BIOMETRIC IDENTIFICATION SYSTEM

Submitted in partial fulfilment of the Degree of
Bachelor of Technology

MAY-2012

AKSHAY SHARMA (081100)

ANIRUDH SINGHAL (081105)

NAME OF SUPERVISOR: Prof. TAPAN KUMAR JAIN

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT (H.P.)

CERTIFICATE

This is to certify that the thesis entitled, "**BIOMETRIC IDENTIFICATION SYSTEM**", submitted by **Akshay Sharma** and **Anirudh Singhal** in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: 01.06.2012

Prof. Tapan Kumar Jain

Place: Waknaghat

Department of Electronics and Communication Engineering

Jaypee University of Information Technology

ACKNOWLEDGEMENT

Firstly, we would like to thank God for his grace and our parents for their motivation all through this work.

We express our gratitude and sincere thanks to Prof. Tapan Kumar Jain, Department Of Electronics And Communication Engineering, for providing the oppotunity to undertake the project under his able guidance. His directions and unparalleled support for fetching solutions from the industry helped us immensely in realization of the project.

The zeal to accomplish the task of formulating the project could not have been realized without the support and cooperation of the faculty members of ECE department. We sincerely thank Prof. Dr. T.S. Lamba (Dean A&R) and Prof. Dr. Sunil V. Bhooshan (HOD, ECE) for their consistent support throughout the project work. We are also grateful to Mr. Mohan (Project Lab Engineer, ECE) for his practical guidance and continuous help regarding software details and resources.

Date: 1st June, 2012

Akshay Sharma

Akshay Sharma (081100)

Anirudh Singhal

Anirudh Singhal (081105)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Our project aims at introducing biometric capable technologies, which can be used in various applications like- voting mechanism, attendance system, etc. The goal can be disintegrated into finer sub-targets; fingerprint capture, fingerprint image processing, minutiae extraction and matching. For each sub-tasks various methods from literature are analysed. From the study of entire process, an integrated approach is proposed.

Biometrics based technologies are supposed to be very efficient personal identifiers as they can keep track of characteristics believed to be unique in each person. Among these technologies, fingerprint recognition is universally applied. It extracts minutiae based features from the scanned images of the fingerprint made by different ridges on the fingertips.

Accuracy and reliability are the two most important parameters when it comes to biometric applications. Fingerprint verification is one of the oldest known biometric techniques but still it is the most widely used because of its simplicity and good levels of accuracy. It's a well known fact that every human being is born with different pattern on the fingers and this feature is exploited to identify and differentiate between two different persons. This is the reason that we have chosen these characteristics of human body to create a system which could help in introducing a security feature in the era of growing technology.

CHAPTER 1: INTRODUCTION

1.1 Why Biometrics?

Securing personal privacy and deterring identity theft are national priorities. These goals are essential to our democracy and our economy, and inherently important to our citizens. Moreover, failure to achieve these goals is substantially inhibiting the growth of our most advanced, leading-edge industries, notably including e-commerce, that depend upon the integrity of network transactions. Establishing end-to-end trust among all parties to network transactions is the indispensable basis for success. A large percentage of the public are reluctant to engage in e-commerce or conduct other network transactions owing to a well-founded lack of confidence that the system will protect their privacy and prevent their identity from being stolen and misused. The misgivings of the public are reinforced by recent publicized cases of loss of personal privacy, fraudulent funds transfers, and outright theft and abuse of identity in network transactions.

Biometrics, an emerging set of technologies, promises an effective solution. Biometrics accurately identifies or verifies individuals based upon each person's unique physical or behavioural characteristics. Biometrics work by unobtrusively matching patterns of live individuals in real time against enrolled records. Leading examples are biometric technologies that recognize and authenticate faces, hands, fingers, signatures, irises, voices, and fingerprints.

Biometric data are separate and distinct from personal information. Biometric templates cannot be reverse-engineered to recreate personal information and they cannot be stolen and used to access personal information. Precisely because of these inherent attributes, biometrics are effective means to secure privacy and deter identity theft.

One of the world's largest fingerprint recognition systems is the Integrated Automated Fingerprint Identification System, maintained by the FBI in the US since 1999. The IAFIS currently contains fingerprints of more than 60 million persons, with corresponding demographic information, providing both latent-print search for crime scene investigation and 10-print ID for suspect identification and general-population background checks.

The main reasons for the popularity of fingerprint recognition are

- Its success in various applications in the forensic, government, and civilian domains;
- The fact that criminals often leave their fingerprints at crime scenes;
- The existence of large legacy databases; and
- The availability of compact and relatively inexpensive fingerprint readers.

1.2 Why this project?

The application in an education institute is worth nothing because of the benefits it brings along with it. The fingerprint recognition and verification technique can easily replace an attendance sheet in offices. A finger print detecting device needs to be placed in each office and person would be making to swipe their finger over the sensor so as to mark their presence in the office. The database would contain the entire fingerprint beforehand. So, the moment a finger is swiped, a check would be carried out with the exiting database and the corresponding person would get a present mark on his attendance record maintain in a server.

The transfer of the finger print from the device to the computer can be carried out using the software development kit provided with the fingerprint reader. For further security o the entire system and to detect the illegal activities, a security camera can be installed to keep track of the enrolments made in the offices.

1.3 System Design

The design of fingerprint based identification system can be divided into the 4 different modules. They are:

- Fingerprint Scanning
- Image Processing
- Minutiae Extraction
- Fingerprint Matching

Fingerprint scanning is done using an image scanner, whereas the image processing, minutiae extraction and matching are done on PC using MATLAB.

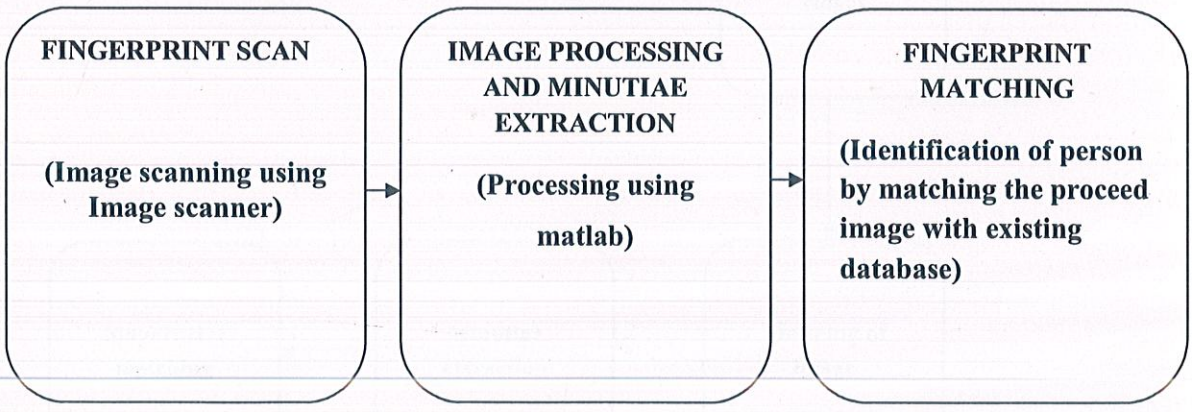


Figure 1.1 A block diagram depicting the working of the project

The module- wise approach to the design of the system helps in better understanding of the individual function levels. Also, a parallel approach to the system helps in distributing the effort on a multi-level range and helps in identifying the best features and available products in the market that suit the design requirements. This has been done in the following chapters.

1.4 Plan of action

The following block diagram shows our plan of action of the implementation of project.

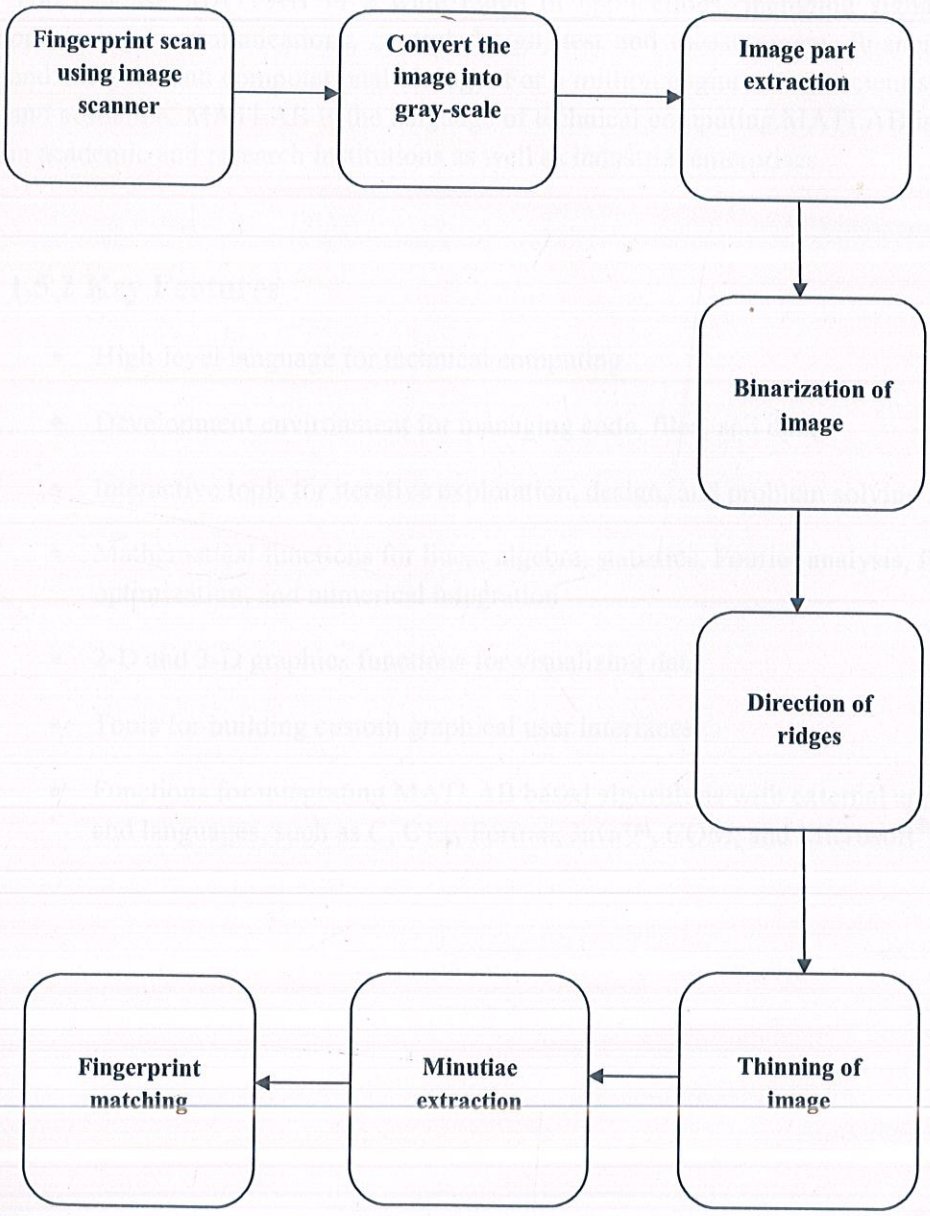


Figure 1.2 Step-wise procedure of project

1.5 Introduction to *MATLAB*

1.5.1 Introduction

MATLAB is a programming environment for algorithm development, data analysis, visualization, and numerical computation. Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and FORTRAN.

You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modelling and analysis, and computational biology. For a million engineers and scientists in industry and academia, MATLAB is the language of technical computing.MATLAB is widely used in academic and research institutions as well as industrial enterprises.

1.5.2 Key Features

- High-level language for technical computing.
- Development environment for managing code, files, and data
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- 2-D and 3-D graphics functions for visualizing data
- Tools for building custom graphical user interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java™, COM, and Microsoft® Excel

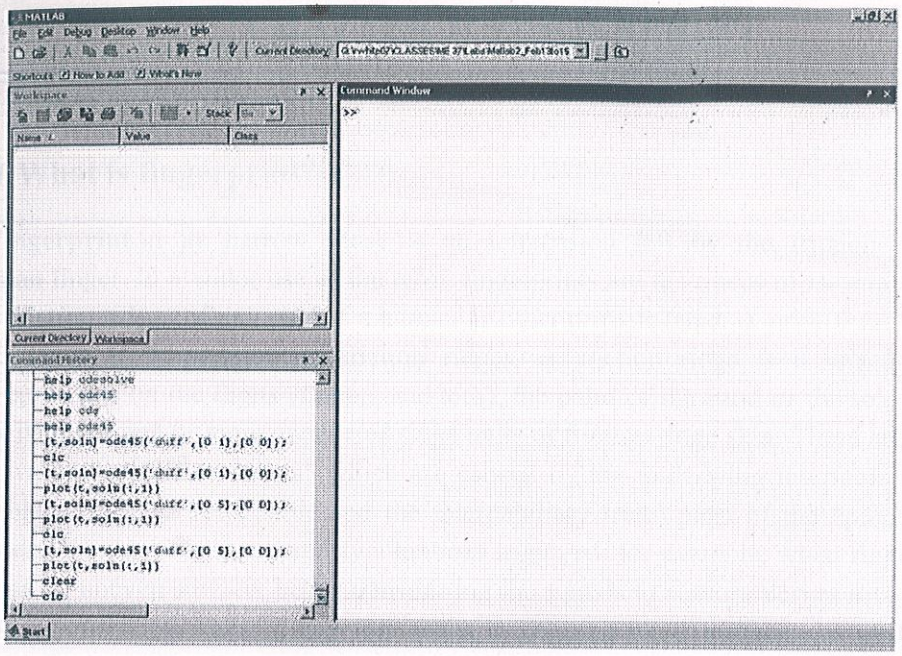


Figure 1.3: MATLAB command window

CHAPTER 2: FINGERPRINT

2.1 What is fingerprint?

A **fingerprint** in its narrow sense is an impression left by the friction ridges of a human finger. In a wider use of the term, fingerprints are the traces of an impression from the friction ridges of any part of a human or other primate hand. A print from the foot can also leave an impression of friction ridges. A friction ridge is a raised portion of the epidermis on the digits (fingers and toes), the palm of the hand or the sole of the foot, consisting of one or more connected ridge units of friction ridge skin. These are sometimes known as "epidermal ridges" which are caused by the underlying interface between the dermal papillae of the dermis and the interpapillary (rete) pegs of the epidermis. These epidermal ridges serve to amplify vibrations triggered, for example, when fingertips brush across an uneven surface, better transmitting the signals to sensory nerves involved in fine texture perception. These ridges also assist in gripping rough surfaces, as well as smooth wet surfaces.

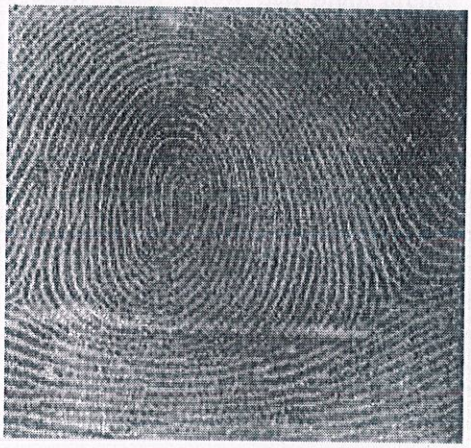


Figure 2.1: Image of fingerprint

2.2 Fingerprint Basics

There are three main fingerprint patterns:

- Arches
- Loops
- Whorls

2.2.1 Arches

Arches are found in about 5% of fingerprint patterns encountered. The ridges run from one side to the other of the pattern, making no backward turn. Ordinarily, there is no delta in an arch pattern but where there a delta; no re-curving ridge must intervene between the core and delta points. There are four types of arch patterns:-

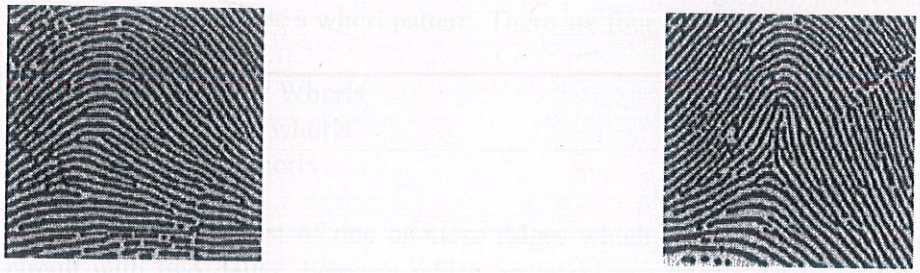
- Plain Arches
- Radial Arches
- Ulnar Arches
- Tented Arches

Plain arches have an even flow of ridges from one side to the other of the pattern; no "significant up thrusts" and the ridges enter on one side of the impression, and flow out the other with a rise or wave in the center.

The ridges of **Radial arches** slope towards the thumb, have one delta and no re-curving ridges.

On **Ulnar arches**, the ridges slope towards the little finger, have one delta and no re-curving ridges.

Tented arches have an angle, an up thrust, or two of the three basic characteristics of the loop. They don't have the same "easy" flow that plain arches do and particularly have "significant up thrusts" in the ridges near the middle that arrange themselves on both sides of a spine or axis towards which the adjoining ridges converge and appear to form tents.



(a) (b)

Figure 2.2: (a) Plain Arches (b) Tented Arches

2.2.2 Loops

Loops occur in about 60-70 % of fingerprint patterns encountered. One or more of the ridges enters on either side of the impression, re-curves, touches or crosses the line running from the delta to the core and terminates on or in the direction of the side where the ridge or ridges entered. Each loop pattern has is one delta and one core and has a ridge count. There are two types of loops patterns:-

- Radial Loops

- Ulnar Loops

Radial loops are named after the radius, a bone in the forearm that joins the hand on the same side as the thumb. The flow of the pattern in radial loops runs in the direction of the radius (toward the thumb). Radial loops are not very common and most of the time radial loops will be found on the index fingers

Ulnar loops are named after the ulna, a bone in the forearm. The ulna is on the same side as the little finger and the flow of the pattern in a ulnar loop runs in the direction of the ulna (toward the little finger)

(a) (b)

Figure 2.3: (a) Radial Loops (b) Ulnar Loops

2.2.3 Whorls

Whorls are seen in about 25-35 % of fingerprint patterns encountered. In a whorl, some of the ridges make a turn through at least one circuit. Any fingerprint pattern which contains 2 or more deltas will be a whorl pattern. There are four types of whorl patterns:-

- Plain Whorls
- Central Pocket Whorls
- Double Loops whorls
- Accidental Whorls

Plain whorls consist of one or more ridges which make or tend to make a complete circuit with two deltas, between which an imaginary line is drawn and at least one re-curving ridge within the inner pattern area is cut or touched.

Central pocket loop whorls consist of at least one re-curving ridge or an obstruction at right angles to the line of flow, with two deltas, between which when an imaginary line is drawn, no re-curving ridge within the pattern area is cut or touched. Central pocket loop whorl ridges make one complete circuit which may be spiral, oval, circular or any variant of a circle.

Double loop whorls consist of two separate and distinct loop formations with two separate and distinct shoulders for each core, two deltas and one or more ridges which make, a complete circuit. Between the two at least one re-curving ridge within the inner pattern area is cut or touched when an imaginary line is drawn.

Accidental whorls consist of two different types of patterns with the exception of the plain arch; have two or more deltas or a pattern which possess some of the requirements for two or more different types or a pattern which conforms to none of the definitions.

(a) (b) (c) (d)

Figure 2.4: (a) Plain Whorl (b) Central Pocket Whorl (c) Double Loop Whorl (d) Accidental Whorl

2.3 Ridge Characteristics

A single rolled fingerprint may have as many as 100 or more identification points that can be used for identification purposes. These points are often ridge characteristics.

There are many different ridge characteristics, although some of them are more common than others. These points can be used as points of comparison for fingerprint identification. Depending on how prevalent the ridge characteristics, fewer or more points of comparison may be needed for a positive identification.

2.3.1 Ridge Dots:- An isolated ridge unit whose length approximates its width in size.

Figure 2.5: Ridge Dots in fingerprint image

2.3.2 Bifurcations:- The point at which one friction ridge divides into two friction ridges.

Figure 2.6: Double Bifurcation in fingerprint image

Figure 2.7: Opposed Bifurcation in fingerprint image

2.3.3 Trifurcations: - The point at which one friction ridge divides into three friction ridges.

Figure 2.8: Trifurcations in fingerprint image

2.3.4 Ending Ridge: -A single friction ridge that terminates within the friction ridge structure.

Figure 2.9: Ending Ridge in fingerprint image

2.3.5 Ridge Crossing:- A point where two ridge units intersect.

Figure 2.10: Ridge Crossing in fingerprint image

2.3.6 Enclosures (Lakes):- A single friction ridge that bifurcates and rejoins after a short course and continues as a single friction ridge.

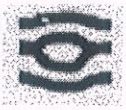


Figure 2.11: Enclosures in fingerprint image

2.3.7 Short Ridges (Islands):- Friction ridges of varying lengths.

Figure 2.12: Short Ridges in fingerprint image

2.3.8 Spurs (Hooks):- A bifurcation with one short ridge branching off a longer ridge.

Figure 2.13: Spurs in fingerprint image

2.3.9 Bridges:- A connecting friction ridge between parallel running ridges, generally right angles.

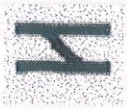


Figure 2.14: Bridge in fingerprint image

2.4 Fingerprint Recognition

Once the fingerprint is captured, the next step is the recognition procedure. The recognition procedure can be broadly sub grouped into

a. Fingerprint identification
b. Fingerprint verification

Fingerprint identification refers to specify one's identity based on his fingerprints. The fingerprints are captured without any information about the identity of the person. It is then matched across a database containing numerous fingerprints. The identity is only retrieved when a match is found with one existing in the database. So, this is a case of one-to-n matching where one capture is compared to several others. This is widely used for criminal cases.

Fingerprint verification is different from identification in a way that the persons' identity is stored along with the fingerprint in a database. On enrolling the fingerprint, the real time capture will be retrieved back the identity of the person. This is however a one-to-one matching. This is used in offices like passport offices etc. Where the identity of a person has to be checked with the one provided at a previous state.

2.5 Minutiae Extraction

In the biometric process of finger scanning, minutiae are specific points in a finger image. A ridge is defined as a single curve segment in fingerprint and a valley is the region of between two adjacent ridges.

Minutiae, which are the local discontinuous in the ridge flow pattern, provided the details of ridge-valley structure in fingerprint. There are 50 to 150 minutiae on single fingerprint image.

The number and locations of the minutiae vary from finger to finger in any particular person, and from person to person for any particular finger (for example, the thumb on the left hand).

The major minutiae features of fingerprint ridges are: ridge ending, bifurcation, and short ridge (or dot). The ridge ending is the point at which a ridge terminates. Bifurcations are points at which a single ridge splits into two ridges. Short ridges (or dots) are ridges which are significantly shorter than the average ridge length on the fingerprint.

CHAPTER 3: FINGERPRINT IMAGE PROCESSING

After the fingerprint scanning from image scanner we did image processing on it. We have used following image processing techniques:-

- Histogram Equalization
- Fast Fourier Transform
- Binarization of Image
- Thinning of Image

3.1 Histogram Equalization

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.
The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In histogram equalization we are trying to maximize the image contrast by applying a gray level transform which tries to flatten the resulting histogram. It turns out that the gray level transform that we are seeking is simply a scaled version of the original image's cumulative histogram.

(a) (b)

Figure 3.1: (a) An unequalized image (b) The same image after histogram equalization

Original image **Enhanced image**

Figure 3.2: Effect of Histogram Equalization

3.2 Fast Fourier Transform

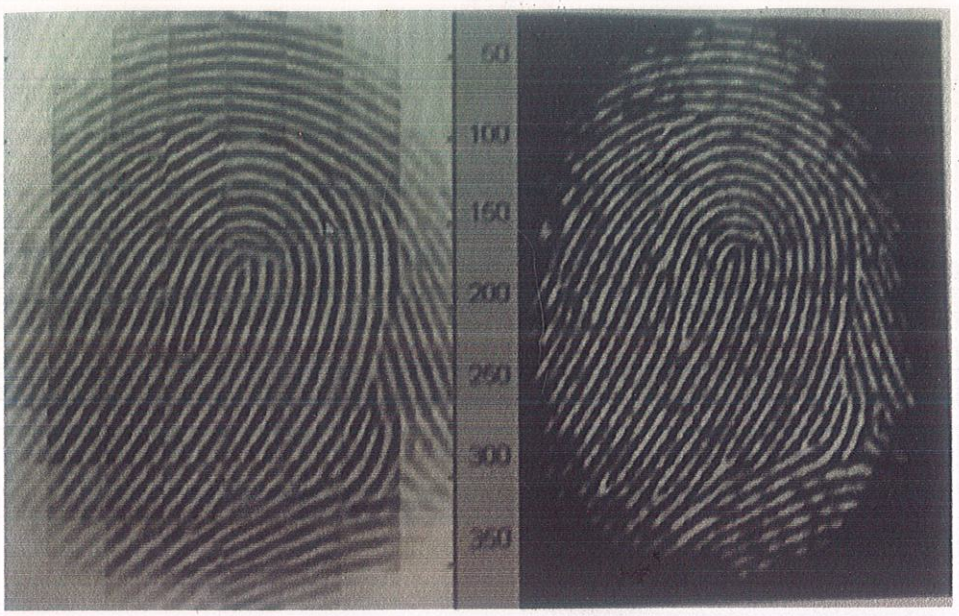
Fast Fourier Transform (FFT) is an efficient implementation of DFT and is used, apart from other fields, in digital image processing. Fast Fourier Transform is applied to convert an image from the image (spatial) domain to the frequency domain. Applying filters to images in frequency domain is computationally faster than to do the same in the image domain.

Fourier Transform decomposes an image into its real and imaginary components which is a representation of the image in the frequency domain. If the input signal is an image then the number of frequencies in the frequency domain is equal to the number of pixels in the image or spatial domain. The inverse transform re-transforms the frequencies to the image in the spatial domain. The FFT and its inverse of a 2D image are given by the following equations:

$$F(x) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi n/N} \qquad (3.1)$$

$$f(n) = 1/N \sum_{n=0}^{N-1} F(x) e^{j2\pi n/N} \qquad (3.2)$$

Where f(m,n) is the pixel at coordinates (m, n), F(x,y) is the value of the image in the frequency domain corresponding to the coordinates x and y, M and N are the dimensions of the image.



Original Image **After FFT**

Figure 3.3: Depicts FFT Image processing

The image after FFT connects falsely broken point on the ridges and removes spurious connections in between the ridges.

3.3 Image Binarization

A **binary image** is a digital image that has only two possible values for each pixel. Typically the two colours used for a binary image are black and white though any two colours can be used. The colour used for the object(s) in the image is the foreground colour while the rest of the image is the background colour. In the document scanning industry this is often referred to as bi-tonal.

Image binarization converts an image of up to 256 gray levels to a black and white image. The simplest way to use image binarization is to choose a threshold value, and classify all pixels with values above this threshold as white, and all other pixels as black.

The original image is a 8-bit greyscale image. This process transforms the original image into a 1-bit image. Binarization changes the pixel value to 1 if he value is found to exceed the mean intensity of the current block to which it belongs.

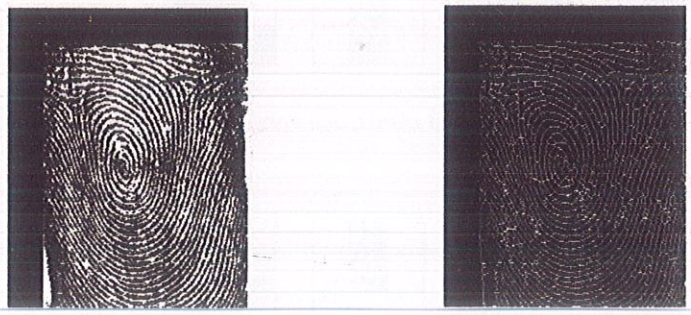
The figure clearly depicts the effect of binarization on a normal greyscale image that has been only enhanced.

Binarized Image **Grey Scale**

Figure 3.4: Effect of Binarization

3.4 Thinning Of Images

In image processing, Image thinning is a fundamental pre-processing. It reduces a large amount of memory usage for structural information storage. Binary digital image can be represented by a matrix, where each element in matrix is either zero (white) or one (black) and the points are called pixels. Thinning is a process that deletes the unwanted pixels and transforms the image pattern one pixel thick. The thinning operation halts when no more pixels can be removed from the image. Thinning is a morphological operation that is used to remove selected foreground pixels from binary images.



Binarized Image **After thinning**

Figure 3.5: Thinning process

CHAPTER 4: MINUTIAE EXTRACTION

An accurate representation of the fingerprint image is critical to automatic fingerprint identification systems, because most deployed commercial large-scale systems are dependent on feature-based matching (correlation based techniques have problems as discussed in the previous section). Among all the fingerprint features, minutia point features with corresponding orientation maps are unique enough to discriminate amongst fingerprints robustly; the minutiae feature representation reduces the complex fingerprint recognition problem to a point pattern matching problem. In order to achieve high-accuracy minutiae with varied quality fingerprint images, segmentation algorithm needs to separate foreground from noisy background which includes all ridge-valley regions and not the background. Image enhancement algorithm needs to keep the original ridge flow pattern without altering the singularity, join broken ridges, clean artifacts between pseudo-parallel ridges, and not introduce false information. Finally minutiae detection algorithm needs to locate efficiently and accurately the minutiae points.
Minutia points are detected by locating the end points and bifurcation points on the thinned ridge skeleton based on the number of neighbouring pixels. The end points are selected if they have a single neighbour and the bifurcation points are selected if they have more than two neighbours. However, methods based on thinning are sensitive to noise and the skeleton structure does not conform to intuitive expectation.

From the binary thinned image, the minutiae are detected by using 3x3 pattern masks. Samples of masks used for identifying the ridge ending and bifurcations point are shown in the figure below. Although the process seems to be simple, it is necessary to consider the elimination of false detected minutiae.

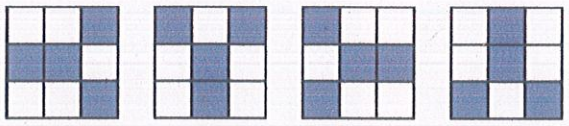


Figure 4.1: Masks for ridge bifurcation

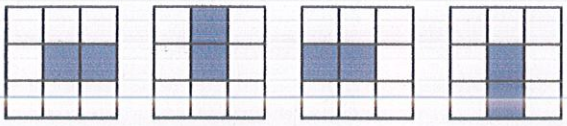


Figure 4.2: Masks for ridge endings

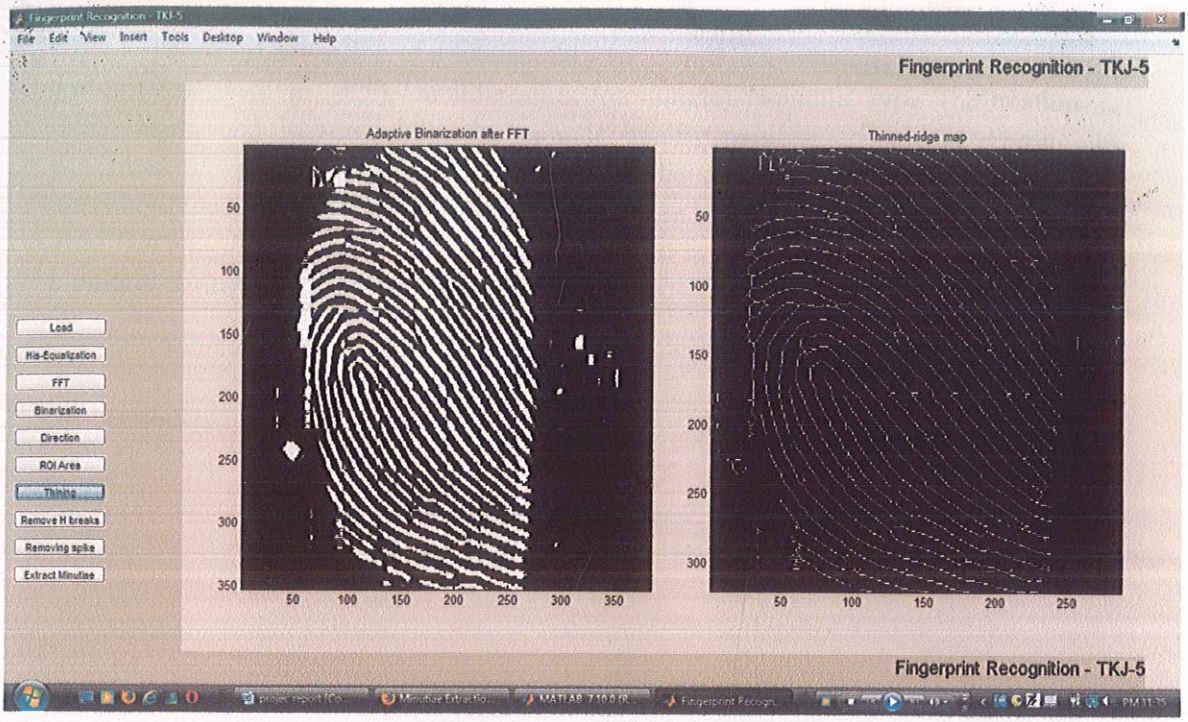


Figure 4.3: Binarization and thinning of fingerprint image

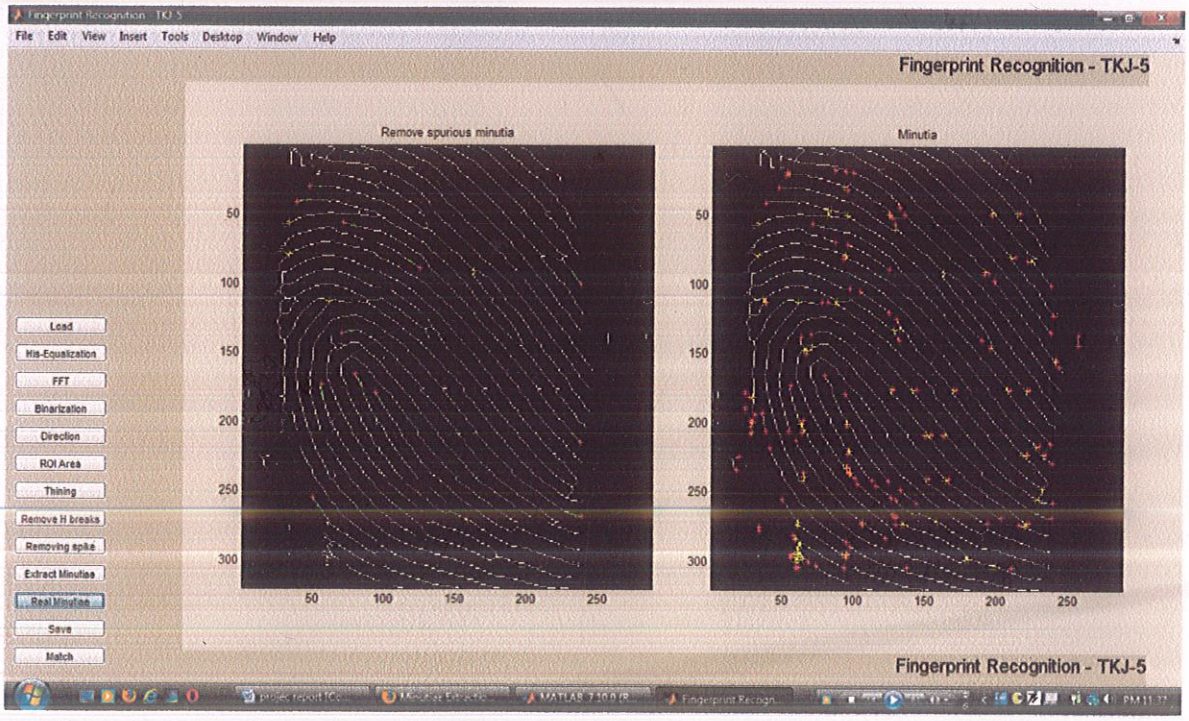


Figure 4.4: Figure shows the marked minutiae points and finally selected minutiae points for matching

After a successful extraction of minutiae, they are stored in a template, which may contain the minutia position (x,y), minutia direction (angle), minutia type (bifurcation or termination), and in some case the minutia quality may be considered. During the enrolment the extracted template are stored in the database and will be used in the matching process as **reference template** or database template. During the verification or identification, the extracted minutiae are also stored in a template and are used as **query template** during the matching.

Chapter 5: FINGERPRINT MATCHING

5.1 Introduction

Fingerprint matching has been successfully used by law enforcement for more than a century. The technology is now finding many other applications such as identity management and access control. The pattern of friction ridges on each finger is unique and immutable, enabling its use as a mark of identity. In fact, even identical twins can be differentiated based on their fingerprints. Superficial injuries such as cuts and bruises on the finger surface alter the pattern in the damaged region only temporarily; the ridge structure reappears after the injury heals Fingerprint matching techniques can be placed into two categories: minutiae-based and correlation based. Minutiae-based techniques first find minutiae points and then map their relative placement on the finger. Correlation-based techniques require the precise location of a registration point and are affected by image translation and rotation.

A fingerprint matching module computes a match score between two fingerprints, which should be high for fingerprints from the same finger and low for those from different fingers. Fingerprint matching is a difficult pattern-recognition problem due to large interclass variations (variations in fingerprint images of the same finger) and large interclass similarity (similarity between fingerprint images from different fingers). Interclass variations are caused by finger pressure and placement—rotation, translation, and contact area—with respect to the sensor and condition of the finger such as skin dryness and cuts. Meanwhile, interclass similarity can be large because there are only three types of major fingerprint patterns (arch, loop, and whorl).
Most fingerprint-matching algorithms adopt one of four approaches: image correlation, phase matching, skeleton matching, and minutiae matching. Minutiae-based representation is commonly used, primarily because

- forensic examiners have successfully relied on minutiae to match fingerprints for more than a century,
- minutiae-based representation is storage efficient, and
- expert testimony about suspect identity based on mated minutiae is admissible in courts of law.

The current trend in minutiae matching is to use local minutiae structures to quickly find a coarse alignment between two fingerprints and then consolidate the local matching results at a global level.

5.2 Matching Techniques

Approaches to fingerprint matching can be coarsely classified into three families:

- Correlation-based matching
- Minutiae-based matching
- Non-minutiae feature-based matching

Correlation-based matching: In order to match two fingerprint using correlation based technique, the fingerprint are aligned and computed the correlation for each corresponding pixel, however as the displacement and rotation are unknown it is necessary to apply the correlation for all possible alignments. The singularity information may be useful in order to find an approximated alignment.

Minutiae-based matching: this is the most popular and widely used technique, being the basis of the fingerprint comparison made by the fingerprint examiners. Minutiae are extracted from the two fingerprints and stored as the set of point in two dimension plane. Minutiae-based matching essentially consists of finding the alignment between the template and the input minutiae feature sets that result in the maximum number of minutiae pairing.

Non-minutiae feature-based matching: It uses features other than characteristics of minutiae from the fingerprint ridge pattern, such as local orientation and frequency, ridge shape, and texture information. It can extract more rich discriminatory information and abandon the pre-processing process such as binarization and thinning and post minutiae processing.

5.3 Matching Algorithm used in the project

In minutiae based fingerprint algorithm, First, the algorithm computes pair wise similarity between minutiae of two fingerprints by comparing minutiae *descriptors* that are invariant to rotation and translation. Next, it aligns two fingerprints according to the most similar minutiae pair. The algorithm then establishes minutiae correspondence—minutiae that are close enough both in location and direction are deemed to be corresponding (mated) minutiae. Finally, the algorithm computes a similarity score to reflect the degree of match between two fingerprints based on factors such as the number of matching minutiae, the percentage of matching minutiae in the overlapping area of two fingerprints, and the consistency of ridge count between matching minutiae.

We have used Fingerprint Minutia Matcher Based on Ridge Alignment, which accepts the two template files and return the maximum similar certainty of the two fingerprints based on x-position of minutiae, y-position of minutiae and orientation of minutiae.

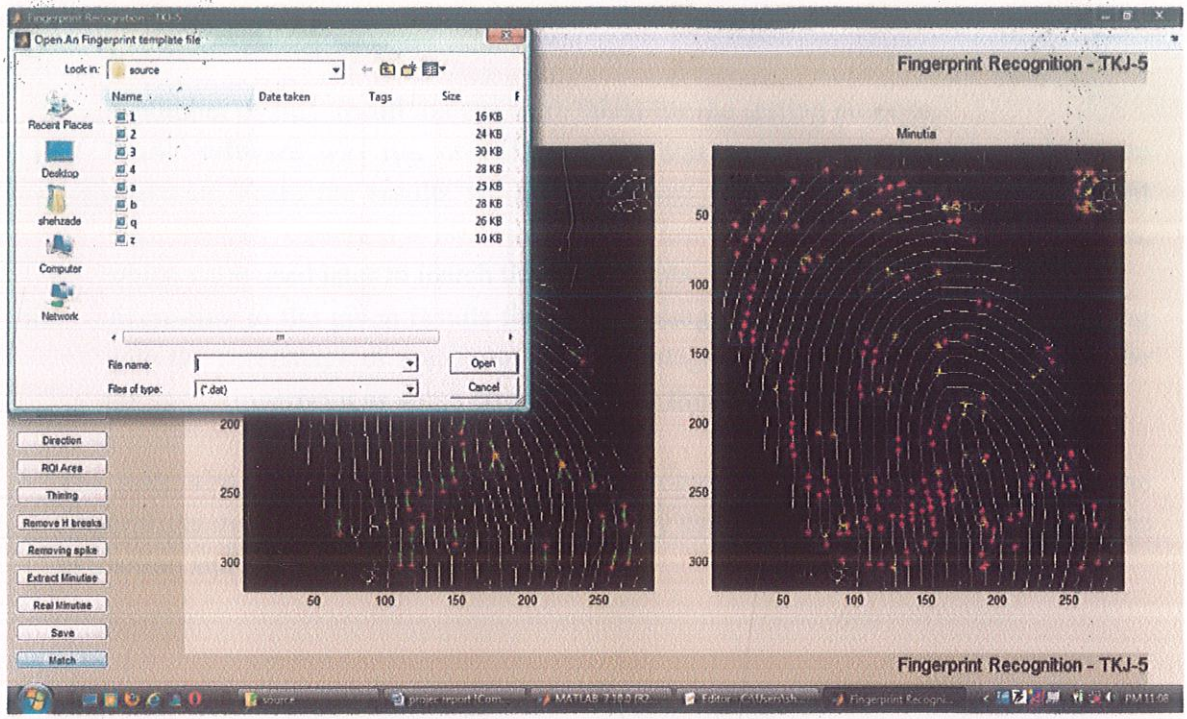


Figure 5.1: Selecting the fingerprints to be matched

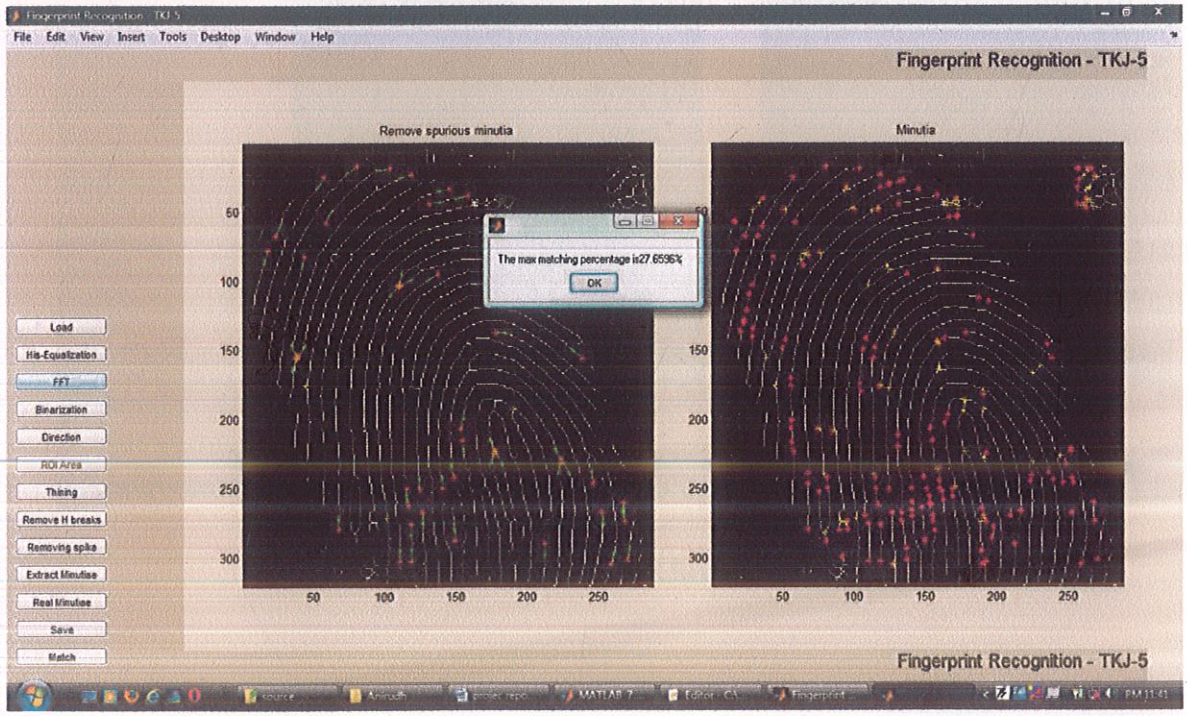


Figure 5.2: Prompt showing the matching percentage of two images

Chapter 6: CONCLUSION

i. A number of fingerprint images were taken for the testing purpose.

ii. Demo software was run on different fingerprint images and their results were analyzed. From the results we could analyze that for different images different *.dat files were created to store the information of the extracted minutiae positions, which were used later to match the fingerprints.

iii. According to the given results the final message is displayed in the prompt along with the percentage of matching. For percentage greater than 80% it is taken to be the same fingerprint as some errors might be there while fingerprint scanning.

The main aim of the project was to create a program that could efficiently match the fingerprints overcoming the various problems by using an efficient method of minutiae extraction and matching.

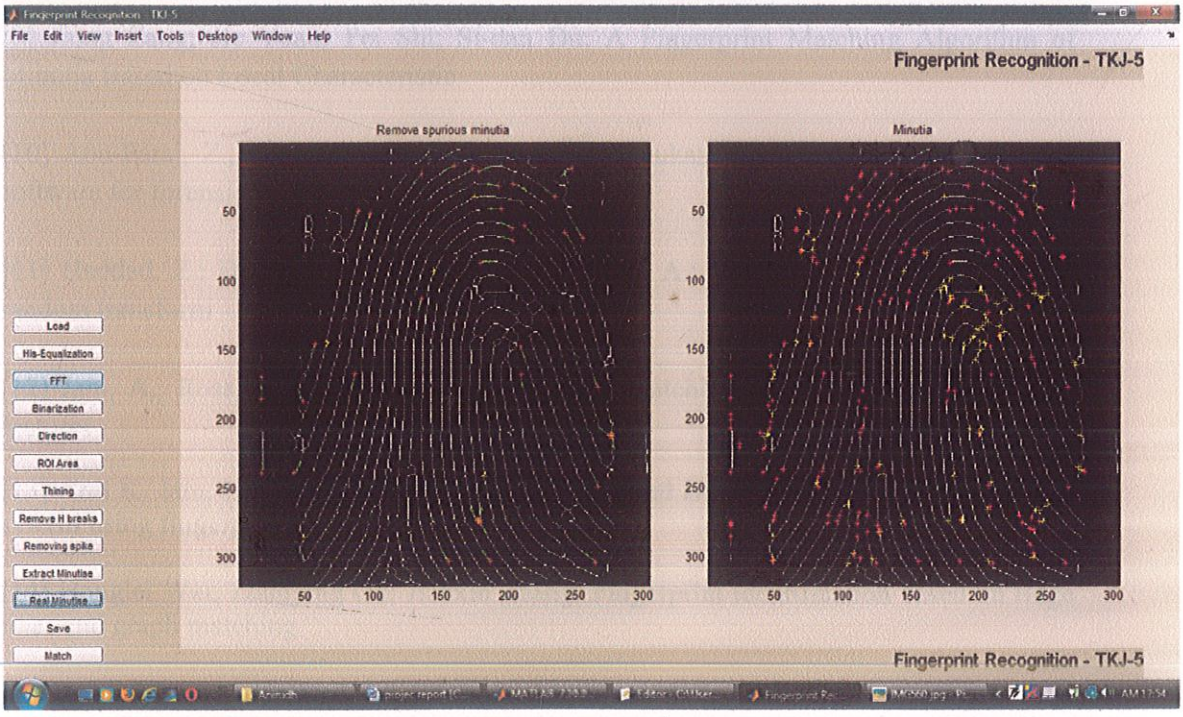


Figure 6.1: Fingerprint Recognition

BIBLIOGRAPHY

[1] Engr. Imran Anwar Ujan and Dr. Imdad Ali Ismaili : Biometric Attendance System

[2] Víctor López Lorenzo, Pablo Huerta Pellitero, José Ignacio Martínez Torre, Javier Castillo Villar : Fingerprint Minutiae Extraction Based On FPGA and MatLab

[3] Ravi. J, K. B. Raja, Venugopal. K. R : Fingerprint Recognition using Minutiae Score Matching

[4] Neeta Nain, Deepak B M, Dinesh Kumar, Biju Gautham, and Manisha Baswal : Minutiae–Based Fingerprint Matching

[5] Joshua Abraham, Paul Kwan and Junbin Gao : Fingerprint Matching using A Hybrid Shape and Orientation Descriptor

[6] Rabia Bakhteri, Mohamed Khalil Hani: Biometric Encryption using Fingerprint Fuzzy Vault for FPGA-based Embedded Systems

[7] Handbook of fingerprint recognition – Salil Prabhakar, Anil k jain

[8] Anil K. Jain, Jianjiang Feng, Karthik Nandakumar : Fingerprint Matching

[9] Jiong Zang; Jie Yuan; Fei Shi; Si-dan Du: A Fingerprint Matching Algorithm of Minutia Based on Local Characteristic

[10] Abu-Faraj, Z.; Abdallah, A.; Chebaklo, K.; Khoukaz, E.: Fingerprint identification software for forensic applications

[11] Haddad, Z.; Beghdadi, A.; Serir, A.; Mokraoui, A.: Fingerprint idntification using random transform

[12] Jain, A.; Ross, A.; Prabhakar, S.: Fingerprint matching using minutiae and texture features

[13] Anil K. Jain, Lin Hong, Sharat Pankanti, and Ruud Bolle: An identity authentication system using fingerprints

[14] Honglei Wei; Zongying Ou; Jianxin Zhang: Fingerprint identification based on ridge lines and graph matching

[15] www.fingerprinting.com

APPENDIX A : SOURCE CODE

1. For GUI

```
%start_gui_single_mode.m
%Script file to generate the GUI
%tkj-5

clear
FigWin = figure('Position',[50 -50 650 500],...
   'Name','Fingerprint Recognition - TKJ-5',...
   'NumberTitle','off',...
   'Color',[ 0.827450980392157 0.815686274509804
0.776470588235294 ]);

AxesHandle1 = axes('Position',[0.2 0.15 0.35 0.7],...
   'Box','on');
AxesHandle2 = axes('Position',[0.6 0.15 0.35 0.7],...
   'Box','on');

BackColor = get(gcf,'Color');
%[ 0.827450980392157 0.815686274509804 0.776470588235294 ]

%[ 0.741176470588235 0.725490196078431 0.658823529411765 ]

FrameBox = uicontrol(FigWin,...
   'Units','normalized', ...
   'Style','frame',...
   'BackgroundColor',[ 0.741176470588235 0.725490196078431
0.658823529411765 ],...
   'ForegroundColor',[ 0.741176470588235 0.725490196078431
0.658823529411765 ],...
    'Position',[0 0 0.15 1]);

%create static text.
Text2 = uicontrol(FigWin,...
   'Style','text',...
   'Units','normalized', ...
   'Position',[0 0.95 .97 0.05],...
   'FontSize',15,...
   'BackgroundColor',[ 0.741176470588235 0.725490196078431
0.658823529411765 ],...
   'HorizontalAlignment','right', ...
   'String','Fingerprint Recognition - TKJ-5');


Text2 = uicontrol(FigWin,...
   'Style','text',...
```

```
    'Units','normalized', ...
    'Position',[0 0 .97 0.05],...
    'FontSize',15,...
    'BackgroundColor',[ 0.741176470588235 0.725490196078431
0.658823529411765 ],...
    'HorizontalAlignment','right', ...
    'String','Fingerprint Recognition - TKJ-5');

w=16;
textLoad='Load Fingerprint Image';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
    'Position',[10,380,100,20],...
    'String','Load',...
    'Callback',...
    ['image1=loadimage;'...
     'subplot(AxesHandle1);'...
     'imagesc(image1);'...
     'title(textLoad);'...
      'colormap(gray);']);

text_filterArea='Orientation Flow Estimate';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
     'Position',[10,260,100,20],...
     'String','Direction',...
     'Callback',...

['subplot(AxesHandle2);[o1Bound,o1Area]=direction(image1,16)
;title(text_filterArea);']);

text_ROI='Region Of Interest(ROI)';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
     'Position',[10,230,100,20],...
     'String','ROI Area',...
     'Callback',...

['subplot(AxesHandle2);[o2,o1Bound,o1Area]=drawROI(image1,o1
Bound,o1Area);title(text_ROI);']);

text_eq='Enhancement by histogram Equalization';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
    'Position',[10,350,100,20],...
    'String','His-Equalization',...
    'Callback',...

['subplot(AxesHandle2);image1=histeq(uint8(image1));imagesc(
image1);title(text_eq);']);
```

```
text21='Adaptive Binarization after FFT';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
    'Position',[10,290,100,20],...
    'String','Binarization',...
    'Callback',...
    [%'W=inputdlg(text);W=str2num(char(W));'...
       'subplot(AxesHandle1);'...

'image1=adaptiveThres(double(image1),32);title(text21);']);

text='Please input the FFT factor(0~1)';
text_fft='Enhancement by FFT';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
    'Position',[10,320,100,20],...
    'String','FFT',...
    'Callback',...
    ['W=inputdlg(text);W=str2double(char(W));'...

'subplot(AxesHandle1);image1=fftenhance(image1,W);imagesc(im
age1);title(text_fft);']);

text31='Thinned-ridge map';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
    'Position',[10,200,100,20],...
    'String','Thining',...
    'Callback',...

['subplot(AxesHandle2);o1=im2double(bwmorph(o2,''thin'',Inf)
);imagesc(o1,[0,1]);title(text31);']);

text41='Remove H breaks';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
    'Position',[10,170,100,20],...
    'String','Remove H breaks',...
     'Callback',...

['subplot(AxesHandle2);o1=im2double(bwmorph(o1,''clean''));o
1=im2double(bwmorph(o1,''hbreak''));imagesc(o1,[0,1]);title(
text41);']);


textn1='remove spike';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
    'Position',[10,140,100,20],...
    'String','Removing spike',...
    'Callback',...
```

```
['subplot(AxesHandle2);o1=im2double(bwmorph(o1,''spur''));im
agesc(o1,[0,1]);title(textn1);']);

%% locate minutia and show all those minutia
text51='Minutia';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
     'Position',[10,110,100,20],...
     'String','Extract Minutiae',...
     'Callback',...

['[end_list1,branch_list1,ridgeMap1,edgeWidth]=mark_minutia(
o1,o1Bound,o1Area,w);'...

'subplot(AxesHandle2);show_minutia(o1,end_list1,branch_list1
);title(text51);']);

%Process for removing spurious minutia
text61='Remove spurious minutia';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
     'Position',[10,80,100,20],...
     'String','Real Minutiae',...
    'Callback',...

['[pathMap1,real_end1,real_branch1]=remove_spurious_Minutia(
o1,end_list1,branch_list1,o1Area,ridgeMap1,edgeWidth);'...

'subplot(AxesHandle1);show_minutia(o1,real_end1,real_branch1
);title(text61);']);

%save template file, including the minutia
position,direction,and ridge information
textSaveName='file name';
h=uicontrol(FigWin,...
    'Style','pushbutton',...
     'Position',[10,50,100,20],...
     'String','Save',...
    'Callback',...
    ['W=inputdlg(textSaveName);W=char(W);'...
     'save(W,''real_end1'',''pathMap1'',''-ASCII'');']);

%invoke template file loader and do matching
h=uicontrol('Style','pushbutton',...
    'String','Match',...
     'Position',[10,20,100,20],...
     'Callback',...
    ['finger1=fingerTemplateRead;'...
     'finger2=fingerTemplateRead;'...
     'percent_match=match_end(finger1,finger2,10);']);
```

2. For Image Loading

```
function image1=loadimage
% dialog for opening fingerprint files

[imagefile1, pathname]=
uigetfile('*.bmp;*.BMP;*.tif;*.TIF;*.jpg','Open An
Fingerprint image');
if imagefile1 ~= 0
w = strcat(pathname,imagefile1);
a=imread(w);
image1 = double(a);
image1=255-double(image1);
end
figure(1)
imshow(image1)
```

3. For Getting Direction Of Fingerprint Image

```
function [p,z] = direction(image,blocksize,noShow)
%   DIRECTION cacluates the local flow orientation in each
local window
%  with size (blocksize x blocksize)
%
direction(grayScaleFingerprintImage,blocksize,graphicalShowD
isableFlag)
%  return p ROI bound
%  return z ROI area
%  tkj-5

%image=adaptiveThres(image,16,0);

[w,h] = size(image);
direct = zeros(w,h);
gradient_times_value = zeros(w,h);
gradient_sq_minus_value = zeros(w,h);
gradient_for_bg_under = zeros(w,h);

W = blocksize;
theta = 0;
sum_value = 1;
bg_certainty = 0;
```

```
blockIndex = zeros(ceil(w/W),ceil(h/W));
%directionIndex = zeros(ceil(w/W),ceil(h/W));


times_value = 0;
minus_value = 0;

center = [];

%Note that the image coordinate system is
%x axis towards bottom and y axis towards right

filter_gradient = fspecial('sobel');
%to get x gradient
I_horizontal = filter2(filter_gradient,image);

%to get y gradient
filter_gradient = transpose(filter_gradient);
I_vertical = filter2(filter_gradient,image);


gradient_times_value=I_horizontal.*I_vertical;
gradient_sq_minus_value=(I_vertical-
I_horizontal).*(I_vertical+I_horizontal);
gradient_for_bg_under = (I_horizontal.*I_horizontal) +
(I_vertical.*I_vertical);


for i=1:W:w
    for j=1:W:h
      if j+W-1 < h & i+W-1 < w
            times_value =
sum(sum(gradient_times_value(i:i+W-1, j:j+W-1)));
          minus_value =
sum(sum(gradient_sq_minus_value(i:i+W-1, j:j+W-1)));
          sum_value = sum(sum(gradient_for_bg_under(i:i+W-1,
j:j+W-1)));

          bg_certainty = 0;
          theta = 0;

          if sum_value ~= 0 & times_value ~=0
             %if sum_value ~= 0 & minus_value ~= 0 &
times_value ~= 0
             bg_certainty = (times_value*times_value +
minus_value*minus_value)/(W*W*sum_value);

            if bg_certainty > 0.05
             blockIndex(ceil(i/W),ceil(j/W)) = 1;

             %tan_value = atan2(minus_value,2*times_value);
```

```
                tan_value = atan2(2*times_value,minus_value);


                    theta = (tan_value)/2 ;
                    theta = theta+pi/2;
                %now the theta is within [0,pi]

                %directionIndex(ceil(i/W),ceil(j/W)) = theta;
                %center = [center;[round(i + (W-1)/2),round(j +
(W-1)/2),theta,bg_certainty]];
                 center = [center;[round(i + (W-1)/2),round(j +
(W-1)/2),theta]];
                    end;
            end;
        end;
                 times_value = 0;
              minus_value = 0;
              sum_value = 0;

    end;
end;



if nargin == 2
    imagesc(direct);

    hold on;
    [u,v] = pol2cart(center(:,3),8);
   quiver(center(:,2),center(:,1),u,v,0,'g');
   hold off;
end;


x = bwlabel(blockIndex,4);
%map = [0 0 0;jet(3)];
%figure
%imshow(x+1,map,'notruesize')

y = bwmorph(x,'close');
%figure
%imshow(y,map,'notruesize');

%z is the region of interest (ROI)
%with the index format

z = bwmorph(y,'open');
%figure
%imshow(z,map,'notruesize');

%p is the bound of ROI
```

```
p = bwperim(z);
%figure,
%imshow(p,map,'notruesize');

%directMap = directionIndex;
```

4. For FFT Enhancement

```
function [final]=fftenhance(image,f)
% tkj-5

I = 255-double(image);

[w,h] = size(I);
%out = I;

w1=floor(w/32)*32;
h1=floor(h/32)*32;

inner = zeros(w1,h1);

for i=1:32:w1
   for j=1:32:h1
      a=i+31;
      b=j+31;
      F=fft2( I(i:a,j:b) );
      factor=abs(F).^f;
      block = abs(ifft2(F.*factor));

      larv=max(block(:));
      if larv==0
         larv=1;
      end;

      block= block./larv;
      inner(i:a,j:b) = block;
   end;
end;

%out(1:w1,1:h1)=inner*255;
final=inner*255;

%d=max(out(:)); %Find max pixel value in C.
%c=min(out(:));
%figure
```

```
%imshow(out,[c d]);

final=histeq(uint8(final));
%final=adaptiveThres2(final,32,0);

%imagesc(final);
%colormap(gray);
```

5. For Finding ROI

```
function [roiImg,roiBound,roiArea] =
drawROI(in,inBound,inArea,noShow)
%
drawROI(grayLevelFingerprintImage,ROIboundMap,ROIareaMap,fla
gToDisableGUI)
%  construct a ROI rectangle for the input fingerprint image
and return the
%  segmented fingerprint
%  With the assumption that only one ROI region for each
fingerprint image
%  tkj-5

[iw,ih]=size(in);
tmplate = zeros(iw,ih);
[w,h] = size(inArea);
tmp=zeros(iw,ih);
%ceil(iw/16) should = w
%ceil(ih/16) should = h

left = 1;
right = h;
upper = 1;
bottom = w;

le2ri = sum(inBound);
roiColumn = find(le2ri>0);
left = min(roiColumn);
right = max(roiColumn);

tr_bound = inBound';

up2dw=sum(tr_bound);
roiRow = find(up2dw>0);
upper = min(roiRow);
bottom = max(roiRow);
```

```
%cut out the ROI region image

%show background,bound,innerArea with different gray
intensity:0,100,200

for i = upper:1:bottom
   for j = left:1:right
      if inBound(i,j) == 1
         tmplate(16*i-15:16*i,16*j-15:16*j) = 200;
         tmp(16*i-15:16*i,16*j-15:16*j) = 1;

      elseif inArea(i,j) == 1 & inBound(i,j) ~=1
         tmplate(16*i-15:16*i,16*j-15:16*j) = 100;
         tmp(16*i-15:16*i,16*j-15:16*j) = 1;

      end;
   end;
end;

in=in.*tmp;



roiImg = in(16*upper-15:16*bottom,16*left-15:16*right);

roiBound = inBound(upper:bottom,left:right);
roiArea = inArea(upper:bottom,left:right);



%inner area
roiArea = im2double(roiArea) - im2double(roiBound);

if nargin == 3
    colormap(gray);
   imagesc(tmplate);
end;
```

6. For Finding Minutiae and Minutiae Matching

```
function show_minutia(image,end_list,branch_list);

%show the image of all points in the list
% tkj-5

%[x,y] = size(end_list);
%imag = zeros(200,200);
%imag = image;
%x
%for i=1:x
%   xx = end_list(i,1);
%   yy = end_list(i,2);
%
%   imag(xx-2:xx+2,yy-2:yy+2) = 1;
%   imag(xx,yy) = 0;
%end;


%[x,y] = size(branch_list);
%for i = 1:x
%   xx = branch_list(i,1);
%   yy = branch_list(i,2);

%   imag(xx-2:xx+2,yy-2:yy+2) = 1;
%   imag(xx-1:xx+1,yy-1:yy+1) = 0;
    %imag(xx,yy) = 0;
%end;

%figure;
colormap(gray);imagesc(image);
hold on;

if ~isempty(end_list)

plot(end_list(:,2),end_list(:,1),'*r');
if size(end_list,2) == 3
   hold on
    [u,v] = pol2cart(end_list(:,3),10);
    quiver(end_list(:,2),end_list(:,1),u,v,0,'g');
end;
end;

if ~isempty(branch_list)
hold on
plot(branch_list(:,2),branch_list(:,1),'+y');
end;
```

```
function [end_list,branch_list,ridgeOrderMap,edgeWidth] =
mark_minutia(in, inBound,inArea,block);
% tkj-5

[w,h] = size(in);

[ridgeOrderMap,totalRidgeNum] = bwlabel(in);

imageBound = inBound;
imageArea = inArea;
blkSize = block;

%innerArea = im2double(inArea)-im2double(inBound);


edgeWidth = interRidgeWidth(in,inArea,blkSize);

end_list    = [];
branch_list = [];


for n=1:totalRidgeNum
   [m,n] = find(ridgeOrderMap==n);
   b = [m,n];
   ridgeW = size(b,1);

   for x = 1:ridgeW
      i = b(x,1);
      j = b(x,2);

      %if imageArea(ceil(i/blkSize),ceil(j/blkSize)) == 1 &
imageBound(ceil(i/blkSize),ceil(j/blkSize)) ~= 1
if inArea(ceil(i/blkSize),ceil(j/blkSize)) == 1
      neiborNum = 0;
      neiborNum = sum(sum(in(i-1:i+1,j-1:j+1)));
      neiborNum = neiborNum -1;


   if neiborNum == 1
     end_list =[end_list; [i,j]];

   elseif neiborNum == 3
      %if two neighbors among the three are connected
directly
      %there may be three braches are counted in the nearing
three cells
```

```
        tmp=in(i-1:i+1,j-1:j+1);

        tmp(2,2)=0;
        [abr,bbr]=find(tmp==1);
        t=[abr,bbr];

        if isempty(branch_list)
           branch_list = [branch_list;[i,j]];
        else

        for p=1:3
           cbr=find(branch_list(:,1)==(abr(p)-2+i) &
branch_list(:,2)==(bbr(p)-2+j) );
           if ~isempty(cbr)
              p=4;
              break;
           end;
        end;
```

```
function
[percent_match]=match_end(template1,template2,edgeWidth,noSh
ow)
% MATCH_END Fingerprint Minutia Matcher Based on Ridge
Alignment
%    match_end(template1,template2) accepts the two template
files
%    and return the maximum similiar certainty of the two
fingerprints
%       template file stores an Nx3 matrix with the
following specified format:
%     ------------------------------------------------------
-------------
%       minutia_1_position_x  minutia_2_position_y
minutia_1_orientation
%       ...                   ...                    ...
%     minutia_n_position_x  minutia_n_position_y
minutia_n_orientation
%     ridge_1_point_1_posx    ridge_1_point_1_posy
ridge_ID(1)
%     ...                     ...                      ...
%     ridge_1_point_m_posx    ridge_1_point_m_posy
ridge_ID(1)
%     ridge_2_point_1_posx    ridge_2_point_1_posy
ridge_ID(2)
%     ...                     ...                      ...
%     ridge_n_point_1_posx    ridge_n_point_1_posy
ridge_ID(n)
```

```
%      ...                    ...                    ...
%      ridge_n_point_m_posx   ridge_n_point_m_posy
ridge_ID(n)
%
%        n refers to the total miniutia number
%        m is set to the value of average inter-ridge width
%      ------------------------------------------------------
------------
%    match_end(template1,template2,noShow) also accepts the
flag 'noShow' to disable the
%    popup dialog showing final matching percentage. The
value can be simply set to 0.
%    This function is used for batch process.

%    tkj-5

%   Decompose the template file into minutia and ridge
matrixes seperately
if or(edgeWidth == 0,isempty(edgeWidth))
   edgeWidth=10;
end;

if or(isempty(template1), isempty(template2))
   percent_match = -1;
else
length1 = size(template1,1);
minu1 = template1(length1,3);
real_end1 = template1(1:minu1,:);
ridgeMap1= template1(minu1+1:length1,:);

length2 = size(template2,1);
minu2 = template2(length2,3);
real_end2 = template2(1:minu2,:);
ridgeMap2= template2(minu2+1:length2,:);

ridgeNum1 = minu1;
minuNum1 = minu1;
ridgeNum2 = minu2;
minuNum2 = minu2;


max_percent=zeros(1,3);

for k1 = 1:minuNum1
      %minuNum2

      %calculate the similarities between ridgeMap1(k1) and
ridgeMap(k2)
      %choose the current two minutia as origins and adjust
other minutia
      %based on the origin minutia.
```

```
        newXY1 = MinuOriginTransRidge(real_end1,k1,ridgeMap1);
    for k2 = 1:minuNum2

        newXY2 = MinuOriginTransRidge(real_end2,k2,ridgeMap2);

        %choose the minimum ridge length
        compareL = min(size(newXY1,2),size(newXY2,2));
        %compare the similarity certainty of two ridge
        eachPairP =
newXY1(1,1:compareL).*newXY2(1,1:compareL);
        pairPSquare = eachPairP.*eachPairP;
        temp = sum(pairPSquare);

        ridgeSimCoef = 0;

        if temp > 0
        ridgeSimCoef = sum(eachPairP)/( temp^.5 );
        end;

if ridgeSimCoef > 0.8
          %transfer all the minutia in two fingerprint based
on
          %the reference pair of minutia
           fullXY1=MinuOrigin_TransAll(real_end1,k1);
           fullXY2=MinuOrigin_TransAll(real_end2,k2);

           minuN1 = size(fullXY1,2);
           minuN2 = size(fullXY2,2);
           xyrange=edgeWidth;
           num_match = 0;

           %if two minutia are within a box with width 20 and
height 20,
           %they have small direction variation pi/3
           %then regard them as matched pair

for i=1:minuN1
   for j=1:minuN2
      if (abs(fullXY1(1,i)-fullXY2(1,j))<xyrange &
abs(fullXY1(2,i)-fullXY2(2,j))<xyrange)
          angle = abs(fullXY1(3,i) - fullXY2(3,j) );
          if or (angle < pi/3, abs(angle-pi)<pi/6)
          num_match=num_match+1;
          break;
          end;
      end;
   end;
end;
```

```
% get the largest matching score
current_match_percent=num_match;
if current_match_percent > max_percent(1,1);
   max_percent(1,1) = current_match_percent;
   max_percent(1,2) = k1;
   max_percent(1,3) = k2;
end;
num_match = 0;

end;
end;
end;

percent_match = max_percent(1,1)*100/minuNum1;
end;

%if function is called in GUI mode, popup out the message
box
%for final result
if nargin == 3
   text=strcat('The max matching percentage is
',num2str(percent_match),'%');
msgbox(text);
end;
```

APPENDIX B: SIMULATION IMAGES

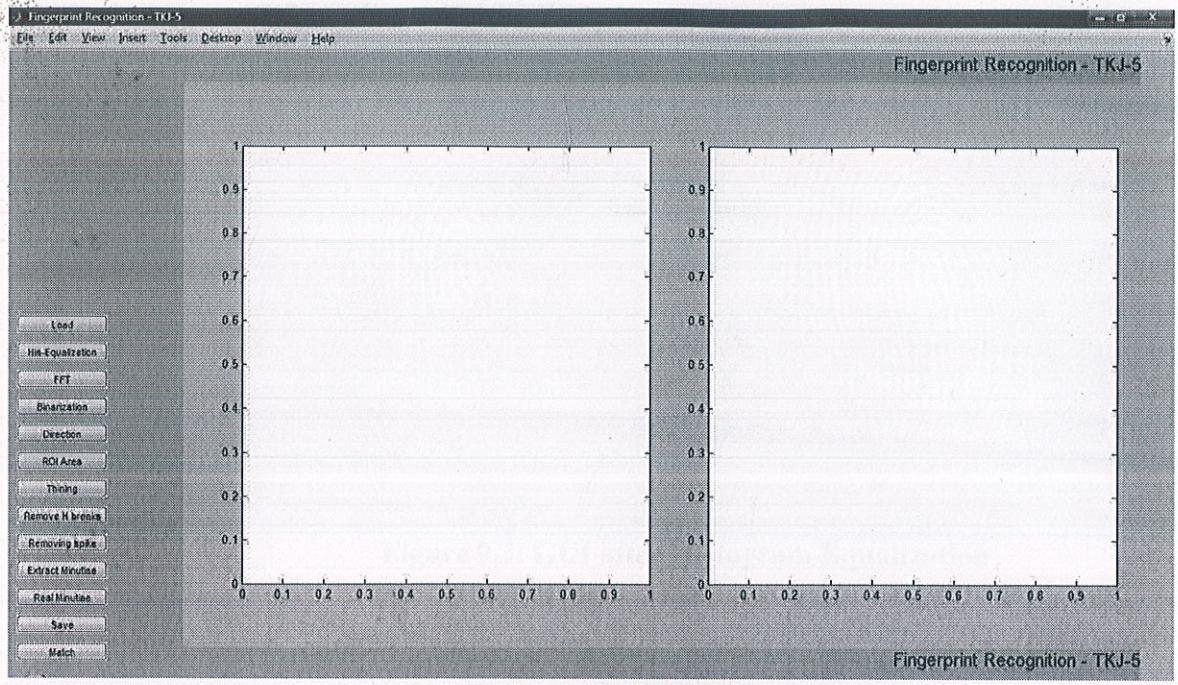


Figure 9.1: GUI Interface of program

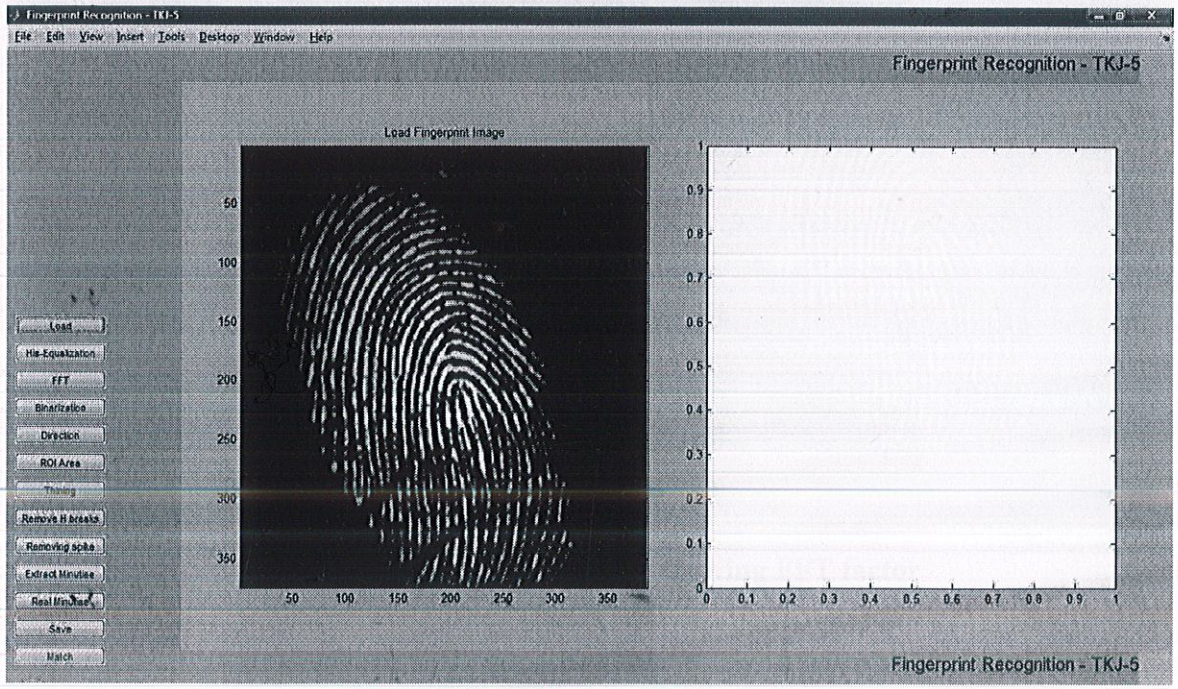


Figure 9.2: GUI after loading an image

Figure 9.3: GUI after Histogram Equalization

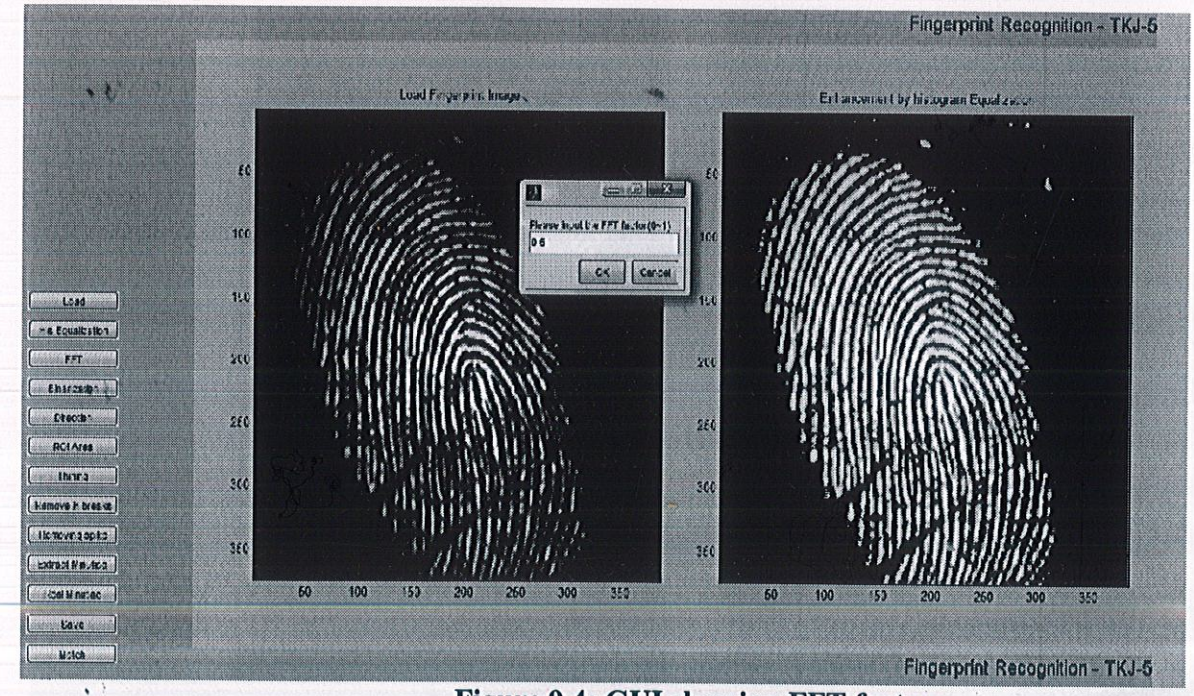


Figure 9.4: GUI showing FFT factor

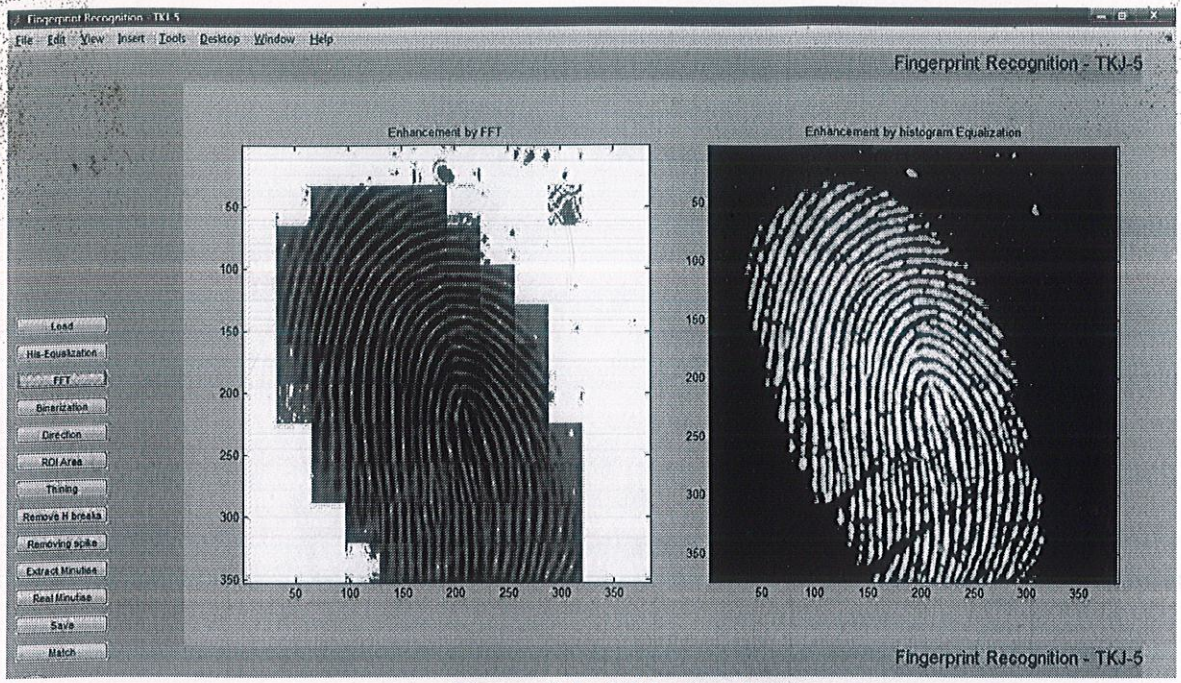


Figure 9.5: GUI after FFT

Figure 9.6: GUI after Binarization

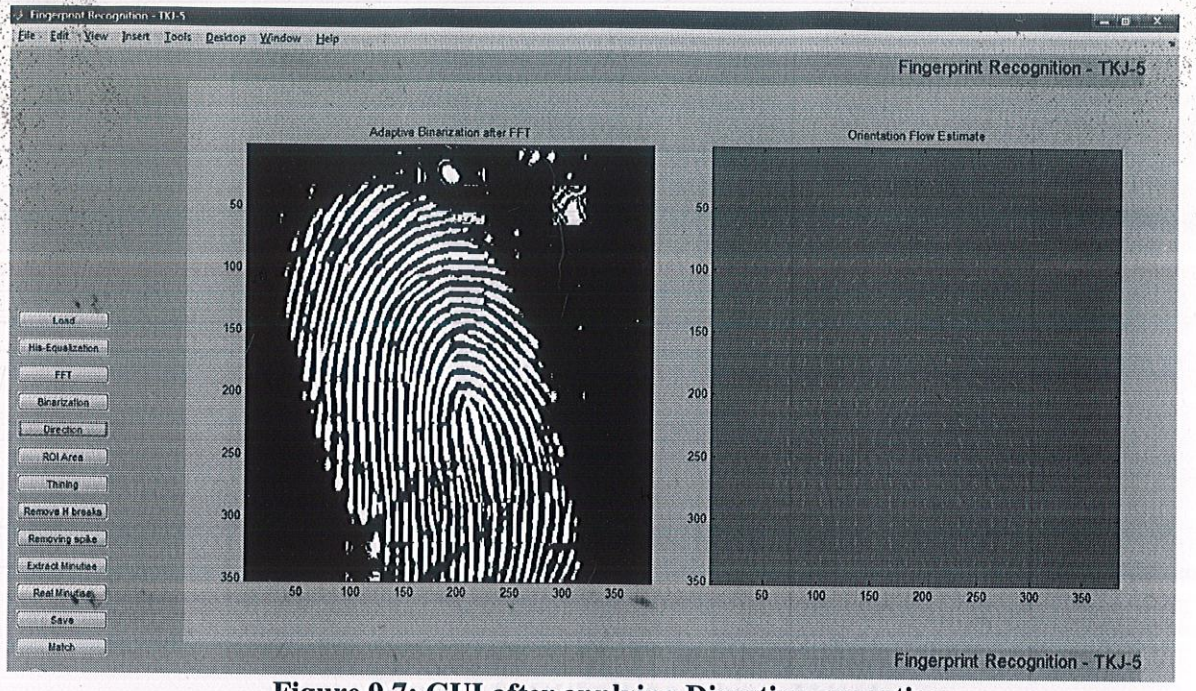


Figure 9.7: GUI after applying Direction operation

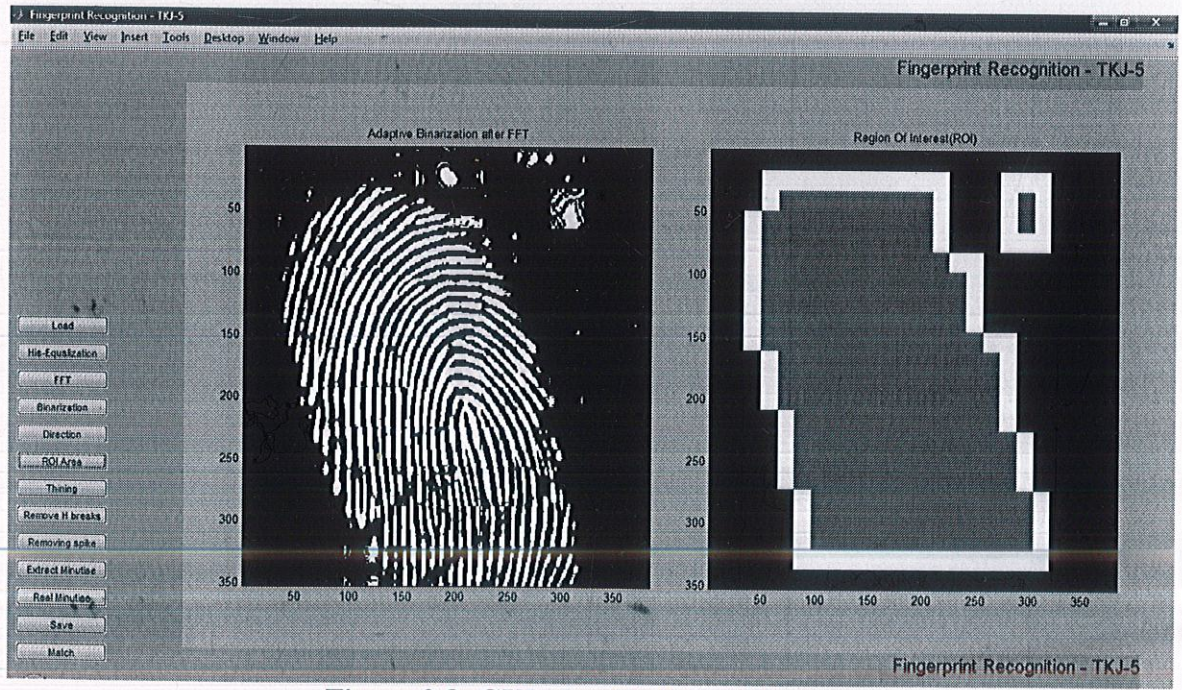


Figure 9.8: GUI after finding ROI

Figure 9.9: GUI after applying Thinning

Figure 9.10: GUI after removing H Spikes

Figure 9.11: GUI after removing spikes

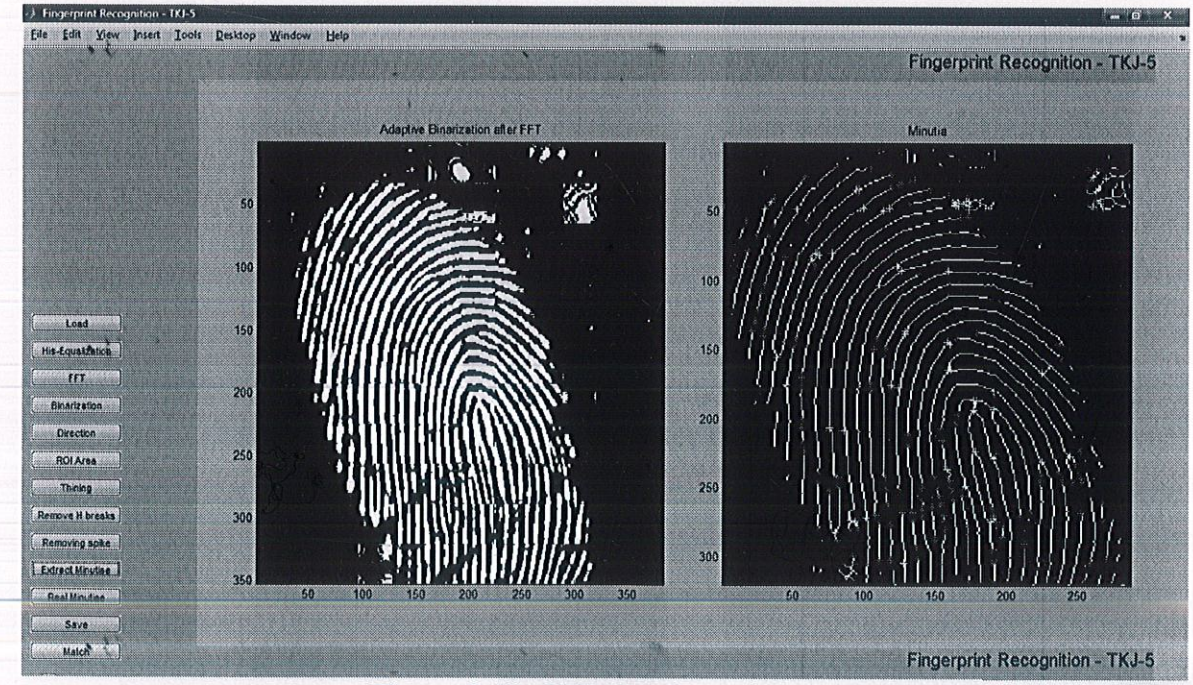


Figure 9.12: GUI showing minutiae extraction

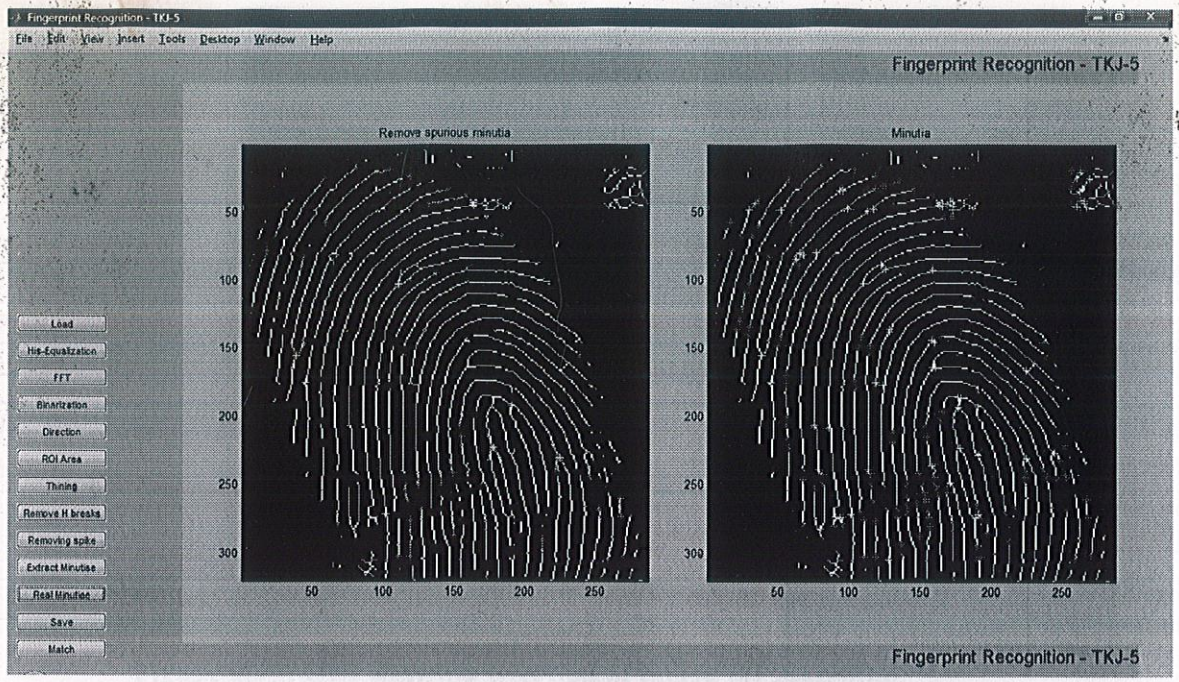


Figure 9.13: GUI showing selected minutiae

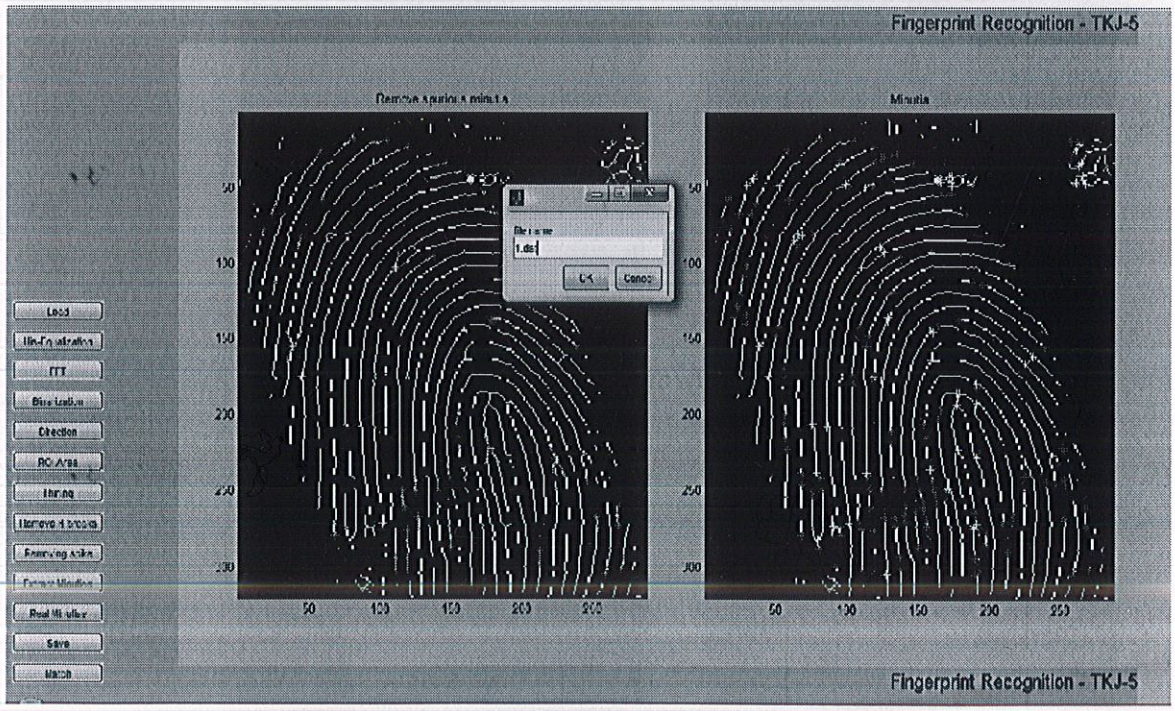


Figure 9.14: GUI showing saving of image file

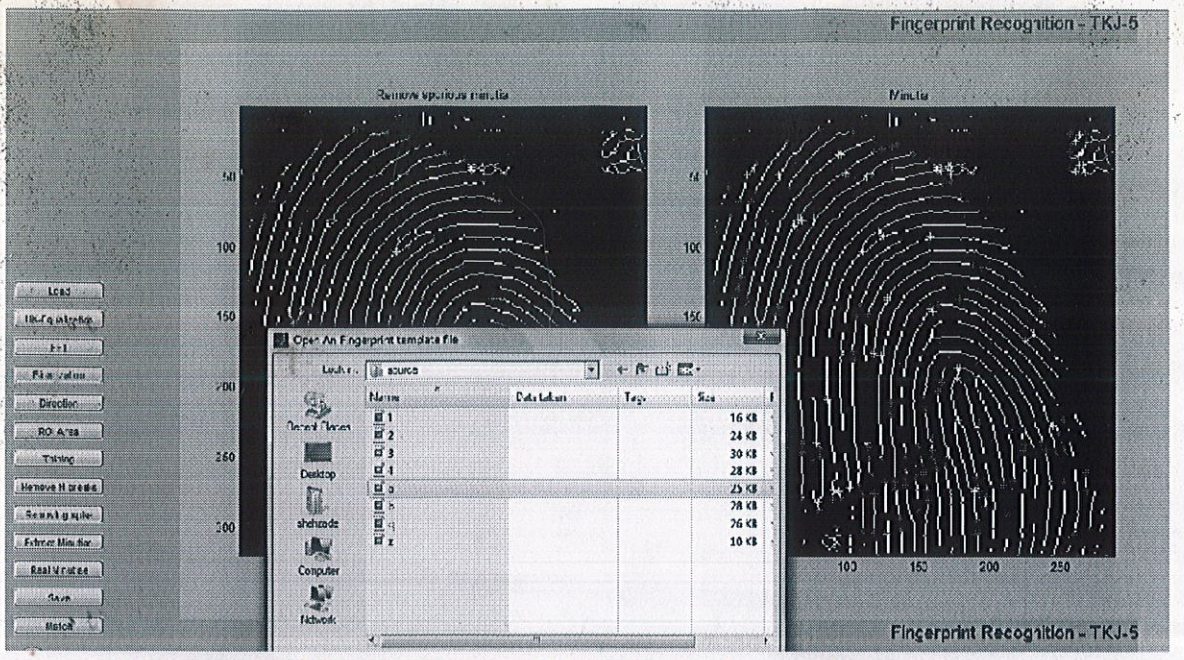


Figure 9.15: GUI showing selection of image for matching

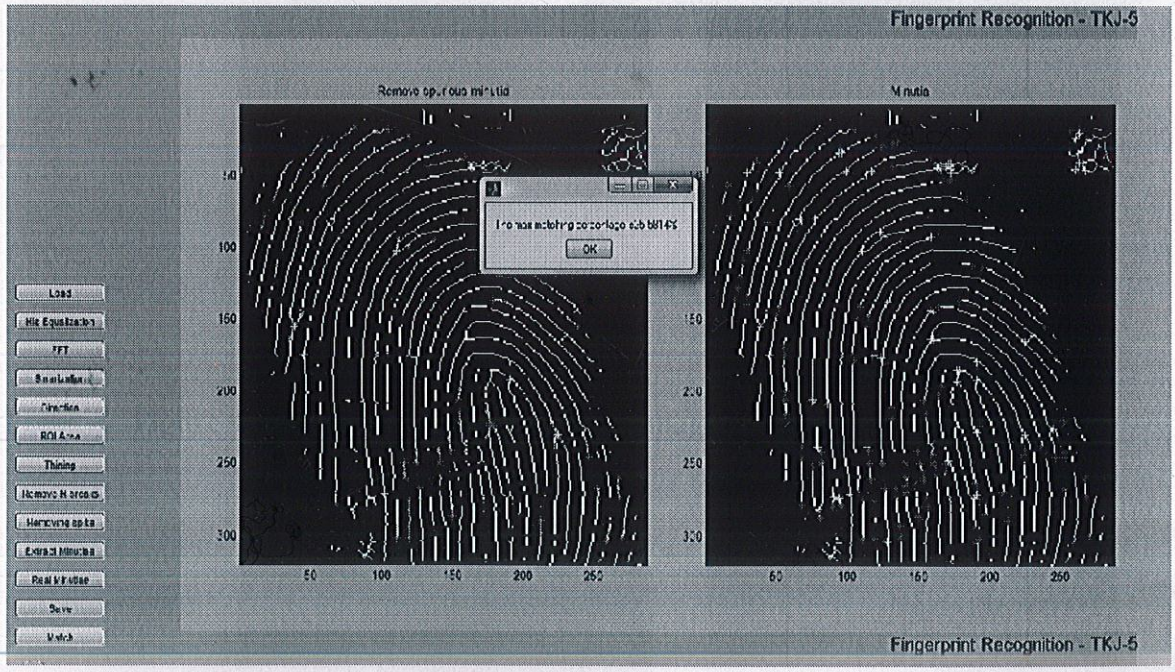


Figure 9.16: GUI showing result after matching two images