FPGA IMPLEMENTATION OF HAND VEINS RECOGNITION SYSTEM

By:

AMMAR HASSAN

AHMAD RAZA KHAN

UMAR FAROOQ

PROJECT DS: MAJ DR ADIL MASOOD

Thesis Submitted to the Faculty of Department of Electrical Engineering, Military College of Signals National University of Sciences and Technology, Rawalpindi in partial fulfillment for the requirements of a B.E Degree in Telecommunication Engineering

JUNE 2012

ABSTRACT

FPGA IMPLEMENTATION OF HAND VEINS RECOGNITION SYSTEM

The demand for a reliable and robust personal identification system arises as technology advances and develops. A biometric system is the key to provide a more secure and reliable personal identification system. A biometric system verifies user identity by comparing the behavioral or physiological trait possessed by the user to a previously stored sample of the trait. This biometric system prevents identity theft by providing verification accurately. Among them, hand vein biometric system is gaining popularity and has emerged as new technology in biometric system. This project proposed an design of biometric hand vein authentication system that involves interface design and software development. This hand vein system can be divided into two main parts; The Graphical User Interface (GUI) on a host PC and the biometric hand vein hardware system . The hand vein image that was captured earlier by specially designed camera that was connected to the host computer is accessed by the GUI and displayed in GUI. The Biometric Hand Vein System contains four stages; image acquisition, image processing, feature extraction and matching stage. After image acquisition through camera, image processing stage will enhance the hand vein image, the feature extraction stage will extract the valid minutiae points in template based, and the matching stage will match the minutiae template with a previously stored template. The results of this work shows the proposed system with no doubt is functioning correctly.

DISSERTATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATED TO THE MARTYRS OF PAKISTAN ARMY

ACKNOWLEGMENTS

We are grateful to Dr Adil Masood for providing us an opportunity to perform our BE final year project in his Image processing group. As supervisor, his support and guidance has always been a valuable asset for our project. His keen interest and discussions over the work always provided a ray of hope in difficult times. We are really thankful for his guidance in our project work.

In the end we would like to thank Rector National University of Sciences & Technology for funding this project. We are also grateful to the whole staff of the Department of Electrical Engineering especially to Head of the department for their support in many forms and enthusiasm

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVATIONS

GUI	-	Graphical User Interface
CLI	-	Command Line Interface
HD	-	Hamming Distance
IR	-	Infrared
LED	-	Light Emitting Diode
USB	-	Universal Serial Bus
PC	-	Personal Computer

CHAPTER 1

INTRODUCTION

This thesis proposes the hand vein biometric system. The design incorporates a complete system that can receive image from specially designed infrared webcam and perform image processing, extraction of the biometric feature from the image and authenticate the template stored in database.

1.1 Background

Nowadays, as technology is advancing very fast, public are getting more and more conscious and aware towards information safety. Often we heard in news about bank accounts are hacked into, credits cards are stolen and etc. In order to preserve information safety, security access system has to be employed. Traditional security access systems based on passwords, personal identification numbers (PINS), swipe-cards, keys, and so on offers limited security and usually unreliable. These methods can be compromised and breached very easily where these methods can be hacked, duplicated or forged. This breaching of security happens when password is divulged to an unauthorized user or a badge is stolen by an impostor. The key to overcome the disadvantages of traditional methods is to use biometric system.

Biometrics are automated methods of recognizing a person based on a physiologicalor behavioral characteristic. Among the features measured are; face, fingerprints, handgeometry, handwriting, iris, retinal, vein, and voice. These features possess the properties of universality, uniqueness, permanence, collectability, acceptability and circumvention. Thus, a biometric technique has become a new face for authentication in security access system. Biometric system provides more reliable feature because these features cannot be forgotten. Biometric systems are superiorbecause they provide a nontransferable means of identifying peoplenot just cards orbadges. Apart from that, these features are difficult to duplicate and most importantly it requires the person to present there for authentication.

The key point about an identification method that is ”nontransferable" means itcannot be given or lent to another individual so nobody can get around the system -they personally have to go through the control point.

In a practical biometric system (i.e., a system that employs biometrics for personalrecognition), there are a number of other issues that should be considered, including:Performance, which refers to the achievable recognition accuracy and speed, theresources required to achieve the desired recognition accuracy and speed, as well asthe operational and environmental factors that affect the accuracy and speed;Acceptability, which indicates the extent to which people are willing to accept theuse of a particular biometric identifier (characteristic) in their daily lives;Circumvention, which reflects how easily the system can be fooled using fraudulentmethods.

A key advantage of biometric authentication is that biometric data is based onphysical characteristics that stay constant throughout one’s lifetime and are difficult(some more than others) to fake or change. Biometric identification can provideextremely accurate, secured access to information; fingerprints, palm vein and iris scansproduce absolutely unique data sets (when done properly). Automated biometricidentification can be done rapidly and uniformly, without resorting to documents thatmay be stolen, lost or altered. It is not easy to determine which method of biometricdata gathering and reading does the "best" job of ensuring secure authentication. Each of the different methods has inherent advantages and disadvantages. Some are less invasive than others; some can be done without the knowledge of the subject; others are very difficult to fake.

Palm vein authentication has a high level of authentication accuracy due to the uniqueness and complexity of vein patterns of the palm. Because the palm vein patternsare internal to the body, this is a difficult method to forge. Also, the system is contactless and hygienic for use in public areas. It is more powerful than otherbiometric authentication such as face, iris, and retinal.

Palm vein authentication uses an infrared beam to penetrate the users hand as it is held over the sensor; the veins within the palm of the user are returned as black lines. Palm vein authentication has a high level of authentication accuracy due to the uniqueness and complexity of vein patterns of the palm. Because the palm vein patterns are internal to the body, this is a difficult method to forge. Also, the system is contactless and hygienic for use in public areas.

Embedded systems have become increasingly popular as advances in IC-technology and processor architecture allow for flexible computational parts and high-performance modules integrated on a single carrier. Embedded system interacts with the physical world. It executes on machines that are not computers. They are cars, airplanes, telephones, audio equipment, robots, appliances, toys, security systems and so on. These embedded systems perform functions that are carefully partitioned in software and hardware to strike a fine balance between flexibility, reusability, performance and cost.

Embedded system provides a medium of secure communication, secure information storage, and tamper resistance which protect from both physical and software attacks.

1.2 Problem Statement

It is important to have reliable personal identification due to growing importance of information technology. Nowadays most of security applications are switching to embedded system design due to the fact that it is easier to interact with the physical world, low power consumption and can be treated as a stand-alone application. The hand vein biometric also proves to be ease in embedded implementation and make way in terms of prototyping for further research and integration with other components.

Some problem with existing biometric systems was identified. One of the problems with existing biometric hand vein system is that it is more prone to data theft or information theft. Specially designed camera was used to capture the hand vein image and this camera is connected to a computer. Thus, the captured image is processed further

Apart from that, registration is also unavailable for this hand vein system. Registration here means adding new user in the database, deleting unwanted user from database and authenticating a user. With existing system, adding a new user into database is a complicated work where images are renamed first and saved in respective folder to be processed. Removing a user from database is even more complicated from adding user into database. This is because the images of corresponding user have to be deleted from the entire folders. The complexity of authenticating a user is the same as adding a user. The captured image is saved in respective folder before the image can be successfully authenticated.

1.3 Objective of the Project

The objective of this project is to design and implement a complete hand biometric identification and authentication system which takes an image of hand by a modified webcam illuminated by an infrared source and then it is further processed through different techniques in image pre-processing, extracting the important features from it and then in turn matching it for authentication, depicted in figure 1.1.

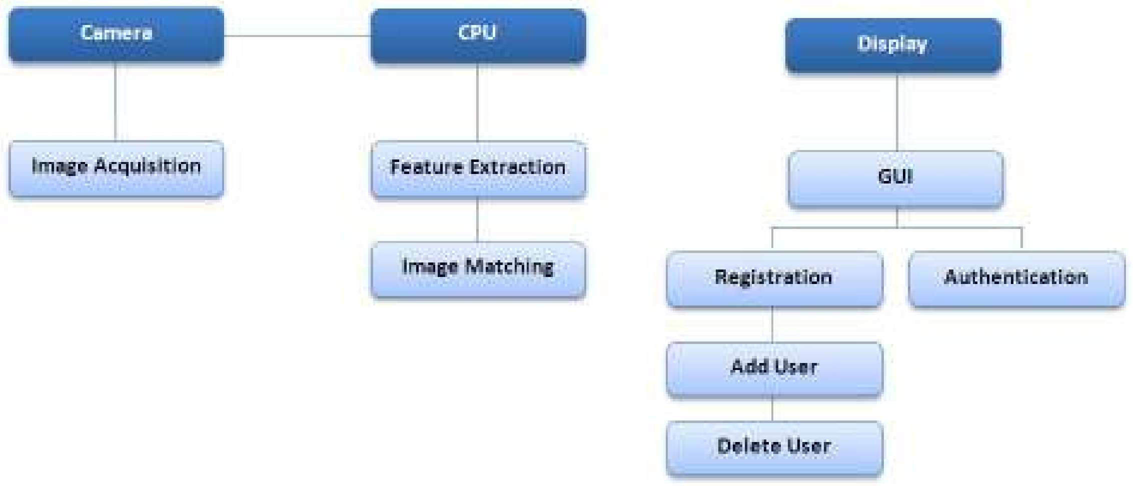

Figure 1.1 Overview of system

The second objective of this project is to design and create a graphical user interface to be used as front end for hand vein system. The Graphical User Interface (GUI) will provide means of communication between computer and the user. This GUI will be linked image acquisition, image pre-processing, feature extraction and image matching. So, by creating a simple and attractive GUI this system can be used by everyone as GUI provides a user friendly platform. This GUI will be designed to provide a simple way to register a user into the database, delete a user from database or authenticate a user from database.

1.5 Project Contribution

A prototype of the enhanced hand vein biometric system is implemented in a compact and efficient design. The biometric system is able to receive image from the infrared camera, enhance the image through image processing stages, extract the minutiae points in template based and match the template stored in database with the image that need to be authenticated. Besides, to make this hand vein system more user friendly graphical user interface will be designed.

1.6 Thesis Organization

This thesis is organized into four chapters. The first chapter introduces the background, motivation, research objectives, and scope of work and contribution of this project. The second chapter describes the theory of hand vein image processing steps, feature extraction.The third chapter presents the research methodology, system design procedures and application tools and last chapter discusses the results obtained followed by the discussions.

CHAPTER 2

LITERATURE REVIEW

This chapter embraces all the paper works and related research as well as the studies with regards to this project. The chapter includes all the important studies which have been done previously by other research work. The related works have been referred carefully since some of the knowledge and suggestions from the previous work can be implemented for this project.

Literature review was an ongoing process throughout the whole process of the project. It is very essential to refer to the variety of sources in order to gain more knowledge and skills to complete this project. These sources include reference books, thesis, journals and also the materials obtained from internet.

2.1 Biometric

Biometrics technology is defined as an automated method of identifying or authenticating the identity of a living person based on a physiological or behavioral characteristic.

The phrase "automated methods" refers to three basic methods connected with biometric devices: A mechanism to scan and capture a digital or analog image of a living personal characteristic, Compression, processing and comparison of the image to a database of stored images, Interface with applications systems (Industry Information: Biometrics, 1996). These methods can be configured in a number of different topographies depending upon the biometric device and application. For example, a common issue is whether the stored images (reference templates) reside on a card, in the device or at a host or database. (Industry Information: Biometrics, 1997)

The term "living person" may seem obvious, but it is an important component in defining biometrics. One of the first questions newcomers to the field ask is, "What about a latex finger, digital audio tape, plasters hand, or prosthetic eye." The answer is that biometric devices can incorporate specific algorithms that determine whether there is a live characteristic being presented. The term "living" also separates the biometric industry from the forensic identification field, although basic principles transcend both areas (Industry Information: Biometrics, 1997).

Biometric security ensures user access to an embedded system through trait verification. These traits have to be permanent, easy-to-obtain (minimally intrusive), universal (in the sense that generally everyone possesses it) yet unique (in order to differentiate a certain user's trait from the others').

2.1.1 Biometric Systems

A biometric system is essentially a pattern-recognition system that recognizes a person basedon a feature vector derived from a specific physiological or behavioral characteristic that theperson possesses as shown in figure 2.1.That feature vector is usually stored in a database (or recorded on asmart card given to the individual) after being extracted.

A biometric system based on physiological characteristics is generally more reliable than one which adopts behavioralcharacteristics, even if the latter may be easier to integrate within certain specificapplications. Biometric system can than operate in two modes:

verification or *identification*. While identification involves comparing the acquired biometric information against templates corresponding to all users in the database, verification involves comparison with only those templates corresponding to the claimed identity. This implies that identification and verification are two problems that should be dealt with separately.

A simple biometric system consists of four basic components. First is the sensor module witch acquires the biometric data. Second is the feature extraction module where the acquired data is processed to extract feature vectors. Third is the matching module where feature vectors are compared against those in the template and the last one is decision-making module in which the user's identity is established or a claimed identity is accepted or rejected.

Figure 1.1 Biometric systems

2.1.2 Overview of Commonly Used Biometrics

Since there are number of biometric methods in use (some commercial, some "not yet"), abrief overview of various biometric characteristics will be given, starting with newer technologies and then progressing to older ones: Infrared thermogram (facial, hand or hand vein).It is possible to capture the pattern ofheat radiated by the human body with an infrared camera. That pattern is considered to beunique for each person. It is a noninvasive method, but image acquisition is rather difficultwhere there are other heat emanating surfaces near the body. The technology could be usedfor covert recognition. A related technology using near infrared imaging is used to scan the back of a fist to determine hand vein structure, also believed to be unique. Like face recognition, it must deal with the extra issues of three-dimensional space and orientation of the hand. Set-back is the price of infrared sensors.

2.1.2.1 Gait

This is one of the newer technologies and is yet to be researched in more detail.Basically, gait is the peculiar way one walks and it is a complex spatio-temporal biometrics, figure 2.2.

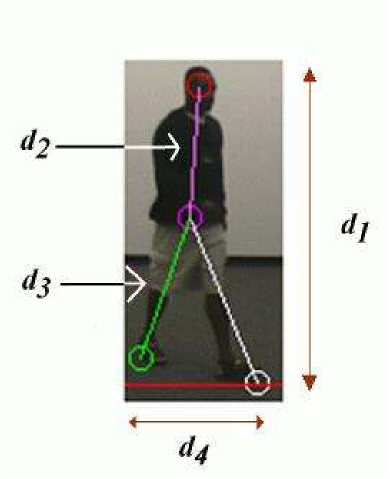

Figure 2.2 Gait recognition

It is not supposed to be very distinctive but can be used in some low-security applications. Gait is a behavioral biometric and may not remain the same over a long period of time, due to change in body weight or serious brain damage. Acquisition of gait is similar to acquiring a facial picture and may be an acceptable biometric. Since video-sequence is used to measure several different movements this method is computationally expensive.

2.1.2.2 Keystroke

It is believed that each person types on a keyboard in a characteristic way, figure 2.3. This is also not very distinctive but it offers sufficient discriminatory information to permit identity verification. Keystroke dynamics is a behavioral biometric; for some individuals, one could expect to observe large variations in typical typing patterns. Advantage of this method is that keystrokes of a person using a system could be monitored unobtrusively as that person is keying information. Another issue to think about here is privacy.

Figure 2.3 Keystroke Recognition

2.1.2.3 Odor

Each object spreads around an odor that is characteristic of its chemical composition and this could be used for distinguishing various objects. This would be done with an array of chemical sensors, each sensitive to a certain group of compounds. Deodorants and perfumes could lower the distinctiveness.

2.1.2.4 Ear

It has been suggested that the shape of the ear and the structure of the cartilaginous tissue of the pinna are distinctive. Matching the distance of salient points on the pinna from a landmark location of the ear is the suggested method of recognition in this case. This method is not believed to be very distinctive.

2.1.2.5 Hand geometry

The essence of hand geometry is the comparative dimensions of fingers and the location of joints, shape and size of palm. One of the earliest automated biometric systems was installed during late 60s and it used hand geometry and stayed in production for almost 20 years. The technique is very simple, relatively easy to use and inexpensive. Dry weather or individual anomalies such as dry skin do not appear to have any negative effects on the verification accuracy. Since hand geometry is not very distinctive it cannot be used for identification of an individual from a large population, but rather in a verification mode.Further, hand geometry information may not be invariant during the growth period of children. Limitations in dexterity (arthritis) or even jewelry may influence extracting the correct hand geometry information. This method can find its commercial use in laptops rather easy. There are even verification systems available that are based on measurements of only a few fingers instead of the entire hand. These devices are smaller than those used for hand geometry.

2.1.2.6 Fingerprint

A fingerprint is a pattern of ridges and furrows located on the tip of each finger.Fingerprints were used for personal identification for many centuries and the matching accuracy was very high. Patterns have been extracted by creating an inked impression of the fingertip on paper. Today, compact sensors provide digital images of these patterns. Fingerprint recognition for identification acquires the initial image through live scan of the finger by direct contact with a reader device that can also check for validating attributes such as temperature and pulse. Since the finger actually touches the scanning device, the surface can become oily and cloudy after repeated use and reduce the sensitivity and reliability ofoptical scanners. Solid state sensors overcome this and other technical difficulties because thecoated silicon chip itself is the sensor. Solid state devices use electrical capacitance to sense the ridges of the fingerprint and create a compact digital image. Today, a fingerprint scanner costs about 20 USD and has become affordable in a large number of applications (laptopcomputer). In real-time verification systems, images acquired by sensors are used by the feature extraction module to compute the feature values. The feature values typically correspond to the position and orientation of certain critical points known as minutiae points. The matching process involves comparing the two-dimensional minutiae patterns extracted from the user's print with those in the template. One problem with the current fingerprint recognition systems is that they require a large amount of computational resources.

2.1.2.7 Face

Facial images are the most common biometric characteristic used by humans to make a personal recognition, hence the idea to use this biometric in technology. This is a nonintrusive method and is suitable for covert recognition applications. The applications of facial recognition range from static ("mug shots") to dynamic, uncontrolled face identification in a cluttered background (subway, airport). Face verification involves extracting a feature set from a two-dimensional image of the user's face and matching it with the template stored in a database Figure 2.4. The most popular approaches to face recognition are based on either: 1) the location and shape of facial attributes such as eyes, eyebrows, nose, lips and chin, and their spatial relationships, or 2) the overall (global) analysis of the face image that represents a face as aweighted combination of a number of canonical faces. Although performance ofcommercially available systems is reasonable there is still significant room for improvement since false reject rate (FRR) is about 10% and false accept rate (FAR) is 1%. These systems also have difficulties in recognizing a face from images captured from two differentangles and under different ambient illumination conditions.

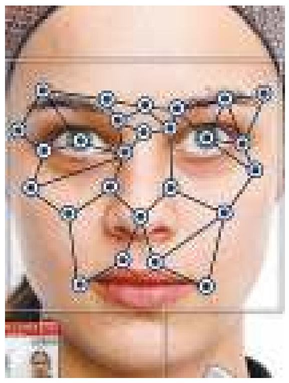

Figure 2.4 Face recognition

It is questionable if a face itself is a sufficient basis for recognizing a person from a large number of identities with an extremely high level of confidence. Facial recognition system should be able to automatically detect a face in an image, extract its features and then recognize it from a general viewpoint (i.e., from any pose) which is a rather difficult task. Another problem is the fact that the face is a changeable social organ displaying a variety of expressions.

2.1.2.8 Retina

Retinal recognition creates an "eye signature" from the vascular configuration of the retina which is supposed to be a characteristic of each individual and each eye, respectively. Since it is protected in an eye itself, and since it is not easy to change or replicate the retinal vasculature, this is one of the most secure biometric. Image acquisition requires a person to look through a lens at an alignment target. Therefore it implies cooperation of the subject. Also retinal scan can reveal some medical conditions and as such public acceptance is questionable.

2.1.2.9 Iris

The iris begins to form in the third month of gestation and the structures creating its pattern are largely complete by the eight month. Its complex pattern can contain many distinctive features such as arching ligaments, furrows, ridges, crypts, rings, corona, frecklesand a zigzag collarets. Iris scanning is less intrusive than retinal because the iris is easilyvisible from several meters away. Responses of the iris to changes in light can provide an important secondary verification that the iris presented belongs to a live subject. Irises of identical twins are different, which is another advantage. Newer systems have become more user-friendly and cost-effective. A careful balance of light, focus, resolution and contrast is necessary to extract a feature vector from localized image. While the iris seems to be consistent throughout adulthood, it varies somewhat up to adolescence.

2.1.2.10 Palmprint

Like fingerprints, palms of the human hands contain unique pattern of ridges and valleys. Since palm is larger than a finger, palmprint is expected to be even more reliable than fingerprint. Palmprint scanners need to capture larger area with similar quality as fingerprint scanners, so they are more expensive. A highly accurate biometric system could be combined by using a high-resolution palmprint scanner that would collect all the features of the palm such as hand geometry, ridge and valley features, principal lines, and wrinkles.

2.1.2.11 Voice

The features of an individual's voice are based on physical characteristics such as vocal tracts, mouth, nasal cavities and lips that are used in creating a sound. These characteristics of human speech are invariant for an individual, but the behavioral part changes over time due to age, medical conditions and emotional state.

2.1.2.12 Signature

Signature is a simple, concrete expression of the unique variations in human hand geometry. The way a person signs his or her name is known to be characteristic of that individual. Collecting samples for this biometric includes subject cooperation and requires the writing instrument. Signatures are a behavioral biometric that change over a period of time and are influenced by physical and emotional conditions of a subject. In addition to the general shape of the signed name, a signature recognition system can also measure pressure and velocity of the point of the stylus across the sensor pad.

2.1.2.13 DNA

Deoxyribonucleic acid (DNA) is probably the most reliable biometrics. It is in fact aone-dimensional code unique for each person. Exceptions are identical twins. This method, however, has some drawbacks. It is easy to steal a piece of DNA from an individual and use it for an ulterior purpose. Secondly no real-time application is possible because DNA matching requires complex chemical methods involving expert's skills. All this limits the use of DNA matching to forensic applications. It is obvious that no single biometric is the "ultimate" recognition tool and the choice depends on the application.

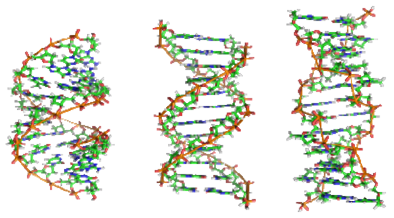

Figure 2.5 DNA recognition

2.1.3 Physiological and Behavioral Characteristics

Biometric system is a pattern recognition system that confirms the authenticity of a user's specific physiological or behavioral characteristic. A physiological characteristic is relatively stable, as it is basically unchanging throughout one's life (unless injured/altered with significant force). Physiological characteristics include fingerprint, hand silhouette and iris pattern.

Behavioral characteristics are a reflection of an individual's psychological makeup, even though general physical traits, like size and sex, play a huge role (Industry Information: Biometrics, 1997). Examples of behavioral traits used to identify individuals include a person's typing patterns at a keyboard, commonly referred to as keystroke dynamics, and the unique characteristics of how one speaks or speech identification and/or verification.

2.1.4 Identification and Authentication:

Two main aspects of biometrics are:

2.1.4.1 Identification

Identification occurs when an individual's characteristic is being selected from a group of stored images. Identification is the way the human brain performs most day-to-day identifications (Industry Information: Biometrics, 1997). For example, if a person encounters a familiar individual, the brain processes the information by comparing what the person is seeing to what is stored in memory. Biometric devices that implement identification techniques can be quite time consuming. Often anywhere from five to 15 seconds or more are required in identifying the appropriate individual.

2.1.4.2 Authentication

In many cases, verification is used to authenticate a user's identity. A biometric device that uses verification requires that the individual make a claim of identity by presenting a code or a card. The matching formula or algorithm then needs only to compare the live and enrolled images of the user's characteristic. The question put to the machine is, "Are you who you say you are?" instead of, "Do I know who you are?" (Industry Information: Biometrics, 1997). Verification can be viewed as adding another level of security. A good analogy is when a person goes to add a dead-bolt to a door. In this case, a dead-bolt is usually added to increase the security of the door or entrance because generally a lock of some sorts that was on the door beforehand.

There are other notable details to consider in addition to the terms used to define biometrics. Other biometric performance factors that need to be thoroughly investigated include accuracy, speed, reliability, acceptability, resistance to counterfeiting, enrollment time, database storage requirements, intrusiveness, and cost.

2.1.5 Biometric Testing

The identifying power of a particular biometric encompasses two terms: False Rejection Rate (FRR), or a Type I Error, and False Acceptance Rate (FAR), or a Type II Error. False Rejection Rate and False Acceptance Rate are complementary in determining how stringent a biometric device is in allowing access to individuals.

As a result, biometric devices commonly include features to allow for variable threshold or sensitivity settings. For example, if the false acceptance rate threshold is increased to make it more difficult for impostors to gain access, it also will become harder for authorized people to gain access. As FAR goes down, FRR rises. On the other hand, if the false acceptance threshold is lowered as to make it very easy for authorized users to gain access, then it will be more likely that an impostor will slip through. Hence, as FRR goes down, FAR rises. In understanding the impact of FRR and FAR rates consider an automated teller machine (ATM) access system: a "False Acceptance" means you may lose a few dollars, whereas a "False Rejection" means you may lose a valuable customer (FAR POINT Consulting Inc., 1997). Another good example in understanding the inverse relationship of FRR and FAR rates, involves a car alarm. When your car alarm is very sensitive, the probability of the bad guys stealing it is low. Yet the chance of your accidentally setting off the alarm is high. Reduce the sensitivity, and the number of false alarms goes down, but the chance of someone stealing your car increases (Recognition Systems, Inc. 1999).

While the terms "false reject" and "false accept" are still commonly used in quantifying a biometrics' ability to rightfully identify an individual, the federal government has recently adopted a new standard of error rate measurement. Dr. Jim Wayman, Director of the United States National Biometric Test Center, has promoted the terms "false match" and "false non-match" as the new de facto terminology in determining a biometrics identifying power. According to Dr. Wayman, the problem with the terms "false accept" and "false reject" and even more so with "Type I" and "Type II" errors is that their meaning depends upon the claim of the user. For example, depending upon the biometric application, users make either a positive or a negative claim to identity. In a positive identification system, a rejection occurs if a person in not matched to a claimed record. In a negative identification system, a rejection occurs if a person is matched to a non-claimed record. Consequently, the words "false reject/accept" have opposite meanings, depending upon whom you are speaking to (Wayman, J. 1999).

2.2 Hand Vein

The vein patterns are the vast network of blood vessels underneath a person’s skin. The vein pattern in the back of the hand is unique for each individual. Every individual has their own vein pattern and this also applies to identical twins as they also have different vein patterns. Because of this uniqueness of vein pattern, it provides a good distinction between individuals. In addition to that, vein patterns also differ for each hand. Right hand and left hand has different vein patterns.

Vein patterns are large robust internal patterns where it does not change with time except for the size of the hand. This causes the vein pattern in the back of the hand remains stable over a long period of time. Apart from that; veins are hidden underneath the skin. This makes the vein patterns not easily observed as it is invisible to human eyes. Hand vein is not affected by situation of the outer skin (e.g. dirty hand) and less vulnerable to attacks. Intruders will find it hard to forge, duplicate, replicate or reproduce the vein patterns as compared to other biometric features.

Vein patterns are not easily observed, damaged, obscured or changed. The properties of uniqueness, stability and strong immunity to criminal tampering, makes it a potentially good biometric which offers secure and reliable features for authentication system.

2.3 Comparison of Hand Vein and Other Biometric Features

Existing biometric system such as fingerprint, iris, face and voice are confronted with minor problem and are very much likely vulnerable to attacks. Fingerprint is the surface character of a body. When someone touches a surface, his or her fingerprint is easily accessible and this can cause the fingerprint pattern easily stolen or forged. Physical contact is needed for fingerprint scanning and due to this direct contact with the finger the sensor gets dirty. Thus, the possibility of successful matching is reduced. Success ratio also decrease when the usable finger is too wet or dry which causes the fingerprint image become blur or dilapidated. Sometimes, people may lose the useable finger; the skin on the usable finger is cut.

As for iris, the eye pattern can be easily captured and forged by using high resolution camera. In addition, people may find it is intrusive to align the eye with a camera to capture the eye pattern. This often gives uncomfortable feeling and cause displeasure to the eye. The iris is very small and it is very hard to scan from distance. In addition, people who is blind and have cataract pose difficulty in reading the iris. The camera is highly sensitive to natural body motion, as the camera view has to be narrow to capture the resolution of the iris, it is.

The face is greatly affected by growth and time. Face features of an adult will be different when compared to their teenage or childhood years. Face also affected by variable lights and shadows. These uncertainties cause the system to confuse. Facial expression, facial hairs do play part in face recognition.

As for voice, this biometric trait is easily affected by environment. This biometric is not reliable in noisy environment like public places. Moreover, this type of system is sensitive to hoarse throat condition when people are sick with colds. Voice can be easily stolen with much more advanced technology which is available in the market.

Unlike fingerprint, iris, voice and face where the biological information is being scanned on the exterior of the body, hand vein authentication scans information in the interior of the body and therefore makes falsification extremely difficult. Thus, hand vein authentication has emerged as a promising component of biometrics study. Table 1 compares the biometric systems.

Table 2.1 Comparison of Biometric Systems

Biometrics	Universality	Uniqueness	Permanence	Collectability	Performance	Acceptability	Circumvention
Face	High	Low	Medium	High	Low	High	High
Fingerprint	Medium	High	High	Medium	High	Medium	High
Hand Geom.	Medium	Low	Low	High	Low	High	Medium
Hand Vein	Medium	Medium	Medium	Medium	Medium	Medium	Low
Iris	High	High	High	Medium	High	Low	Low
Retina	High	High	Medium	Low	High	Low	Low
Ear	Medium	Medium	High	Medium	Medium	High	Medium
Signature	Low	Low	Low	High	Low	High	Low
Voice	Medium	Low	Low	Medium	Low	High	Low
Thermogram	High	High	Low	High	Medium	High	High

2.4 Hand Vein Processing

Biometric hand vein system involves four stages. The stages are image acquisition, image pre-processing, feature extraction and image matching. Image acquisition is done by the modified webcam while image pre-processing is done at PC. Canny edge detection and matching are done at the FPGA board.

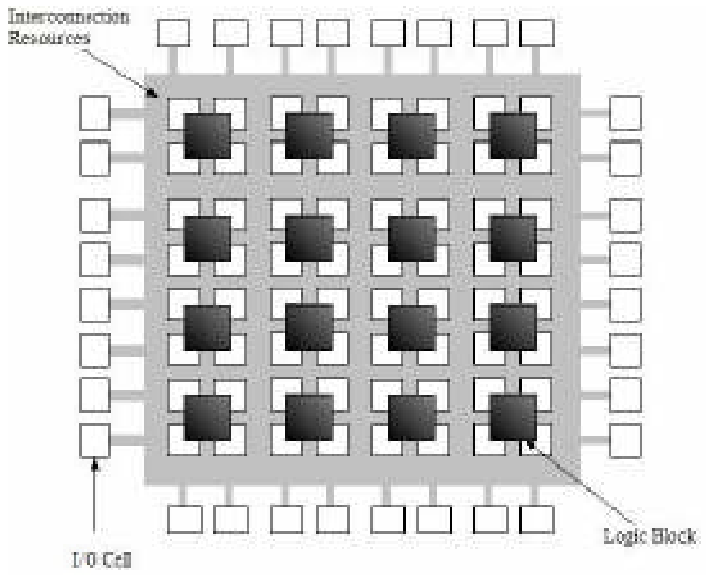

Figure 2.6 FPGA Board

A Field Programmable Gate Array (FPGA) as name suggests is a programmable device in which the final logic structure can be directly configured by the end user for a variety of applications. In its simplest form an FPGA consists of an array of uncommitted elements that can be programmed or interconnected according to a user's specification. The ability to reprogram these devices over and over again of the flexibility of interconnection resources makes FPGAs an ideal device for implementing & testing ASIC prototypes. The Figure 2.6, portrays the architecture of a conceptual FPGA.

2.4.1 Image Acquisition

Hand vein image was captured by using specially designed camera which is connected to a computer. This camera is a thermal camera modified from a webcam. By illuminating infrared light beam at the back of the hand, figure 2.7, the vein patterns appeared darker and shadowed as hemoglobin in the blood absorb the infrared light.

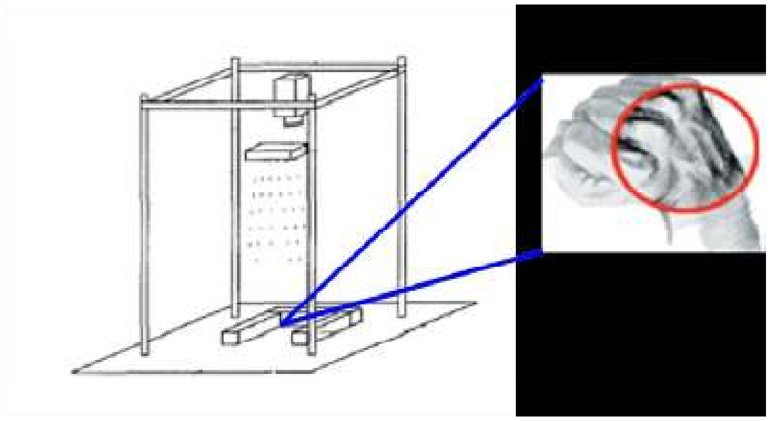

Figure 2.7 Initial images

2.4.2 Image Pre-processing

From this part, image processing is done. Captured image is transferred to PC so that the captured image can be processed further. This stage is essential because in this stage the captured image is enhanced and processed before feature extraction is done. There are nine sub processes in this stage.

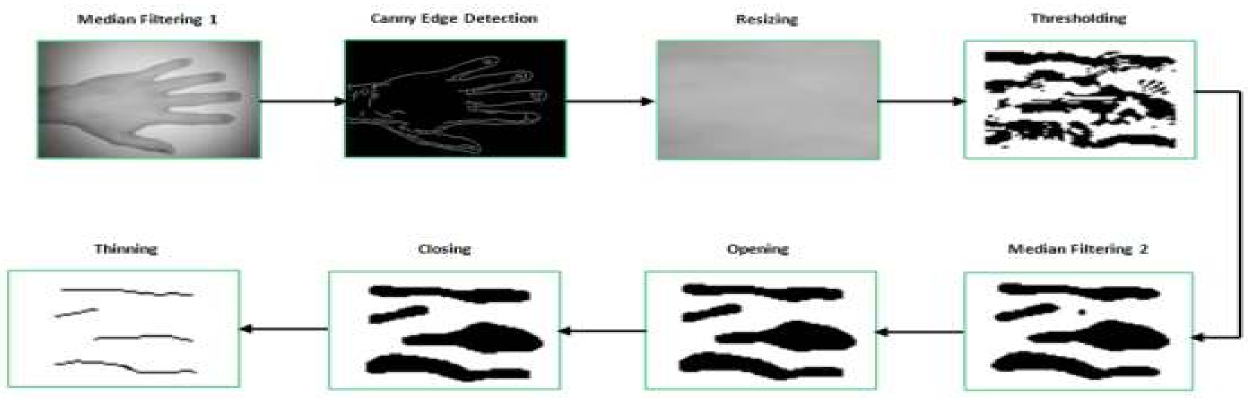

Figure 2.8 Overview of Image Pre-processing

2.4.3 Edge Detection

Edge detection is a fundamental tool used inmost image processing applications to obtain information from the frames as a precursor step tofeature extraction and object segmentation. This process detects outlines of an object and boundaries between objects and the background in the image. An edge-detection filter can also be used to improve the appearance of blurred or anti-aliased video streams

2.4.4 Image Matching

After the image has been processed and the required features are extracted, image matching is done. Different algorithms can be followed for image matching depending upon the nature of database and requirement .the most commonly used for biometric recognition are hamming and hausdroff distance.

2.5 Running an Application

2.5.1 Command Line Interface

A command line interface or CLI is a method of interacting with a computer by giving it lines of textual commands either from keyboard input or from a script. Command line interface was originally developed for interfacing with computers over teletype machines in the 1950s. It is occasionally also referred to as a CLUE, for Command Line User Environment. Some argue that the CLI is not actually a user interface at all, but a programming language, entered one line at a time, and has very little utility for users compared to developers. Indeed, command lines are most often used in scientific or engineering environments for programming.

2.5.2 Graphic User Interface

A graphical user interface (GUI) is a type of user interface which allows people to interact with electronic devices such as computers, hand-held devices and etc. rather than text based commands. GUI is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. A GUI uses a combination of technologies and devices to provide a platform the user can interact with, for the tasks of gathering and producing information. A GUI offers graphical icons, and visual indicators. The actions are usually performed through direct manipulation of the graphical elements.

CHAPTER 3

SYSTEM MODULES

3.1 Project Workflow

This chapter will discuss about the method and alternatives that have been used to make this project successful. This method includes the discussion of the project workflow, followed by the system design procedure, techniques and tools utilized in this work.

This project involved the effort of embedded system design process, which involves interface design and software development. Hence, it calls for software system design, in which the software are designed and downloaded in an embedded system.

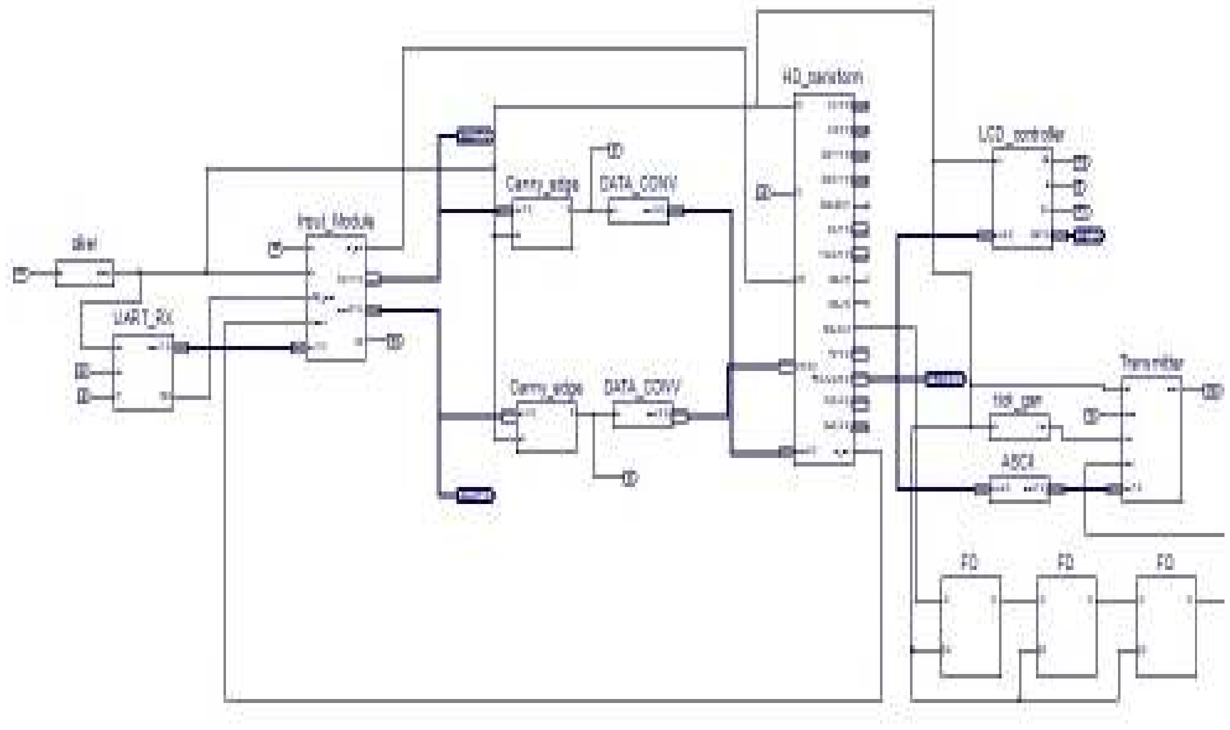

Figure 3.1– Project Block Diagram

3.2 Program Planning

The works continues with the literature reviews on state-of-the-art biometrics technologies and hand vein authentication system. It is important to understand the fundamental concept and operations carried out in the biometric hand vein system.

To design the system, several tools and techniques have to be familiarized and mastered. These include familiarizing with IR band. This system is running on Microsoft windows, thus its greater understanding were required to make this project successful. Matlab was understood thoroughly to create and design the graphic user interface. For co-harware simulation initially, block approach on xilinks system generator was understood and later same was used to extract the code.

The project workflows continue with designing and creating the GUI, developing the complete prototype. The process is repeated until the successful prototype is obtained. Finally, the GUI running on host PC is interfaced with the modified webcam. This GUI is used to get data from the hand vein infrared camera, image is captured in Matlab, then sent to the FPGA board, processing is done on the board, and then data is sent back to Matlab. At the end, output is displayed, both on Matlab, as well as character LCD of the FPGA board.

3.3 The Hardware:

The hardware of the project shown in figure 3.2 was designed in such a way to ensure that it is cost effective and fulfills all the requirements of the project.

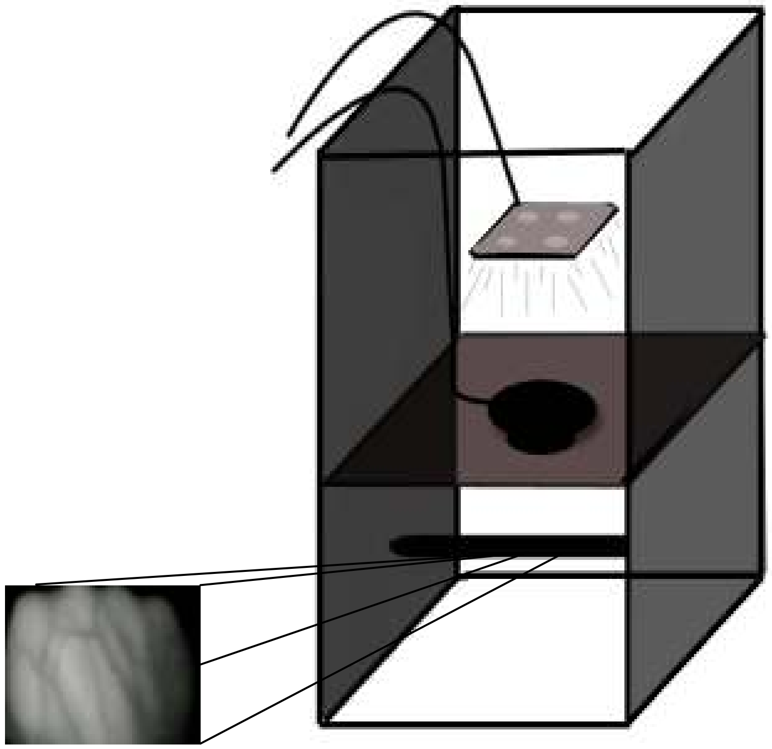

Figure 3.2 The Hardware

3.3.1 The Setup:

The requirement of the project was that image acquisition should be done in a confined environment. The reason for this was to make sure that the physical conditions like temperature, light ,humidity etc are least altered during the course of database compilation as these factors greatly affect the image acquisition.This results in different alterations to observed in the image preprocessing phase.

The setup is made of wood with dimensions of 12*10*18 inches .The inner walls are blackened as to ensure minimum reflection .A glass serves as a base to mount the modified camera and also for the diffusion paper.

To ensure that the subject's hand remains in the same place every time a bar has been placed. The subject is instructed to hold the bar which is supported by a rest at the bottom .Thus every time the hand is probably at the same place.

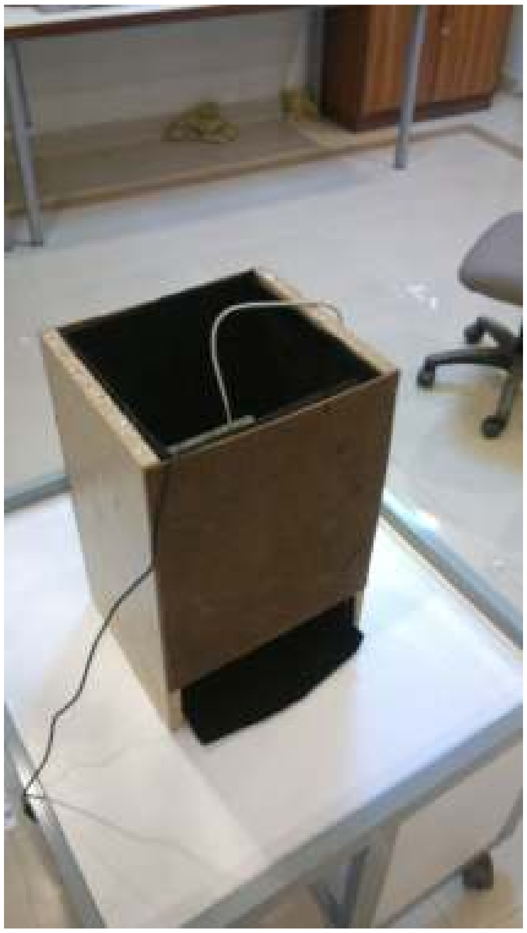

Figure 3.3 Setup

3.3.2 Modifying the camera:

One of the main aim of this project was to make it cost effective , for this reason the camera which are available to come up to the requirement of the project were not suitable so a simple low cost camera was modified. The objective of modification was to make the camera IR sensitive .Generally the cameras have an infrared filter just before the lens, which stops the IR rays from entering the lens of the camera .this filter is made of opaque red glass and is simply aligned just in front of the lens.

The modification figure 3.4 is done by opening the camera carefully so that the lens is not damaged. The filter is removed and the camera is closed again. Now this camera is capable of detecting the IR rays.

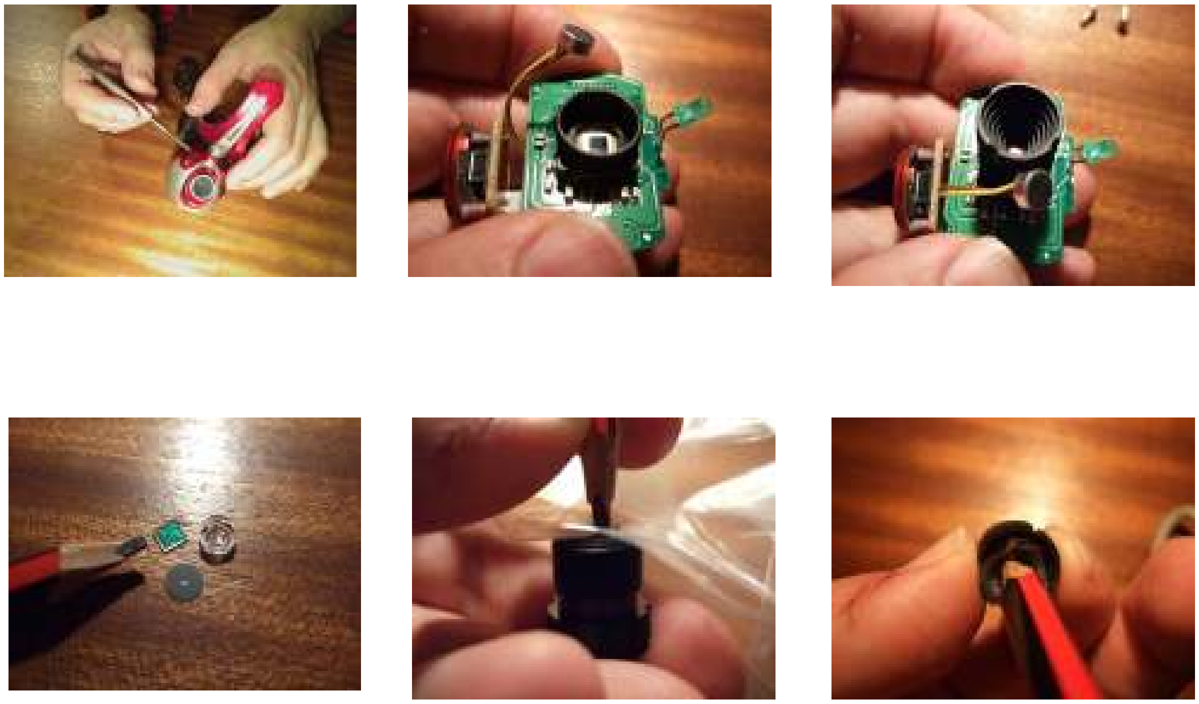

Figure 3.4 Modifying the Camera

3.3.3 Infrared source:

The infrared source used in the project is a circular array of concentric LED's that operates in near infrared .This source is attached at the top of the hand on the glass base .The LED light falls on the diffusion paper and is diffused, the light that is reflected from the walls also passes through the diffusion paper and then falls on the hand. The infrared source is operating on 12 V and in near infrared region (750 nm)

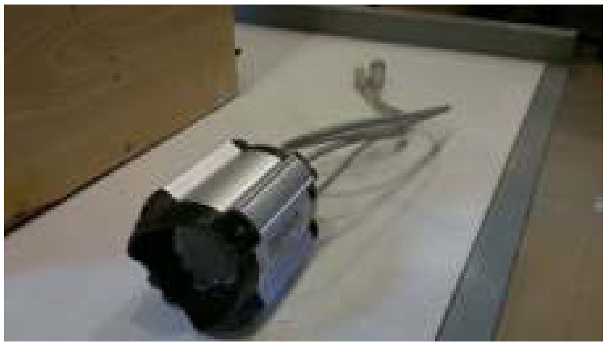

Figure 3.5 Infrared Source

3.3.4 Diffusing the Light:

A diffusion paper, figure 3.6, is used to evenly distribute the light on the subject's hand. This paper is stretched across the complete surface of the glass.

Figure 3.6 Diffusing Paper

3.4 The Techniques:

Different techniques have been used. Each of them is described in detail.

3.4.1 Canny edge detection

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny's aim was to discover the optimal edge detection algorithm. In this situation, an "optimal" edge detector means: Good detection means the algorithm should mark as many real edges in the image as possible. Good localization means edges marked should be as close as possible to the edge in the real image. Minimal response means a given edge in the image should only be marked once, and where possible, image noise should not create false edges.

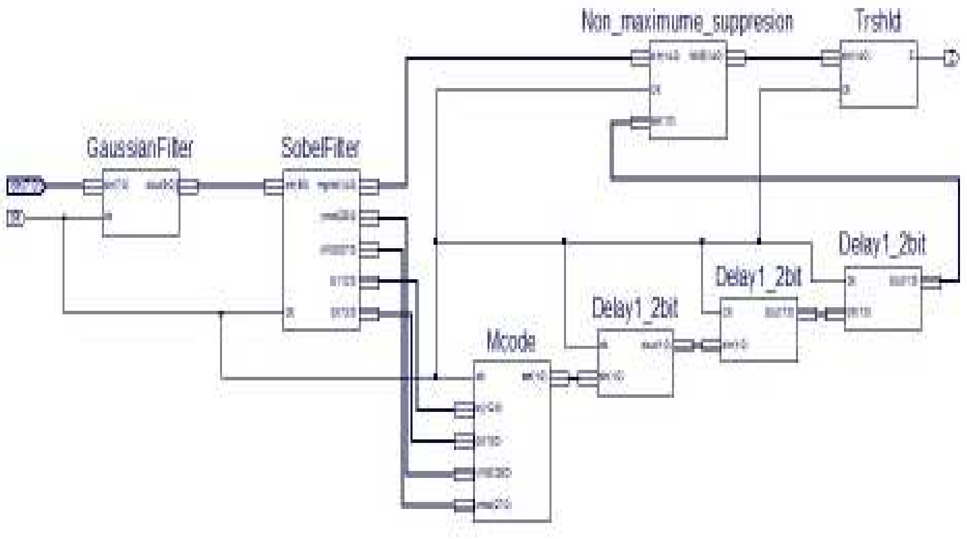

Figure 3.7 – Canny Edge Detection Model

To satisfy these requirements Canny used the calculus of variations – a technique which finds the function which optimizes a given functional. The optimal function in Canny's detector is described by the sum of four exponential terms, but can be approximated by the first derivative of a Gaussian.

3.4.2 Stages of the Canny algorithm

3.4.2.1 Gaussian filter for Noise reduction

The Canny edge detector uses a filter based on the first derivative of a Gaussian, because it is susceptible to noise present on raw unprocessed image data, so to begin with, the raw image is convolved with a Gaussian filter. The result is a slightly blurred version of the original which is not affected by a single noisy pixel to any significant degree. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen. Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function; this is also known as a two-dimensional Weierstrass transform. Since the Fourier transform of a Gaussian is another Gaussian, applying a Gaussian blur has the effect of reducing the image's high-frequency components; a Gaussian blur is thus a low pass filter. Using a Gaussian Blur filter before edge detection aims to reduce the level of noise in the image, which improves the result of the following edge-detection algorithm.

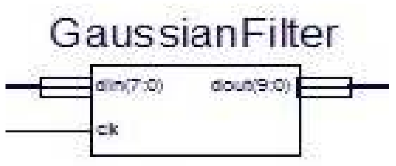

Figure 3.8 – Guassian Filter Model

An example of a 5x5 Gaussian filter is shown in equation 1, that can be used to filter the image with σ = 1.4.

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

……… Equation 1

3.4.2.2 Sobel Filtering (Finding the intensity gradient of the image)

Sobel filter computes an approximation of the gradient of the image intensity function. At each point, its result is corresponding gradient vector.

In simple terms, the operator calculates the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point, and therefore how likely it is that that part of the image represents an edge, as well as how that edge is likely to be oriented.

An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operatorSobel returns a value for the first derivative in the horizontal direction (Gx) and the vertical direction (Gy). From this the edge gradient and direction can be determined.

The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example).

Mathematically, the gradient of a two-variable function (here the image intensity function) is at each image point a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each image point, the gradient vector points in the direction of largest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction. This implies that the result of the Sobel operator at an image point which is in a region of constant image intensity is a zero vector and at a point on an edge is a vector which points across the edge, from darker to brighter values.

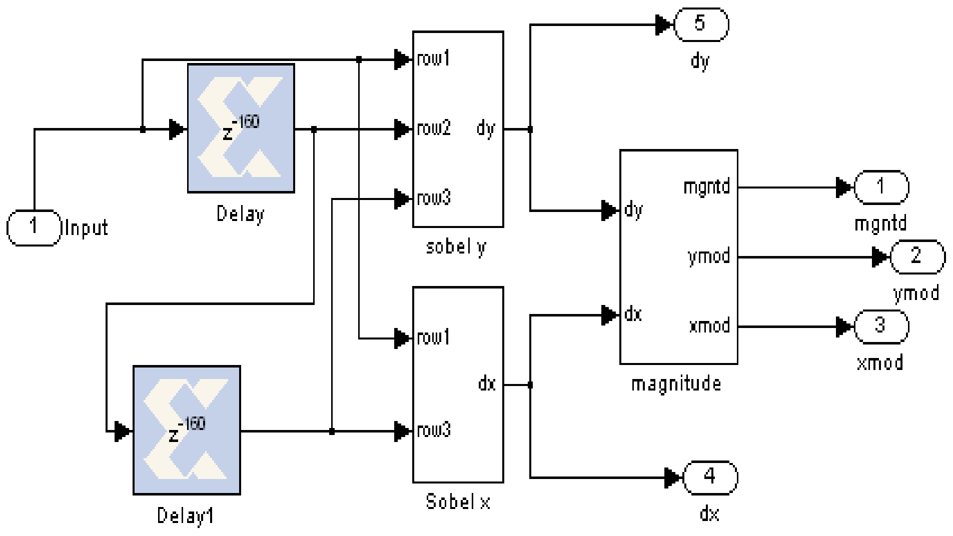

Figure 3.9 – Sobel Filter Model

Figure 3.9, shows a Sobel Filter Model. The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. For approximations, the computations are as follows:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

Equation 2 Equation 3

where * here denotes the 2-dimensional convolution operation.

The *x*-coordinate is here defined as increasing in the "right"-direction, and the *y*-coordinate is defined as increasing in the "down"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

Equation 4

Using this information, we can also calculate the gradient's direction:

$$\mathbf{\Theta} = \operatorname{arctan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right).$$

Equation 5

where, for example, **Θ** is 0 for a vertical edge which is darker on the right side.

3.4.2.3 Non-maximum suppression

To get rid of ridges, the edge strength of each candidate edge pixel is set to zero if its edge strength is not larger than the edge strength of the two adjacent pixels in the gradient direction.

Given estimates of the image gradients, a search is then carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction. So, for example, if the rounded gradient angle is zero degrees (i.e. the edge is in the north-south direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes in the west and east directions, if the rounded gradient angle is 90 degrees (i.e. the edge is in the east-west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes in the north and south directions.

Similarly, if the rounded gradient angle is 135 degrees (i.e. the edge is in the north east-south west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes in the north west and south east directions and if the rounded gradient angle is 45 degrees (i.e. the edge is in the north west-south east direction)the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes in the north east and south west directions.

From this stage referred to as non-maximum suppression, a set of edge points, in the form of a binary image, is obtained. These are sometimes referred to as "thin edges".

3.4.2.4 Tracing edges through the image and hysteresis thresholding

Large intensity gradients are more likely to correspond to edges than small intensity gradients. It is in most cases impossible to specify a threshold at which a given intensity gradient switches from corresponding to an edge into not doing so. Therefore Canny uses thresholding with hysteresis.

Two thresholds are selected, high and low. If the any edge is above high, it is considered a definite edge. If any edge is below low, it is not considered as an edge. If it is in between high and low, it is considered as may be definite edge. This category is properly analyzed to finally decide whether it is an edge or noisy pixel.

Making the assumption that important edges should be along continuous curves in the image allows us to follow a faint section of a given line and to discard a few noisy pixels that do not constitute a line but have produced large gradients. Therefore we begin by applying a high threshold. This marks out the edges we can be fairly sure are genuine. Starting from these, using the directional information derived earlier, edges can be traced through the image. While tracing an edge, we apply the lower threshold, allowing us to trace faint sections of edges as long as we find a starting point.

Once this process is complete, we have a binary image where each pixel is marked as either an edge pixel or a non-edge pixel. From complementary output from the edge tracing step, the binary edge map obtained in this way can also be treated as a set of edge curves, which after further processing can be represented as polygons in the image domain.

3.4.3 FPGA Implementation

Gaussian filter is used to reduce noise and minimize extra details of the image to perform smooth edge detection with the help of 2D-convolution. 2D-Convolution is most important to modern image processing. The basic idea is that a window of some finite size and shape is scanned over an image. The output pixel value is the weighted sum of the input pixels within the window where the weights are the values of the filter assigned to every pixel of the window. The window with its weights is called the convolution mask. Mathematically, convolution on image can be represented by the following equation.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \text{...Equation 5}$$

where *x* is the distance from the origin in the horizontal axis, *y* is the distance from the origin in the vertical axis, and *σ* is the standard deviation of the Gaussian distribution. When applied in two dimensions, this formula produces a surface whose contours are concentric circles with a Gaussian distribution from the center point. Values from this distribution are used to build a convolution matrix which is applied to the original image. Each pixel's new value is set to a weighted average of that pixel's neighborhood. The original pixel's value receives the heaviest weight (having the highest Gaussian value) and neighboring pixels receive smaller weights as their distance to the original pixel increases. This results in a blur that preserves boundaries and edges better than other, more uniform blurring filters.

An important aspect of convolution algorithm is that it supports a virtually infinite variety of masks, each with its own feature. This flexibility allows many powerful applications.

The idea of Gaussian convolution is to use this 2-D distribution as a point spread function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels. A discrete approximation to the Gaussian function is required to perform the convolution. In theory, the Gaussian distribution is non- zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so convolution kernel is truncated.

The 5x5convolution mask has been used by us to get better noise reduction

$$\frac{1}{115}\begin{array}{|c|c|c|c|c|}\hline 2 & 4 & 5 & 4 & 2 \\ \hline 4 & 9 & 12 & 9 & 4 \\ \hline 5 & 12 & 15 & 12 & 5 \\ \hline 4 & 9 & 12 & 9 & 4 \\ \hline 2 & 4 & 5 & 4 & 2 \\ \hline \end{array}$$

…..Equation 7

The system generator model created to get the results is shown in figure 3.10.

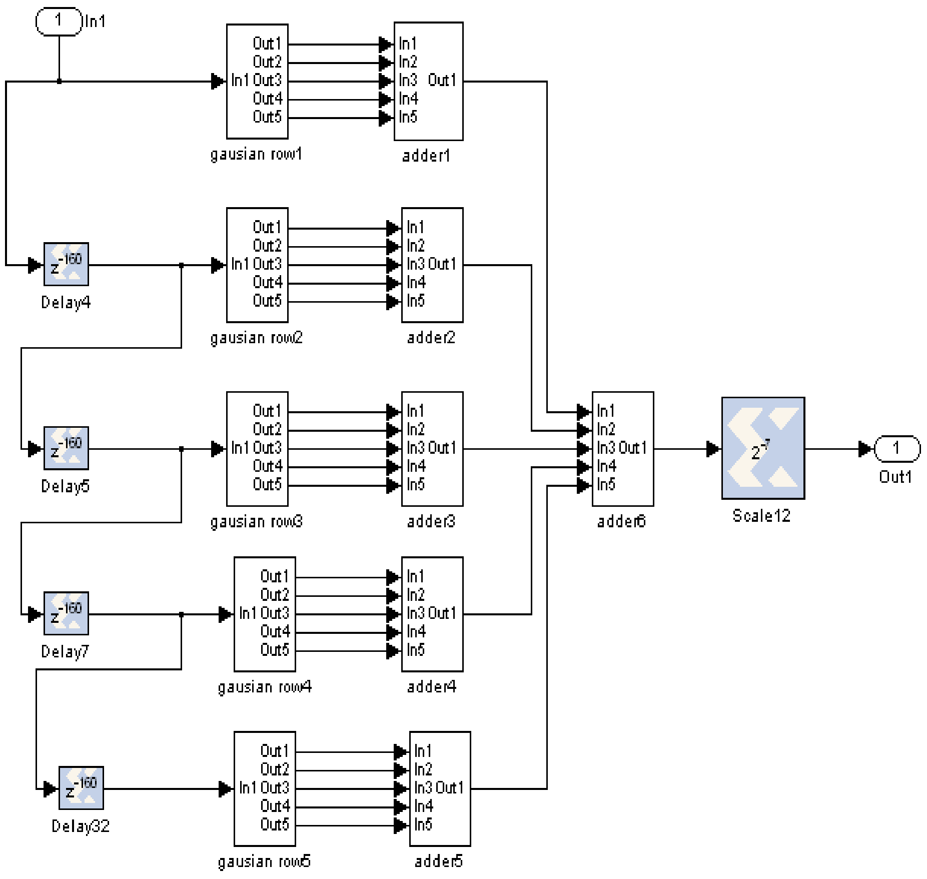

Figure 3.10 – System Generator Model for Guassian Filter

The images obtained after applying the noise reduction are shown in figure 3.11.

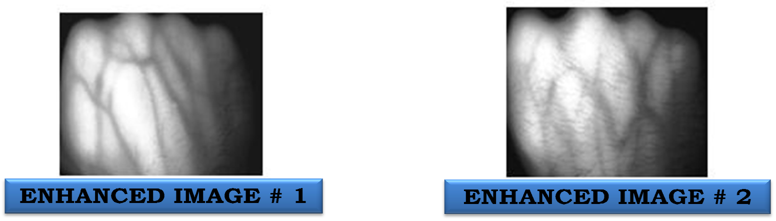

Figure 3.11 –Results

Since the convolution mask is fixed for the whole image a dedicated hardware can be designed. Some of the window coefficients are contains multiple of 2. The multiplication of these coefficients with the corresponding pixels in the window can be carried out using left shift operations and the non powers of 2 digits can be implemented using multiplier less multiplication.

Division is also very expensive operation on FPGA, instead of using division operator it is much simple to use right shift operator, so a divide by 128 was implemented instead of divide by 115.

3.4.4 Image Matching

Matching technique used was hausdorff Distance transform. In mathematics, the Hausdorff distance, or Hausdorff metric, also called Pompeiu–Hausdorff distance,[1] measures how far two subsetsof a metric space are from each other. It turns the set of non-empty compact subsets of a metric space into a metric space in its own right. It is named after Felix Hausdorff.

Informally, two sets are close in the Hausdorff distance if every point of either set is close to some point of the other set. The Hausdorff distance is the longest distance you can be forced to travel by an adversary who chooses a point in one of the two sets, from where you then must travel to the other set. In other words, it is the farthest point of a set that you can be to the closest point of a different set.

It may be defined as: Let *X* and *Y* be two non-empty subsets of a metric space (*M*, *d*). We define their Hausdorff distance $d_H(X, Y)$ by

$$d_{\mathrm{H}}(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\},$$

…..Equation 8

where *sup* represents the supremum and *inf* the infimum.

Equivalently

$$d_H(X,Y) = \inf\{\epsilon > 0\,;\ X \subseteq Y_\epsilon \text{ and } Y \subseteq X_\epsilon\},$$ ....Equation 9

where

$$X_\epsilon := \bigcup_{x \in X} \{z \in M\,;\ d(z,x) \le \epsilon\}$$,..............................Equation 10

That is, the set of all points within ε of the set *X* (sometimes called a generalized ball of radius ε around *X*).

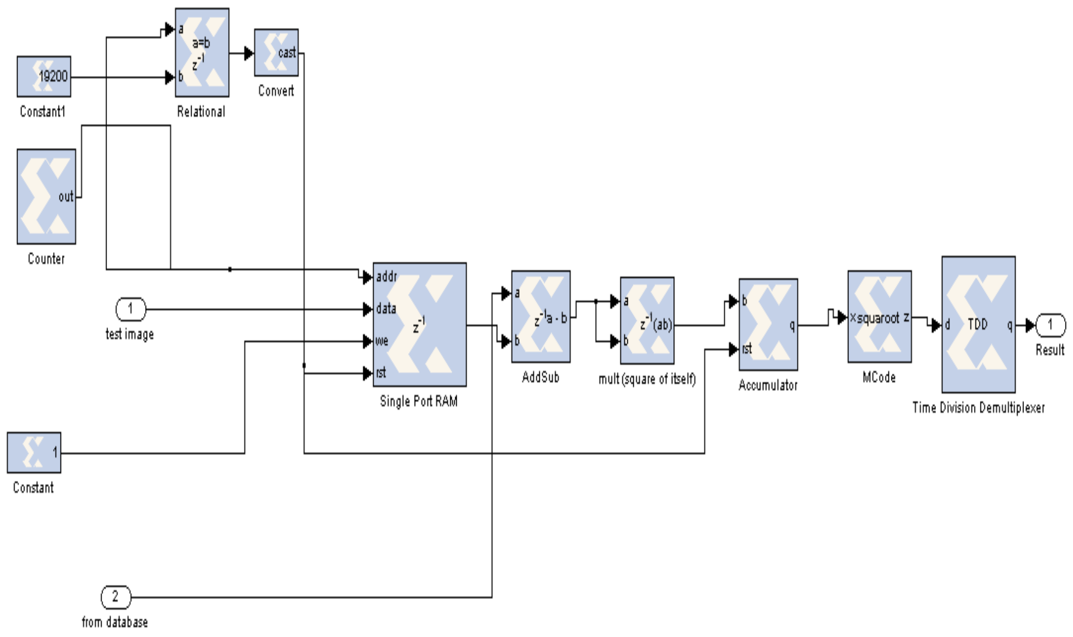

Figure 3.12 – Hausdorff Transform Model

In computer vision, the Hausdorff distance can be used to find a given template in an arbitrary target image. The template and image are often pre-processed via an edge detector giving a binary image. Next, each 1 (activated) point in the binary image of the template is treated as a point in a set, the "shape" of the template. Similarly, an area of the binary target image is treated as a set of points.

The algorithm then tries to minimize the Hausdorff distance between the template and some area of the target image. The area in the target image with the minimal Hausdorff distance to the template can be considered the best candidate for locating the template in the target.In Computer Graphics the Hausdorff distance is used to measure the difference between two different representations of the same 3D object particularly when generating level of detail for efficient display of complex 3D models.

CHAPTER 4

APPLICATION DEVELOPMENT AND WORKING:

4.1 MAIN MENU

The figure 4.1 shows the main menu of the graphical user interface. It contains the buttons for creating a new user, verifying an existing user, deleting an existing user and closing the program.

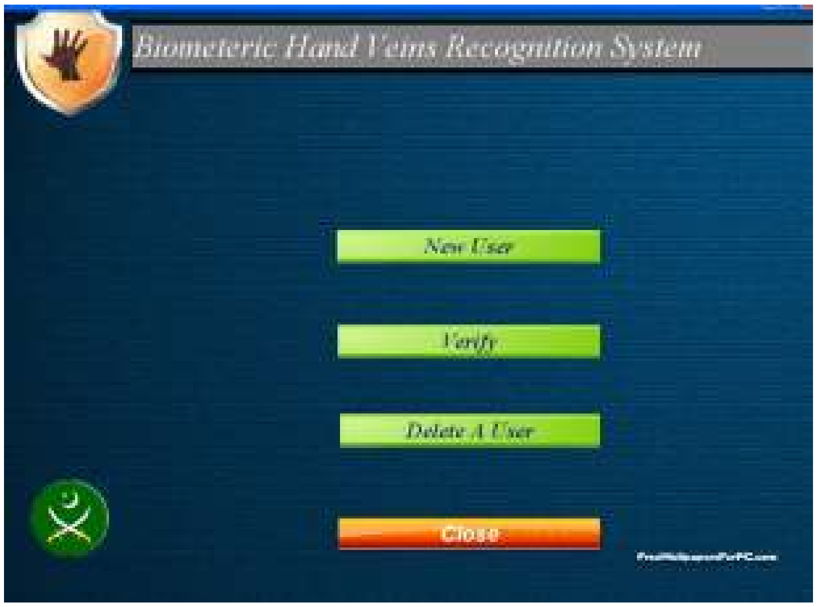

Figure 4.1 Main menu

4.2 ADD NEW USER

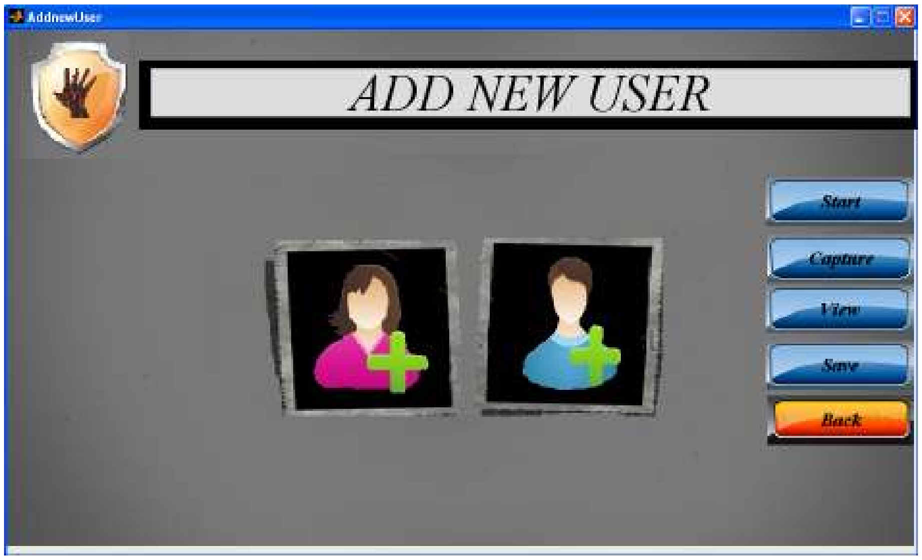

Figure 4.2 Add user

4.3 IDENTIFICATION

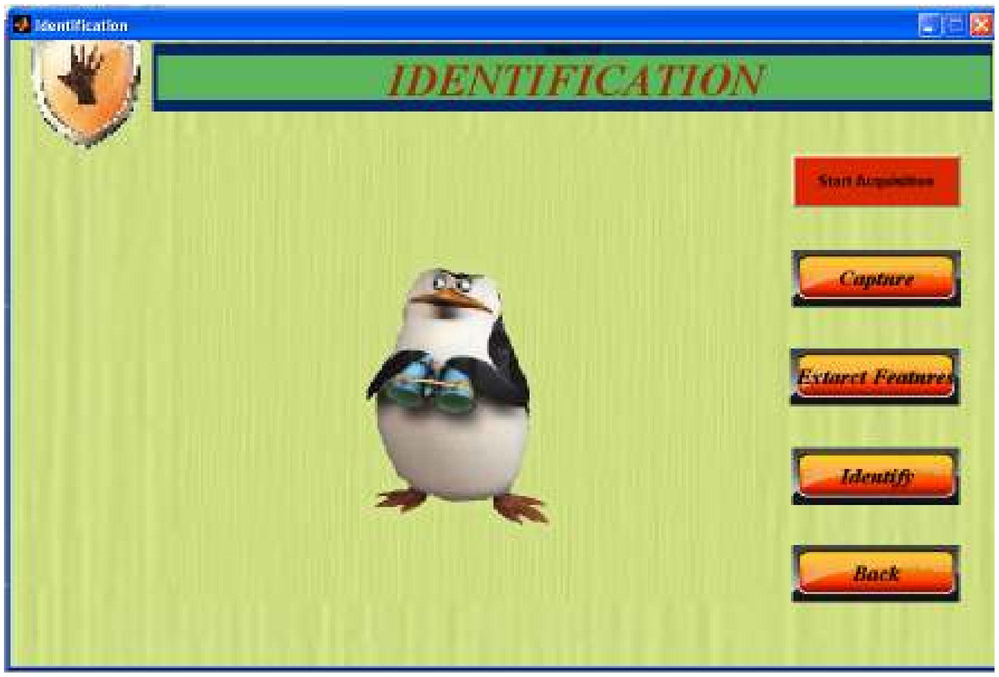

Figure 4.3 Identification

4.4 MATCHING

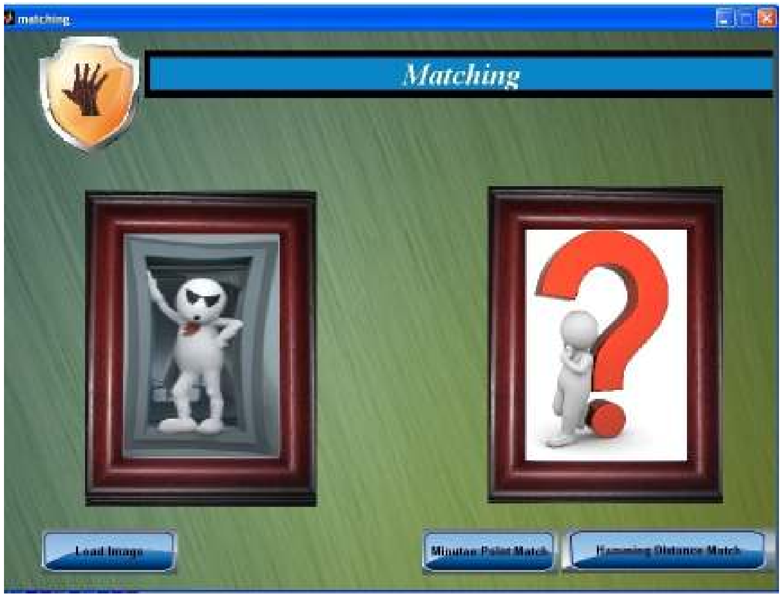

Figure 4.4 Matching

4.5 RESULTS:

The database has been compiled .Among which asubject ischosen to show the following development of software part:

Acquired image as shown in figure 4.5

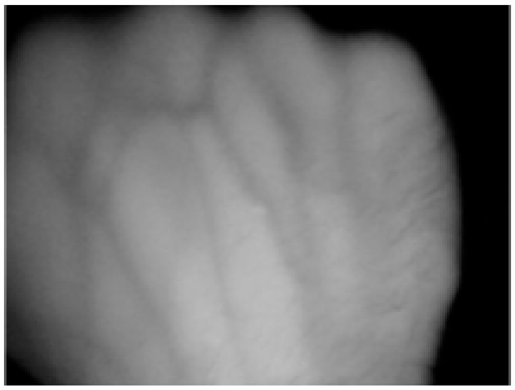

Figure 4.5 Sample from data base

Extracted features are shown in figure 4.6.

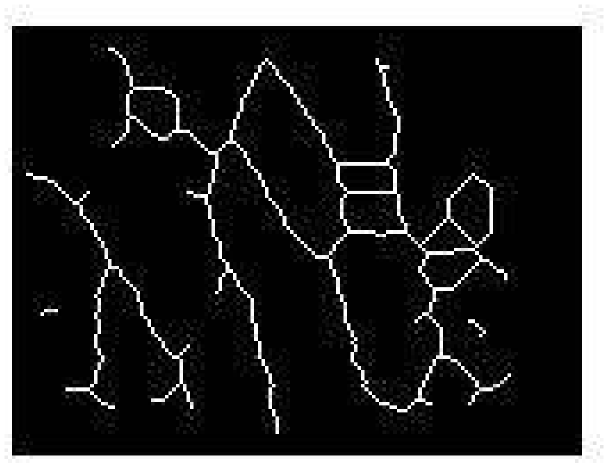

Figure 4.6 Extracted Features

Recognition rates of up to 80% of correctly classified people out of 50 attempts were achieved by the program, further adding to the notion that hand vein can be used as a biometric. We are sure that with more advanced classification techniques, higher rates of recognition could have been achieved.

4.6 FUTURE IMPROVMENTS:

A memory module can be directly interfaced with FPGA board to enhance the memory and reduce the data transfer delays. Hardware language can be written to accelerate the time taken for image to be processed. By doing so, time to process a hand vein image can be reduced. A script file can be written in order to execute front end and back end. At the moment, front end and back end are executed separately. And execution of front end and back end separately is really troublesome sometimes. The best of results can be achieved by interfacing a webcam directly with the fpga board so that the time delay for capturing an image and sending it to fpga board can be minimized. The total processing time of implementation of project on fpga can also be reduced by using a more powerful fpga board but that will increase the cost. So a compromise between performance and cost has to be made for effective implementation.

REFRENCES

1. P. C. Eng and M. K. Hani, “Hand Vein Biometric Authentication System," in IEEE TENCON 2009, Singapore, 2009.

2. Zhang Yu, Han Xiao and Ma Si Liang, “ Feature Extraction of Hand- Vein Patterns Based on Rigelet Transform and Local Interconnection Structure Neural Network”, Springer-Verlag Berlin Heidelberg 2006

3. Wang Lingyu, Leedham Graham and David Cho Siu-Yeung, “ Minutiae Feature Analysis for Infrared Hand Vein Pattern Biometrics”, Pattern Recognition Society, Elsevier, July 2007

4. Chen Liukui, Zheng Hong, Li Li, XiePeng and Liu Shuo, “ Near-infrared Dorsal hand Vein Image Segmentation by Local Thresholding Using Grayscale Morphology,” IEEEXplore

5. SuleymanMalki and Spaaneburg Lambert, “Hand Veins Feature Extraction Using DT-CNNS”, Lund University (Lund Sweeden)

6. Wang Kejun, Ding Yuhang and Zhuang Dayan, “A Study of Hand Vein Recognition Method,” International Conference on Mechatronics and Automation, July 2205

7. http://www.hitachi-ics.co.jp/product/english/about_fv.htm

8. Penny Khaw, “Iris Technology For Improved Authentication”, SANS Security Essentials (GSEC), 2002

9. Lu Xiaoguang, “Image Analysis for Face Recognition”, Michigan State University, East Lansing

10.http://dsonline.computer.org/portal/cms_docs_dsonline/dsonline/topics/os/embedded

11. http://en.wikipedia.org/wiki/Universal_Serial_Bus

12. http://en.wikipedia.org/wiki/Real-time_operating_system

13. http://en.wikipedia.org/wiki/Command-line_interface

14. http://www.spiritus-temporis.com/command-line-interface/disadvantages-of-a-command-line-interface.html

15. www.micahcarrick.com

APPENDIX A

-Source Code of the GUI for Main Menu-

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @Main_OpeningFcn, ...
'gui_OutputFcn',  @Main_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Main is made visible.
function Main_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Main (see VARARGIN)

% Choose default command line output for Main
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes Main wait for user response (see UIRESUME)
% uiwait(handles.figure1);
[a,map]=imread('gui_button_frame_clip_art_9825.jpg');
[r,c,d]=size(a);
x=ceil(r/57);
y=ceil(c/493);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.NewUser,'CData',g);

[a,map]=imread('gui_button_frame_clip_art_9825.jpg');
```

```
[r,c,d]=size(a);
x=ceil(r/57);
y=ceil(c/493);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Verify,'CData',g);

[a,map]=imread('gui_button_frame_clip_art_9825.jpg');
[r,c,d]=size(a);
x=ceil(r/57);
y=ceil(c/493);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.DeleteAUser,'CData',g);

[a,map]=imread('orange.png');
[r,c,d]=size(a);
x=ceil(r/57);
y=ceil(c/493);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Close,'CData',g);

% --- Outputs from this function are returned to the command line.
function varargout = Main_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;




% --- Executes on button press in Close.
function Close_Callback(hObject, eventdata, handles)
% hObject    handle to Close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
delete all
close(gcf)



% --- Executes on button press in NewUser.
function NewUser_Callback(hObject, eventdata, handles)
% hObject    handle to NewUser (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

AddnewUser
```

```
% --- Executes on button press in Verify.
function Verify_Callback(hObject, eventdata, handles)
% hObject    handle to Verify (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all
delete all
identification

% --- Executes on button press in DeleteAUser.
function DeleteAUser_Callback(hObject, eventdata, handles)
% hObject    handle to DeleteAUser (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
prompt = {'Enter your name:','Enter your course:'};
dlg_title = 'Input for new entry database';
num_lines = 1;
def = {'name','course'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
deleteuser(answer(1),answer(2))
%addnewpic(answer(1),answer(2),image)

% --- Executes during object creation, after setting all properties.


% --- Executes during object creation, after setting all properties.
function axes3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
ax1=gcbo
axes(ax1)
figure_handle=gcf
set(figure_handle, 'Position', get(0,'ScreenSize'))
imshow('Blue hills.jpg')
% Hint: place code in OpeningFcn to populate axes3


% --- Executes during object creation, after setting all properties.
function axes5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes5
ax1=gcbo
axes(ax1)
imshow('Blue hills.jpg')



% --- Executes during object creation, after setting all properties.
function axes6_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to axes6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes6




% --- Executes during object creation, after setting all properties.
function sumthing_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sumthing (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate sumthing

ax2=gcbo
axes(ax2)
imshow('blue_001_001_1024x768.jpg')


% --- Executes during object creation, after setting all properties.
function logo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to logo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate logo
ax3=gcbo
axes(ax3)
imshow('logo.jpg')



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as
a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as
a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end





% --- Executes during object creation, after setting all properties.
function army_CreateFcn(hObject, eventdata, handles)
% hObject    handle to army (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate army
ax4=gcbo
axes(ax4)
imshow('army3.png')
```

APPENDIX B

-Source code for FPGA

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity accum is
    Port ( clk,rst : in  STD_LOGIC;
           din : in  STD_LOGIC_VECTOR (14 downto 0);
           dout : out  STD_LOGIC_VECTOR (14 downto 0));
end accum;

architecture Behavioral of accum is
signal reg_p,reg_n:std_logic_vector(14 downto 0):=(others=>'0');
begin
process(clk,rst)
begin
if rst='1' then
reg_p<=(others=>'0');
elsif rising_edge(clk) then
reg_p<=reg_n;
end if;
end process;
reg_n<="111111111111111" when reg_p="111111111111111" else
reg_p+din;
dout<=reg_p;

end Behavioral;
```

```
entity ander is
    Port ( clk : in  STD_LOGIC;
           a : in  STD_LOGIC_VECTOR (13 downto 0);
           b : in  STD_LOGIC_VECTOR (13 downto 0);
           c : out  STD_LOGIC_VECTOR (13 downto 0));
end ander;

architecture Behavioral of ander is
```

```
begin
process(clk)
begin
if rising_edge(clk) then
c<=(a and b);
end if;
end process;

end Behavioral;
```

```
entity ASCII is
    Port ( din : in  STD_LOGIC_VECTOR (3 downto 0);
           dout : out  STD_LOGIC_VECTOR (7 downto 0));
end ASCII;

architecture Behavioral of ASCII is

begin

dout<="0000" & din;
end Behavioral;
```

```
entity Baud_rate is
    Port ( clk : in  STD_LOGIC;
           tick : out  STD_LOGIC);
end Baud_rate;

architecture Behavioral of Baud_rate is
signal count_p,count_n:std_logic_vector(7 downto 0):="00000000";
begin
process(clk)
begin
if rising_edge(clk) then
count_p<=count_n;
end if;
end process;
count_n<="00000000" when count_p="10100011"
else count_p+"00000001";

tick<='1' when count_p="10100011"
else '0';
end Behavioral;
```

```
ntity Bin2Ascii is
    Port ( din : in  STD_LOGIC_VECTOR (3 downto 0);
           dout : out  STD_LOGIC_VECTOR (7 downto 0));
end Bin2Ascii;

architecture Behavioral of Bin2Ascii is

begin
```

```
end Behavioral;
```

```
entity calculator is
    Port ( rst : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           load_FHD : in  STD_LOGIC;
           load_RHD : in  STD_LOGIC;
           din : in  STD_LOGIC_VECTOR (4 downto 0);
                  load: out STD_LOGIC;
                  HD  : out STD_LOGIC_VECTOR (10 downto 0);
                  FHD : out STD_LOGIC_VECTOR (10 downto 0);
                  RHD : out STD_LOGIC_VECTOR (10 downto 0));
end calculator;

architecture Behavioral of calculator is
type state is (s0,s1,s2);
signal prsnt_state,nxt_state:state:=s0;
signal count_p,count_n:integer:=0;
signal load_p,load_n:std_logic:='0';
signal temp_p,temp_n:std_logic_vector(4 downto 0):=(others=>'1');
signal FHD_p,FHD_n,RHD_p,RHD_n,HD_p,HD_n:std_logic_vector(10
downto 0):=(others=>'0');
begin
process(clk,rst)
begin
if rst='1' then
prsnt_state<=s0;
count_p<=0;
temp_p<=(others=>'1');
FHD_p<=(others=>'0');
RHD_p<=(others=>'0');
elsif rising_edge(clk) then
prsnt_state<=nxt_state;
count_p<=count_n;
temp_p<=temp_n;
FHD_p<=FHD_n;
RHD_p<=RHD_n;
HD_p<=HD_n;
load_p<=load_n;
end if;
end process;

process(prsnt_state,din,count_p,temp_p,FHD_p,RHD_p,load_FHD,load_
RHD,HD_p)
begin
nxt_state<=prsnt_state;
count_n<=count_p;
temp_n<=temp_p;
FHD_n<=FHD_p;
RHD_n<=RHD_p;
HD_n<=HD_p;
```

```
load_n<='0';

case prsnt_state is

--calculate forward hd--
when s0=>
load_n<='0';
if count_p=120 then
count_n<=0;
temp_n<=(others=>'1');
nxt_state<=s1;
elsif load_FHD='1' then
temp_n<=(others=>'1');
FHD_n<=FHD_p+temp_p;
count_n<=count_p+1;
elsif din<temp_p then
temp_n<=din;
end if;

--calculate reverse hd--
when s1=>
if count_p=120 then
count_n<=0;
temp_n<=(others=>'1');
nxt_state<=s2;
elsif load_RHD='1' then
temp_n<=(others=>'1');
RHD_n<=RHD_p+temp_p;
count_n<=count_p+1;
elsif din<temp_p then
temp_n<=din;
end if;
--take maximume--
when s2=>
if RHD_p > FHD_p then
HD_n<=RHD_p;
else
HD_n<=FHD_p;
end if;
FHD_n<=(others=>'0');
RHD_n<=(others=>'0');
load_n<='1';
nxt_state<=s0;

end case;
end process;
FHD<=FHD_p;
RHD<=RHD_p;
HD<=HD_p;
load<=load_p;
end Behavioral;
```

```
entity clker is
    Port ( clk : in  STD_LOGIC;
           clkout : out  STD_LOGIC);
end clker;

architecture Behavioral of clker is
signal reg_p,reg_n:std_logic:='0';
begin
process(clk)
begin
if rising_edge(clk)
then
reg_p<=reg_n;
end if;
end process;
reg_n<=not(reg_p);
clkout<=reg_p;
end Behavioral;
```

APPENDIX C

-Source Code for Identification Window -

```
function varargout = Identification(varargin)
% IDENTIFICATION M-file for Identification.fig
%      IDENTIFICATION, by itself, creates a new IDENTIFICATION or
raises the existing
%      singleton*.
%
%      H = IDENTIFICATION returns the handle to a new IDENTIFICATION or
the handle to
%      the existing singleton*.
%
%      IDENTIFICATION('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in IDENTIFICATION.M with the given input
arguments.
%
%      IDENTIFICATION('Property','Value',...) creates a new
IDENTIFICATION or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Identification_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Identification_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Identification

% Last Modified by GUIDE v2.5 03-Feb-2012 13:42:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @Identification_OpeningFcn, ...
'gui_OutputFcn',  @Identification_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Identification is made visible.
function Identification_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Identification (see VARARGIN)

% Choose default command line output for Identification
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Identification wait for user response (see UIRESUME)
% uiwait(handles.figure1);
global B1
setappdata(0,'IdentifyGUI',gcf);




[a,map]=imread('orange.png');
[r,c,d]=size(a);
x=ceil(r/57);
y=ceil(c/200);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Back,'CData',g);

[a,map]=imread('orange.png');
[r,c,d]=size(a);
x=ceil(r/57);
y=ceil(c/200);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Capture,'CData',g);

[a,map]=imread('orange.png');
[r,c,d]=size(a);
x=ceil(r/57);
y=ceil(c/200);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.Identify,'CData',g);

[a,map]=imread('orange.png');
[r,c,d]=size(a);
x=ceil(r/57);
```

```
y=ceil(c/200);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.ExtractFeature,'CData',g);


% --- Outputs from this function are returned to the command line.
function varargout = Identification_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in StartAcq.
function StartAcq_Callback(hObject, eventdata, handles)

if get(hObject, 'userdata') == 0, % video is closed
    imaqreset;

    Obj= videoinput('winvideo', 2,'YUY2_640x480');
    preview(Obj);
    set(hObject, 'string', 'Stop Acquisition');
    set(hObject, 'userdata', 1);
    guidata(hObject, handles);
    IdentifyGUI=getappdata(0,'IdentifyGUI');
    setappdata(IdentifyGUI,'vid',Obj);
return
else
    IdentifyGUI=getappdata(0,'IdentifyGUI');
    video=getappdata(IdentifyGUI,'vid');
    delete(video);

    set(hObject, 'string', 'Start Acqusition');
    set(hObject, 'userdata', 0);
    guidata(hObject, handles);

return
end


% --- Executes on button press in Capture.
function Capture_Callback(hObject, eventdata, handles)
% hObject    handle to Capture (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global B1
axes(handles.CapturedImg)
IdentifyGUI=getappdata(0,'IdentifyGUI');
video=getappdata(IdentifyGUI,'vid');
```

```
%capturedimg=imread('C:\Users\GoldenEagles\Desktop\database\database\da
tabase\a.jpg')
capturedimg = getsnapshot(video);
capturedimg=YUY2toRGB(capturedimg);
capturedimg=rgb2gray(capturedimg);
imshow(capturedimg);
capturedimg=imresize(capturedimg,[120 160]);
setappdata(IdentifyGUI,'image',capturedimg);


% --- Executes on button press in ExtractFeature.
function ExtractFeature_Callback(hObject, eventdata, handles)
% hObject    handle to ExtractFeature (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
IdentifyGUI=getappdata(0,'IdentifyGUI');
capturedimg=getappdata(IdentifyGUI,'image');
cropimg=capturedimg;

minutaepnts=im2serial(cropimg)

setappdata(IdentifyGUI,'crpimg',cropimg);
setappdata(IdentifyGUI,'minutaepoints',minutaepnts);


% --- Executes on button press in Identify.
function Identify_Callback(hObject, eventdata, handles)
% hObject    handle to Identify (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
matching

% --- Executes on button press in Back.
function Back_Callback(hObject, eventdata, handles)
% hObject    handle to Back (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
IdentifyGUI=getappdata(0,'IdentifyGUI');
video=getappdata(IdentifyGUI,'vid');
delete(video)
close(gcf)
Main

% --- Executes during object creation, after setting all properties.
function CapturedImg_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CapturedImg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate CapturedImg
% --- Executes during object creation, after setting all properties.
function axes5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes5




% --- Executes during object creation, after setting all properties.
function axes6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes6
axes(hObject)
imshow('686817.jpg')


% --- Executes during object creation, after setting all properties.
function axes9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes9
imshow('logo4.jpg')


% --- Executes during object creation, after setting all properties.
function input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate input


% --- Executes during object creation, after setting all properties.
function background_CreateFcn(hObject, eventdata, handles)
% hObject    handle to background (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate background


% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Obj= videoinput('winvideo', 2);

  preview(Obj);
  IdentifyGUI=getappdata(0,'IdentifyGUI');
  setappdata(IdentifyGUI,'vid',Obj);

 guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function axes11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes11
axes(hObject)
imshow('asdaf.jpg')



function ST6_Callback(hObject, eventdata, handles)
% hObject    handle to ST6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ST6 as text
%        str2double(get(hObject,'String')) returns contents of ST6 as a
double


% --- Executes during object creation, after setting all properties.
function ST6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ST6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function ST7_Callback(hObject, eventdata, handles)
% hObject    handle to ST7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ST7 as text
%        str2double(get(hObject,'String')) returns contents of ST7 as a
double


% --- Executes during object creation, after setting all properties.
function ST7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ST7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

APPENDIX D

-Source Code for Matching Window-

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @matching_OpeningFcn, ...
'gui_OutputFcn',  @matching_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before matching is made visible.
function matching_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to matching (see VARARGIN)

% Choose default command line output for matching
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes matching wait for user response (see UIRESUME)
% uiwait(handles.figure1);
axes(handles.LoadedImage)
imshow('05.jpg')
axes(handles.MatchingImage)
imshow('06.jpg')

[a,map]=imread('blue.png');
[r,c,d]=size(a);
x=ceil(r/57);
y=ceil(c/200);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.MinutaePointMatch,'CData',g);

[a,map]=imread('blue.png');
[r,c,d]=size(a);
x=ceil(r/57);
```

```
y=ceil(c/200);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.LoadImage,'CData',g);


[a,map]=imread('blue.png');
[r,c,d]=size(a);
x=ceil(r/57);
y=ceil(c/237);
g=a(1:x:end,1:y:end,:);
g(g==255)=5.5*255;
set(handles.HammingDistMatch,'CData',g);




% --- Outputs from this function are returned to the command line.
function varargout = matching_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in LoadImage.
function LoadImage_Callback(hObject, eventdata, handles)
% hObject    handle to LoadImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.LoadedImage)
IdentifyGUI=getappdata(0,'IdentifyGUI');
capturedimg=getappdata(IdentifyGUI,'image');


imshow(capturedimg)

% IdentifyGUI=getappdata(0,'IdentifyGUI');
% capturedimg=getappdata(IdentifyGUI,'image');
% imshow(capturedimg)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in MinutaePointMatch.
function MinutaePointMatch_Callback(hObject, eventdata, handles)
% hObject    handle to MinutaePointMatch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
axes(handles.MatchingImage);
IdentifyGUI=getappdata(0,'IdentifyGUI');
minutaepnts=getappdata(IdentifyGUI,'minutaepoints');
s=getappdata(IdentifyGUI,'s');

axes(handles.MatchingImage);
[s,b]=mtchminutae(minutaepnts)
stopasync(s);
fclose(s);
delete(s);
TF = isempty(b);
if TF==1
    b=b-1
end
display

% --- Executes on button press in HammingDistMatch.
function HammingDistMatch_Callback(hObject, eventdata, handles)
% hObject    handle to HammingDistMatch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%IdentifyGUI=getappdata(0,'IdentifyGUI');
%cropimg=getappdata(IdentifyGUI,'crpimg');

close all
delete all
closepreview
identification
```