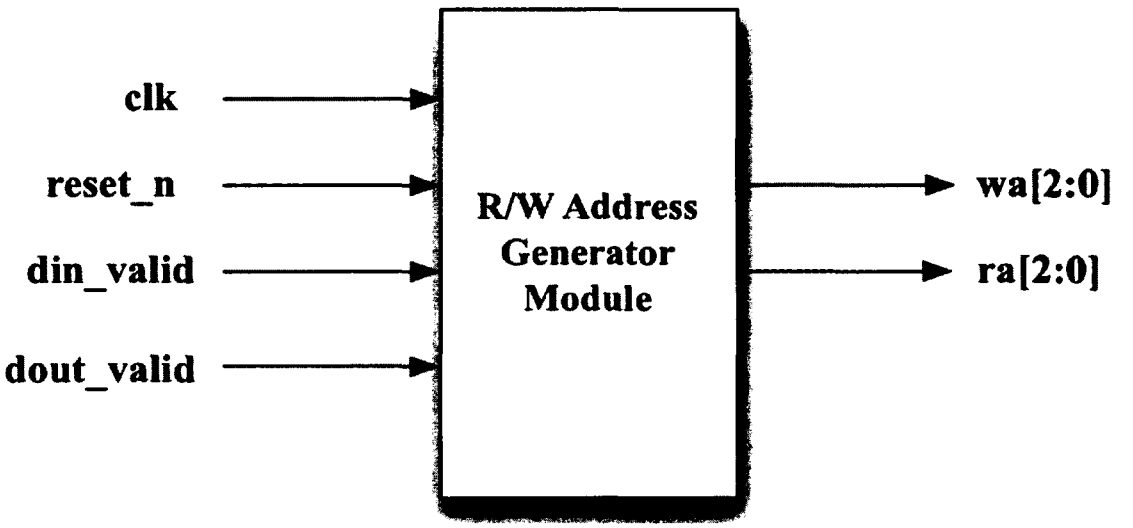

Figure 6.4: Read and Write Address Generator Module Interface

The read and write address generator module is a 3 bit controller that generates the read and write addresses. When the din_valid signal is high, the write counter is incremented and when the dout_valid signal is high, the read counter is incremented. This module was tested and the simulation was successful.

6.2.2.4 Read and Write Data to RAM Module

The simulation in this section is to read and write data to RAM module, which was discussed in section 5.4.1.2. Figure 6.5 illustrates the interface for data read and write to RAM module.

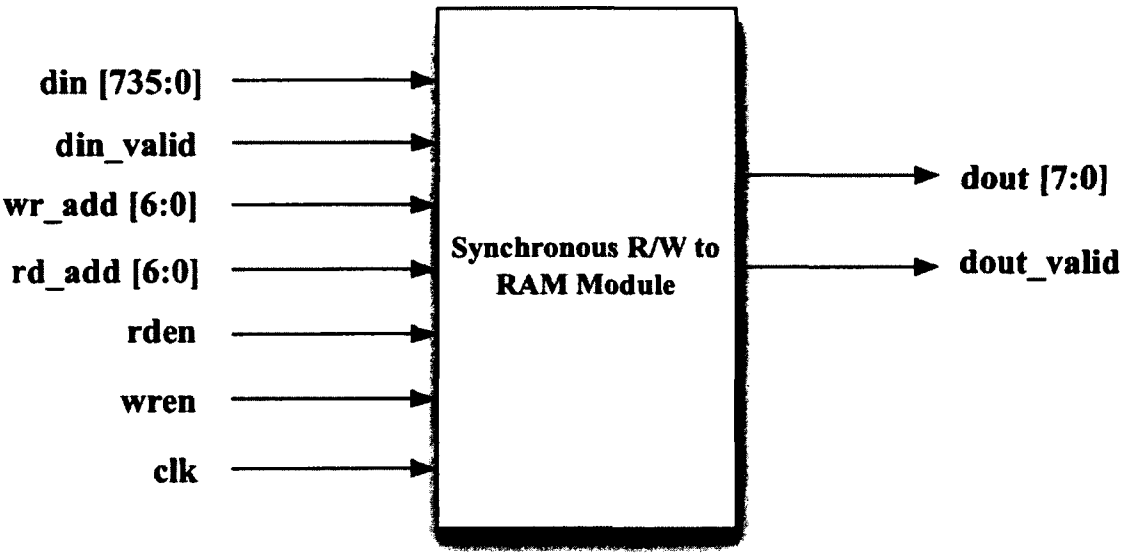

Figure 6.5: Read and Write Data to RAM Module Interface

The size of the image used in our architecture is 82kb. The number of bits for the input din, read address rd_add and write address wr_add were designed according to the size of the image used in our architecture. When wren (write enable) is high and din_valid is high, the data d_in is written into the RAM. When rden (read enable) is high, the data is read out of the memory (dout) and dout_valid is set high. This module was tested and the simulation was successful.

6.3 FPGA Implementation - Feasibility Issues

In the proposed architecture (Fig. 5.1), all modules can be implemented in FPGA except the MATLAB module. Due to lack of time we could not complete its implementation.

In eigenfaces method, we are required to find the eigenvalues and eigenvectors of the covariance matrix, now moving back to Eq. (5.32) the covariance matrix from the model calculation in chapter 5, we have.

$$L = \begin{bmatrix} 4548 & -2316 & -2275 \\ -2316 & 3285 & -993 \\ -2275 & -993 & 3340 \end{bmatrix}_{3\times3}$$

To find the eigenvectors of L matrix, we should introduce a new variable λ and an Identity matrix 'I', we then find the determinant for $[L - \lambda.I]$, this would give us a cubic equation in terms of λ. Here we are getting a cubic equation since our covariance matrix is of the dimension 3×3, this 3×3 dimension of the covariance matrix is because we have used 3 images in our database, for instance, if we have used 10 images in our database, then we would have a 10×10 matrix, that would give us an equation raised to the power of 10, solving this would be an intractable task.

Once we solve the cubic equation, we would get λ_1, λ_2 and λ_3, these are the eigenvalues of the covariance matrix, Then we have to substitute the eigenvalue in $[L - \lambda.I]$ and multiply it with vector v_1, $(L - \lambda_1.I)v_1$, Where v_1 is $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$, solving $(L - \lambda_1.I)v_1$, would give us three simultaneous equations in terms of x, y and z, solving them would give the eigenvector v_1 of λ_1, as shown in Eq. (5.35). Similarly solving $(L - \lambda_2.I)v_2$ *and* $(L - \lambda_3.I)v_3$ and their resulting simultaneous equations, we would arrive at eigenvectors v_2 *and* v_3 of λ_2 and λ_3 respectively as shown in Eq. (5.36) and Eq. (5.37)

$$\lambda_1 = 6866,\ v_1 = \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}$$

$$\lambda_2 = 4306,\ v_2 = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$$

$$\lambda_3 = 1,\ v_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

From this above model calculation (an excerpt from chapter 5), it can be seen that solving the covariance matrix for the eigenvalues and eigenvectors using Verilog HDL for FPGA would be a difficult task. This requires a novel architecture [Brav 08] [Brav 06]. Solving eigenvectors and eigenvalues from the covariance matrix is presently being extensively researched for a simple, unique and efficient solution for FPGA implementation.

Chapter 7

Conclusion and Future Work

7.1 Dissertation Summary

Current security systems based on password and ID card could be lost, forgotten or stolen. To avoid such situations a good solution is to implement an effective biometric system. Face recognition system is advantageous over other biometric systems. They are non-intrusive, cheaper and they do not require any explicit action from the user.

The state of art face recognition technologies [Cogn 10] [Ayon 10] [Auro 08] are implemented using powerful server computers and workstations with large memories. This type of hardware can only be placed in secure location with human supervision. This is a huge drawback even though they offer good recognition rate. This hurdle could be crossed if we could implement face recognition algorithm on FPGA. Currently face recognition algorithms are implemented using programming languages such as C++, Java, MATLAB, Python and Mathematica. They are yet to be written in a HDL.

This thesis explored the feasibility of FPGA implementation of face recognition using eigenfaces and an architecture has been proposed and was elucidated module by module, along with the simulation results for the main modules, using ModelSim Verilog simulator. The limitations of the architecture and the feasibility of FPGA implementation were also discussed.

7.2 Future Work

This thesis is an initial step towards exploring the feasibility of implementing face recognition algorithm on FPGA. This would enable a new generation of face recognition technology that is mobile, flexible and cost effective . These advantages will open up new avenues for the every increasing applications of face recognition technology.

References

[Agua 02] Mark Nixon and Alberto Aguado. *Feature Extraction and Image Processing.* First. Oxford: Academic Press is an imprint of Elsevier, 2002.

[Ahuj 02] Ming-Hsuan Yang, D. J. Kriegman and N. Ahuja. "Detecting faces in images: a survey." *IEEE - Pattern Analysis and Machine Intelligence* 24, no. 1 (January 2002): 34 - 58.

[Alte 11] Altera . "Mentor Graphics ModelSim and QuestaSim Support." Vers. 11.1. *Altera* . November 2011. http://www.altera.com/literature/hb/qts/qts_qii53001.pdf.

[Arab 09] Rabia Jafri and Hamid R. Arbina "A Survey of Face Recognition Techniques." *Journal of Information Processing Systems* 5, no. 2 (June 2009): 41- 68.

[Auro 08] Aurora . *Aurora - The Face of Biometrics* . http://www.auroracs.co.uk/best_solution.html.

[Ayon 10] Ayonix . *Face Recognition SDK.* http://ayonix.com/en/products/face-recognition-sdk.html.

[Bart 02] M. S. Bartlett, J. R. Movellan and T. J. Sejnowski. "Face recognition by independent component analysis." *IEEE, Neural Networks* 13, no. 6 (November 2002): 1450 - 1464.

[Belh 97] P. N. Belhumeur, J. P. Hespanha and D. J. Kriegman. "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection." *IEEE - Pattern Analysis and Machine Intelligence* 19, no. 7 (July 1997): 711 - 720.

[Bled 66] Bledsoe, W. W. "The Model Method in Facial Recognition, ." Palo Alto, CA: Panaromic Research Inc.,, August 1966.

[Brav 06] I. Bravo, P. Jimenez, M. Mazo, J. L. Lazaro and A. Gardel. "Implementation in FPGA's of Jacobi Method to Solve the Eigenvalue and Eigenvector Problem." *Field Programmable Logic and Applications*, August 2006: 1 - 4.

[Brav 08] I. Bravo, M. Mazo, J. L. Lazaro, P. Jimenez, A. Gardel and M. Marron. "Novel HW Architecture Based on FPGAs Oriented to Solve the Eigen Problem." *IEEE* -

Very Large Scale Integration (VLSI) Systems 16, no. 12 (December 2008): 1722 - 1725.

[Bron 04] Alexander M. Bronstein, Michael M. Bronstein, Ron Kimmel and Alon Spira. "3D Face Recognition Without Facial Surface Reconstruction." *8th European Conference on Computer Vision.* Prague: Springer, 2004.

[Camb 02] Cambridge University Computer Laboratory . *AT&T Laboratories Cambridge* . 2002. http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[Carm 09] Carmen Au, Jean Sebastien and Reehan Shaikh. "Face Recognition: Robustness of 'Eigenface' Approach." School of Computer Science & Center for Intelligent Machines, McGill University, Montreal.

[Cogn 10] Cognitec Systems. *FaceVACS-SDK.* http://www.cognitec-systems.de/FaceVACS-SDK.19.0.html.

[Como 92] Comon, Pierre. "Independent component analysis, A new concept?" *Elsevier - Signal Processing* 36, no. 3 (1992): 287–314.

[Cory 03] Warren E. Cory, Hare K. Verma, Atul V. Ghia, Paul T. Sasaki and Suresh M. Menon. Variable Data Width Operation in Multi-Gigabit Transceivers on a Programmable Logic Device. US Patent 6,617,877 B1. September 9, 2003.

[Cory 05] Cory, Warren E. Variable Data Width Converter. US Patent 6,970,013 B1. November 29, 2005.

[Coun 06] National Science and Technology Council. "Biometric History." Office of Science, Technology and Innovation, Goverment of USA, August 7, 2006. 1-27.

[Craw 06] Ian Craw, David Tock and Alan Bennett. "Finding Face Features." *ECCV* (Springer Berlin), January 2006: 92-96.

[Dela 11] Prof. Mislav Grgic and Kresimir Delac *Databases.* http://www.face-rec.org/databases/.

[Fisc 73] Fischer, M. A. and Elschlager, R. A. "The Representation and Matching of Pictorial Structures." *IEEE Transaction on Computers* C-22, no. 1 (January 1973): 67-92.

[Fish 09] Robert Fisher, Simon Perkins, Ashley Walker and Erick Wolfart. *Pixel Division* . 2004. http://homepages.inf.ed.ac.uk/rbf/HIPR2/pixdiv.htm.

[Gold 71] A. J. Goldstein, L. D. Harmon and A. B. Lesk. "Identification of Human Faces." *Proceedings of IEEE.* 1971. 748-760.

[Grgi 07] Kresimir Delac and Mislav Grgic. *Face Recognition.* InTech, 2007.

[Grou 97] Face Databases From Other Research Groups. http://www.ecse.rpi.edu/~cvrl/database/other_Face_Databases.htm.

[Guo 00] Guodong Guo, Kapluk Chan and S. Z. Li. "Face Recognition by Support Vector Machines." *Automatic Face and Gesture Recognition, Fourth IEEE Proceeding*. 2000. 196 - 201.

[Hami 90] Hamilton, A. G. *Linear Algebra : An Introduction with Concurrent Examples.* Vol. 2. Cambridge University Press, 1990.

[Heis 03] Bernd Heisele, Thomas Serre, Sam Prentice and Tomaso Poggio. "Hierarchical Classification and Feature Reduction for Fast Face Detection with Support Vector Machines." *ELSEVIER - Pattern Recognition* (ELSEVIER) 36, no. 9 (September 2003): 2007-2017.

[Jain 00] A. K. Jain, R. P. W Duin and Jianchang Mao. "Statistical Pattern Recognition: A Review." *IEEE - Pattern Analysis and Machine Intelligence* 22, no. 1 (January 2000): 4 - 37.

[Joll 02] Jolliffe, I. T. *Principal Component Analysis.* Second. Aberdeen: Springer Series in Statistics, 2002.

[Jord 02] F. R. Bach and M. I. Jordan "Kernel Independent Component Analysis." Edited by John Shawe -Taylor. *The Journal of Machine Learning Research* 3 (July 2002): 1-48.

[Kana 73] Kanade, Takeo. "Picture Processing System by Computer Complex and Recognition of Human Faces." Doctoral dissertation, Department of Information Science, Kyoto University, 1973.

[Kela 06] N. Kela, A. Rattani and P. Gupta. "Illumination Invariant Elastic Bunch Graph Matching for Efficient Face Recognition." *IEEE - Computer Vision and Pattern Recognition Workshop*, June 2006: 42-42.

[Kend 96] R. Kjeldsen and J. Kender "Finding Skin in Color Images." *IEEE - Automatic Face and Gesture Recognition*, October 1996: 312 - 317.

[Kirb 87] L. Sirovich and M. Kirby "Low Dimensional Procedure for the Characterization of Human Faces." *Journal of Optical Society of America* 4, no. 3 (1987): 519-524.

[Kirb 90] M. Kirby and L. Sirovich. "Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces." *IEEE - Pattern Analysis and Machine Intelligence* 12, no. 1 (January 1990): 103 - 108.

[Kitt 98] J. Kittler, M. Hatef, R.P.W Duin and J. Matas. "On Combining Classifiers." *IEEE - Pattern Analysis and Machine Intelligence* 20, no. 3 (March 1998): 226 - 239.

[Lani 95] A. Lanitis, C. J. Taylor and T. F. Cootes. "Automated Face Identification System Using Flexible Apperance Models." *Image and Vision Computing - Machine Vision Conference* (ELSEVIER) 13, no. 5 (June 1995): 393–401.

[Leun 95] T. K. Leung, M. C. Burl and P. Perona. "Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching." *IEEE - Computer Vision*, June 1995: 637 - 644.

[Lew 96] Lew, M. S. "Information Theoretic View-based and Modular Face Detection." *IEEE - Automatic Face and Gesture Recognition*, October 1996: 198 - 203.

[Liu 99] H. Wechsler and C. Liu "Comparitive Assessment of Independent Component Analysis (ICA) for Face Recognition." *Audio and Video-Based Biometric Person Authentication*, March 1999: 22-24.

[Lu 03] Lu, X. "Image Analysis for Face Recognition." Personal Notes, Computer Science and Engineering, Michigan State University, Michigan, 2003.

[Mals 92] B. S. Manjunath and R. Chellappa and C. Von der. Malsburg "A Feature Based Approach to Face Recognition." *IEEE - Computer Vision and Pattern Recognition*, June 1992: 373 - 378.

[Marq 10] Marques, Ion. "Face Recognition Algorithms." Thesis, Universidad del Pais Vasco, 2010.

[Mitc 96] Tom M. Mitchell *Machine Learning.* Edited by C. L. Liu and A. B. Tucker. McGraw Hill, 1996.

[Mode 05] ALTERA. "About Using the ModelSim Software with the Quartus II Software." *ALTERA.* http://quartushelp.altera.com/current/master.htm#mergedProjects/eda/simulation/modelsim/eda_view_using_msim.htm.

[Naka 96] Ying Dai and Yasuaki Nakano. "Face-Texture Model based on SGLD and its Application in Face Detection in a Color Scene." *ELSEVIER - Pattern Recognition* 29, no. 6 (June 1996): 1007–1017.

[Nefi 98] M. H. Hayes and A. V. Nefian "Hidden Markov models for face recognition." *IEEE - Acoustics, Speech and Signal Processing* 5 (May 1998): 2721 - 2724.

[Nixo 85] Nixon, M. "Eye Spacing Measurment for Facial Recognition." *SPIE - Application of Digital Image Processing* 575 (August 1985): 279-285.

[Osun 97] E. Osuna, R. Freund and F. Girosit. "Training Support Vector Machines: An Application to Face Detection." *IEEE - Computer Vision and Pattern Recognition*, June 1997: 130 - 136.

[Pent 91] Matthew Turk and Alex Pentland "Eigenfaces for Recognition." *Journal of Cognitive Neuroscience* 3, no. 1 (1991): 71-86.

[Pent 97] B. Moghaddam and A. Pentland "Probabilistic Visual Learning For Object Representation." *IEEE - Pattern Analysis and Machine Intelligence* 19, no. 7 (July 1997): 696 - 710.

[Phil 97] P.J. Phillips, Hyeonjoon Moon, P. Rauss, S. A. Rizvi. "The FERET Evaluation Methodology for Face Recognition Algorithms." *IEEE Computer Vision and Pattern Recognition* , June 1997: 137 - 143 .

[Phil 99] Phillips, P. J. "Support Vector Machines Applied to Face Recognition." In *Advances in Neural Information Processing*, by P. J. Phillips, 803-809. Gaithersburg: MIT Press, 1999.

[Phil 03] P. J. Phillips, P.Grother, R.Micheals, D. M. Blackburn, E. Tabassi, M. Bone. "Face Recognition Vendor Test 2002." *IEEE - Analysis and Modeling of Faces and Gestures* (DARPA, Arlington, VA, USA), October 2003: 44.

[Piss 02] Pissarenko, Dimitri. "Eigenface based Facial Recognition." December 2002.

[Piur 96] Piuri, L. Dadda and V. "Pipelined adders." *IEEE - Computer Society* 45, no. 3 (March 1996): 348 - 356.

[Pogg 92] R. Brunelli and T. Poggio "Face Recognition Through Geometrical Features." *Computer Vision — ECCV* (Springer Berlin) 588 (may 1992): 792-800.

[Pogg 98] K. K. Sung and T. Poggio "Example-based Learning for View-Based Human Face Detection." *IEEE - Pattern Analysis and Machine Intelligence* 20, no. 1 (January 1998): 39 - 51.

[Raja 98] A. N. Rajagopalan, K. S. Kumar, J. Karlekar, R. Manivasakan, M. M. Patil, U. B. Desai, P. G. Poonacha and S. Chaudhuri. "Finding Faces in Photographs." *IEEE - Computer Vision*, January 1998: 640 - 645.

[Roli 01] F. Roli and J. Kittler "Multiplier Classifier Systems." Cambridge: Springer, 2001.

[Rorr 04] Howard Anton and Chris Rorres *Elementary Linear Algebra.* Nine. John Wiley & Sons Inc, 2004.

[Rose 03] W. Zhao, R. Chellappa, P. J. Phillips and A. Rosenfeld. "Face Recognition : A Literature Survey." *ACM Computing Survey* 35, no. 4 (December 2003): 399-458.

[Rowl 98] Henry A. Rowley, Shumeet Baluja and Takeo Kanade. "Neural Network-Based Face Recognition." Edited by DC, USA IEEE Computer Society Washington. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, no. 1 (January 1998): 23 - 38.

[Sama 93] F. Fallside and F. Samaria "Face Identification and Feature Extraction Using Hidden Markov Model." In *Image Processing : Theory and Applications*, by F. Fallside and F. Samaria, 295-298. Cambridge: Elsevier, 1993.

[Savv 11] Dr. Marios Savvides *Introduction to Biometric Recognition Technologies and Applications.* Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh: Carnegie Mellon Cylab and ECE, 1-54.

[Schö 98] B. Schölkopf, A. Smola and K. R. Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem." *MIT Press Journals* (Massachusetts Institute of Technology) 10, no. 5 (July 1998): 1299-1319.

[Sebe 02] N. Sebe, M. S. Lew, I. Cohen, A. Garg and T. S. Huang. "Emotion Recognition Using a Cauchy Naive Bayes Classifier." *IEEE - Pattern Recognition* 1 (2002): 17 - 20.

[Shle 05] Jonathon Shlens "A Tutorial on Principal Component Analysis." 51, no. 10003 (December 2005): 52.

[Smit 02] Lindsey I. Smith "A Tutorial on Principal Component Analysis ." Otago, 2002.

[Soft 11] Altera. *ModelSim-Altera Software.* http://www.altera.com/products/software/quartus-ii/modelsim/qts-modelsim-index.html.

[Tagi 54] J. S. Bruner and R. Tagiuri *The Perception of People.* Vol. 2, in *In Handbook of Social Psychology*, by J. S. AND TAGIURI, R. BRUNER, edited by G.Lindzey, 634-654. Reading, MA: Addison-Wesley, 1954.

[Tayl 01] T. F. Cootes, G. J. Edwards, C. J. Taylor. "Active Appearance Models." *IEEE - Pattern Analysis and Machine Intelligence* 23, no. 6 (June 2001): 681 - 685.

[Teka 98] Eli Saber and A. Murat Tekalp"Frontal-view Face Detection and Facial Feature Extraction Using Color, Shape and Symmetry based Cost Functions." *ELSEVIER - Pattern Recognition Letters* (ELSEVIER) 19, no. 8 (June 1998): 669-680.

[Tolb 06] A. S. Tolba, A. H. El-Baz and A. A. El-Harby. "Face Recognition: A Literature Review." *World Academy of Science, Engineering and Technology* 19 (2006): 319-334.

[Triv 09] Trivedi, Shubhendu. *Face Recognition using Eigenfaces and Distance Classifiers: A Tutorial.* Feburary 11, 2009. http://onionesquereality.wordpress.com/2009/02/11/face-recognition-using-eigenfaces-and-distance-classifiers-a-tutorial/.

[Tuly 08] Sergey Tulyakov, Stefan Jaeger, Venu Govindaraju and David Doermann. "Review of Classifier Combination Methods." *Machine Learning in Document Analysis and Recognition* (Springer Berlin) 90 (2008): 361-386.

[Turk 91] Matthew Alan Turk, Alexander Pentland. "Eigenfaces for Recognition." *IEEE - Computer Vision and Pattern Recognition*, June 1991: 586 - 591.

[Turk 01] Turk, Matthew. "A Random Walk Through the Eigenspace." *IEICE Transaction on Information and Systems* 84, no. 12 (December 2001): 1586-1595.

[Vett 03] V. Blanz and T. Vetter "Face Recognition Based on Fitting a 3D Morphable Model." *IEEE - Pattern Analysis and Machine Intelligence* 25, no. 9 (September 2003): 1063 - 1074.

[Walk 00] T. F. Cootes, K. Walker, C. J. Taylor. "View-based Active Appearance Models." *IEEE - Automatic Face and Gesture Recognition*, March 2000: 227 - 232.

[Wang 03] X. Lu, Y. Wang and A. K. Jain. "Combining Classifiers for Face Recognition." *ICME Proceedings - Multimedia and Expo* 3 (July 2003): 13-16.

[Wisk 97] Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, Christopher von der Malsburg. "Face Recognition by Elastic Bunch Graph Matching." *IEEE Transactions on Pattern Analysis and Machine Intelligence* (IEEE Computer Society Washington, DC, USA) 19, no. 7 (July 1997): 775 - 779.

[Work 10] MathWorks . *Getting Information about the Pixels in an Image.* http://www.mathworks.com/help/toolbox/images/f10-40600.html.

[Yang 02] Yang, M H. "Face recognition using kernel methods." *Advances in neural information processing systems* 2 (2002): 1457-1464.

[Yow 97] Cipolla .R and Kin Choong Yow "Feature-based Human Face Detection." *Image and Vision Computing* (ELSEVIER) 15, no. 9 (September 1997): 713–735.

[Zaba 09] Zabarauska, Manfredas. *Eigenfaces Tutorial.* October 2, 2009. http://blog.zabarauskas.com/eigenfaces-tutorial/.

[Zhan 09] M. Mayo and E. Zhang "3D Face Recognition Using Multiview Keypoint Matching." *IEEE - Advanced Video and Signal Based Surveillance*, September 2009: 290 - 295.

[Zhou 04] S. Zhou, R. Chellappa and B. Moghaddam. "Intra-Personal Kernel Space for Face Recognition." *IEEE- Automatic Face and Gesture Recognition*, May 2004: 235 - 240 .

Vita Auctoris

Vinod Anbalagan was born in Chennai, Tamil Nadu, India, in 1986. He received his B.E. degree in electronics and computer engineering in 2004 form Anna University. He is currently a candidate in the electrical and computer engineering M.A.Sc. program at the Universtiy of Windsor. His research interest include Image Processing, FPGAs and Digital Systems.