Tribhuvan University

Institute of Science and Technology

Comparative Study of Fingerprint Recognition using Standardized Fingerprint Model and Minutiae Score Matching

Dissertation

Submitted to

Central Department of Computer Science & Information Technology

Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science & Information Technology

By

Deepak Bhatt

Date: April, 2013

Tribhuvan University

Institute of Science and Technology

Comparative Study of Fingerprint Recognition using Standardized Fingerprint Model and Minutiae Score Matching

Dissertation

Submitted to

Central Department of Computer Science & Information Technology

Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science & Information Technology

Deepak Bhatt

Date: April, 2013

Supervisor

Prof. Dr. Shashidhar Ram Joshi

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science & Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

... ...

Deepak Bhatt

Supervisor's Recommendation

I hereby recommend that this dissertation prepared under my supervision by **Mr. Deepak Bhatt** entitled "**Comparative Study of Fingerprint Recognition using Standardized Fingerprint Model and Minutiae Score Matching**" in partial fulfillment of the requirements for the degree of M.Sc. in Computer Science and Information Technology be processed for the evaluation.

... ...

Prof. Dr. Shashidhar Ram Joshi

Department of Electronics & Computer Engineering,

Institute of Engineering,

Pulchowk, Nepal

Date: 9th April, 2013

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science & Information Technology

LETTER OF APPROVAL

We certify that we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Masters Degree in Computer Science and Information Technology.

Evaluation Committee

... ...
Asst. Prof. Nawaraj Paudel
Central Department of Computer Science& Information Technology, Tribhuvan University, Kathmandu, Nepal
(Acting Head)

....
Prof. Dr. Shashidhar Ram Joshi
Department of Electronics & Computer Engineering, Institute of Engineering, Pulchowk, Kathmandu, Nepal
(Supervisor)

... ...

(Internal Examiner)

... ...

(External Examiner)

Date: 15th April, 2013

Abstract

Among the various biometric techniques, Fingerprint recognition is one of the most popular and accurate. Fingerprints are used to authenticate a person. Since these biometric data are permanent throughout the life of individuals. However there are various challenges associated with fingerprints, like poor quality of input image, deformed images due to elastic nature of the skin, roughness of skin etc. fingerprints are accepted in many real world applications. In recent years many algorithmic models have been proposed to improve the accuracy as well as time invariants of the recognition system. Among these algorithms, here **Fingerprint Recognition using Minutiae Score Matching** and **Fingerprint Recognition using Standardized Fingerprint Model** are chose and projected a comparative analysis of their accuracy. The false acceptance ratio and false rejection ratio as well as time efficiency is also analyzed.

It is found that False Acceptance Ratio and False Rejection Ratio for FRSFM is less than 0.28% and 0.0056% respectively with an average score of 0.85. Whereas the best result for fingerprint recognition is achieved at threshold value 0.6 for time parameter as the main invariant.

Keywords: *Fingerprints, Fingerprint Recognition, Binarization, Transformation, Fingerprint Representation, Threshold value.*

Acknowledgement

With deep sense of gratefulness I would like to express my great gratitude to my honorable supervisor, **Prof. Dr. Shashidhar Ram Joshi,** Head of electronics & computer Engineering Department (Kathmandu, Nepal) for his valuable guidance in carrying out this work under his effective supervision and enlightenment.

I would like to thank my respected teacher **Mr. Bikash Balami,** Lecturer of Central Department of Computer Science and IT (Kathmandu, Nepal) for his crucial ideas during carrying out this work.

I would also like to thank **Dr. Wuzhill,** of Honkong Baptist University for his fingerprint database, which I got from his personal website. I also want to convey my thanks to MATLAB developer team.

My special thanks to **Mr. Kushendra Prasad Bhatta** of Education Department (Kathmandu, Nepal), **Mr. Keshav Bahadur Dhami, Ms. Shikha Karmacharya, Mr. Kepisee Thapa, Mr. Bhupendra singh Saud** and **Mr. Roshan Pulami** of Computer Science and IT Department (Kathmandu, Nepal) for providing their fingerprints.

I also take opportunity to thank all others who give me support for this work or in other aspects of my study at Central Department of Computer Science & IT.

Finally, I would like to thank my family members for their love and blessings. Without whom I may not make any sense to this research work. What I am today is because of their love, guidance and support.

In loving memory of my father ……..

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

ANSI-NIST	American National Standards Institute-National Institute of Standard and Technology
AFAS	Automatic Fingerprint Authentication System
AFIS	Automated Fingerprint Identification Systems
ATM	Automatic Teller Machine
BMP	Bit Map Picture
D	Distance
DPI	Dots per Inch
EER	Equal Error Ratio
FA	False Acceptance
FAR	False Acceptance Ratio
FFT	Fast Fourier Transformation
FMR	False Matching Ratio
FNMR	False Non-Matching Ratio
FR	False Rejection
FRSFM	Fingerprint Recognition using Standarized Fingerprint Model
FRMSM	Fingerprint Recognition using Minutiae Score Matching
FRR	False Rejection Ratio
FVC	Fingerprint Verification Championship
ID	Identity

IPT	Image Processing Toolbox
JPEG	Joint Photographic Experts Group
NTSC	National Television System Committee
PIN	Personal Identification Number
RGB	Red Green Blue
TM	Total Minutiae
TIF	Tagged Image File
WSQ	Wavelet Scalar Quantization

Chapter 1

INTRODUCTION

Fingers are not typical assets of human, because the apes have fingers too. But one attribute is specific for human fingers - we have a unique art of skin corrugation which forms different patterns. Such patterns can be found not only in the fingers, but also on the whole palm and on the sole of our feet, including toes.

1.1 Introduction

A fingerprint is the feature pattern of one finger (Figure 1.1). These patterns are fully developed under pregnancy and are permanent throughout whole lifetime. Biologically these patterns are the growth of epidermis to provide a proper grip to the individual. Prints of these patterns are called fingerprints. Injuries like cuts, burns and bruises can temporarily damage quality of fingerprints but when fully healed, patterns will be restored.

It is believed with strong evidences that each fingerprint is unique. Each person has his/her own fingerprints with the permanent uniqueness. So fingerprints have being used for identification and forensic investigation for a long time [Wuh02].

Figure1.1: Inked impression of fingerprint on paper obtained via scanner

A fingerprint is composed of many ridges and furrows. However, intensive research on fingerprint recognition shows that, fingerprints are not distinguished by their ridges and furrows, but by Minutia, which are some abnormal points on the ridges (Figure 1.2). Among the variety of minutia types reported in literatures, two are mostly significant and in heavy usage: one is called termination, which is the immediate ending of a ridge; the other is called bifurcation, which is the point on the ridge from which two branches derive.

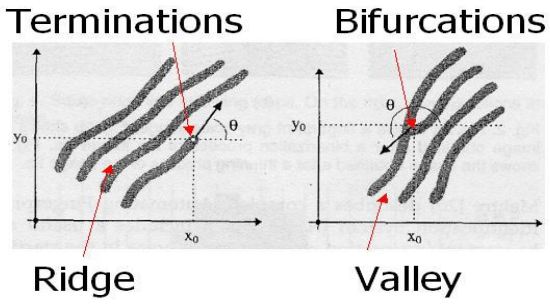


Figure: 1.2: Minutiae

1.1.1 Motivation

In general, fingerprint recognition systems are considered as being of high security strength. Since a fingerprint data is in order of several kilobytes, it provides the security of having a long password without having the overhead of remembering the information. However unlike a password system, in which an exact match is expected for authentication of an individual, a fingerprint recognition system can only provide the individual's identity with a certain confidential level. Thus, some kind of distortion tolerant mechanism is required which can reduce the security strength of the system. Moreover, with the small number of features on the partial fingerprint, the security strength of partial fingerprint recognition is further diminished. Before replacing the password and PIN authentication systems with fingerprint recognition, the question should be answered how efficient and time saving algorithms (system) have been developed for Automated Fingerprint Identification Systems (AFIS).

1.1.2 Fingerprint Recognition

The fingerprint recognition problem can be grouped into two sub-domains:

i. Fingerprint verification

ii. Fingerprint identification

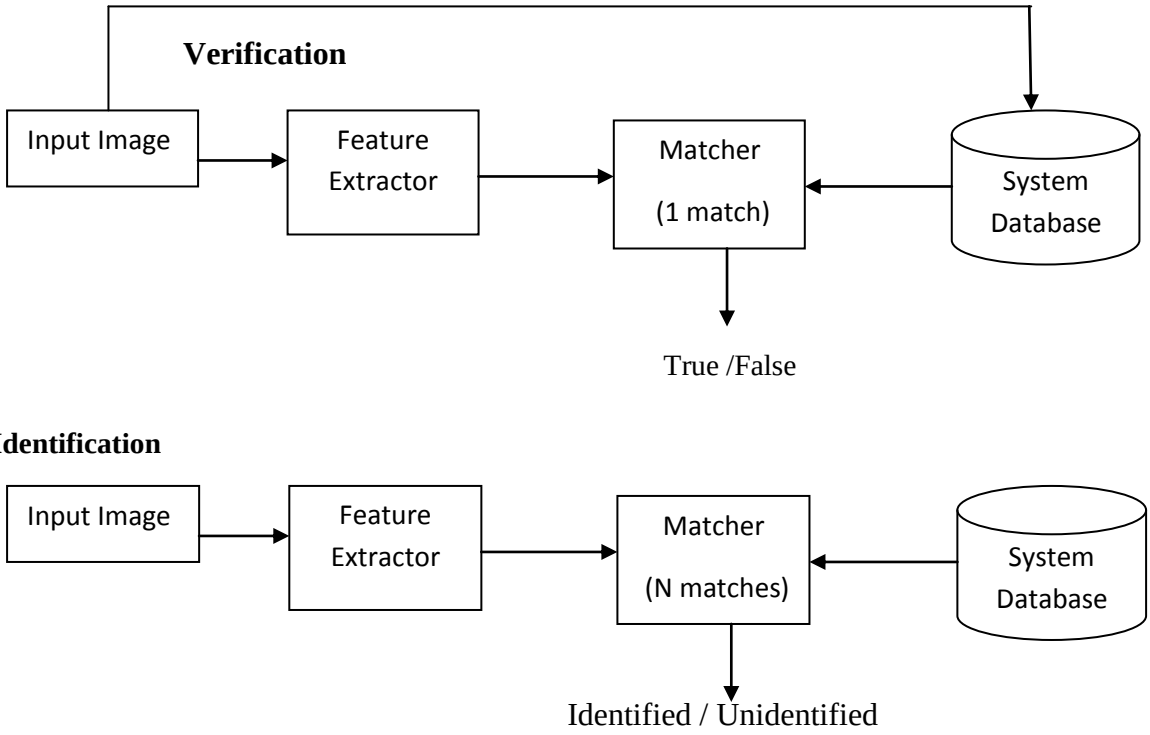


Figure: 1.3: Basic models for fingerprint verification and identification processes.

1.1.2.1 Fingerprint Verification

Fingerprint verification is to verify the authenticity of one person by his/her fingerprint. The user provides his/her fingerprint together with his/her identity information like his/her ID number. The fingerprint verification system retrieves the fingerprint template according to the ID number and matches the template with the real-time acquired fingerprint from the user. Usually it is the underlying design principle of AFAS (Automatic Fingerprint Authentication System).

1.1.2.2 Fingerprint Identification

Fingerprint identification is to specify one person's identity by his/her fingerprint. Without knowledge of the person's identity, the fingerprint identification system tries to match his/her fingerprint with those in the whole fingerprint database. It is especially useful for criminal investigation cases. And it is the design principle of AFIS (Automatic Fingerprint Identification System).

However, all fingerprint recognition problems, either verification or identification, are ultimately based on a well-defined representation of a fingerprint. As long as the representation of fingerprints remains the uniqueness and keeps simple, the fingerprint matching, either for the 1-to-1 verification case or 1-to-m identification case, is straightforward and easy.

1.1.3 Applications of Fingerprint Recognition System

The main application areas of fingerprint recognition are AFAS as well as forensic investigation, where recognition is based on verification mode and identification mode respectively. Besides these, fingerprints are now widely used in the real world applications where security is the major concern.

Application of fingerprint recognition may be found in the future of World Wide Web, electronic businesses, where fingerprints can be used as verifying agent. Most of the ATMs used PIN for verifying the user; fingerprints may be used to replace PINs.

The daily attendance system, sever login, desktop login are some other applications of fingerprint recognition system.

1.2 Challenges

To acquire and match fingerprints is a great challenge. The nature of the skin as well the work the person does plays a vital role in acquiring as well as matching the fingerprint. Since fingerprints are of elastic nature, the image acquired via optical sensor as well as paper impression may vary. Besides these the cuts or spurs present on the finger due to some physical damage is the another challenge. Also the dirt or roughness of skin puts another challenge for the fingerprint recognition system. Some of the major challenges are as:

- Low quality finger prints.
- Deformed fingerprints due to elastic nature of skin.
- Spurs, cuts and roughness of skin.
- Angle of impression.
- Quality of fingerprint acquiring device.

1.3 Problem Definition

In the context of fingerprint recognition, fingerprints or simply prints are generally used to refer to the impression of human finger. The basic goal of Fingerprint Recognition is to identify whether the person using the biometric is the one whose template image is stored in the database or not. This dissertation is focused on comparative study of existing algorithms named FRSFM and FRMSM. The fingerprint recognition can be decomposed into following three fundamental tasks:

- Fingerprint acquisition,
- Preprocessing and
- Fingerprint matching

1.4 Objectives

The objective of this research work is to investigate various feature extraction techniques and to compare fingerprint recognition techniques namely Fingerprint Recognition using Standardized Fingerprint Model and Fingerprint Recognition using Minutiae Score Matching to improve accuracy of Fingerprint Recognition. Comparative Performance Matrices are analyzed. The sub-problem field of fingerprint recognition is also addressed. Main objectives are:

- To compare performance and efficiency of Fingerprint Recognition using Standardized Fingerprint Model and Fingerprint Recognition using Minutiae Score Matching at different threshold values.
- To study and analyze preprocessing techniques (Binarization, Transformation, crossing number computation, minutiae extraction, block filter) for fingerprints.

1.5 Outline of the Thesis

The remaining part of the document is organized as follows:

Chapter 2 Describes the state of art of the fingerprint recognition. It includes the methods and techniques used in the area of fingerprint recognition till now.

Chapter 3 Describes the tool and technologies used to put forward this research work. It also includes the description of how data is collected and analyzed.

Chapter 4 Deals with the description of work like Preprocessing, Minutiae Extraction as well as the Matching procedure fingerprints.

Chapter 5 Deals with the experimentation details, conclusion and further recommendation of the research work.

Chapter 2

RELATED WORKS

2.1 Previous Work

The scientific study of papillary ridges of the hands and feet is credited as beginning with the work of *Joannes Evangelista Purkinje*, a Czech physiologist and biologist in 1823. Though some scientist of 17th century have tried study the fingerprint patterns, the first attempt to systematically categorize fingerprint patterns is found in the work of *Purkinje*. He used a nine pattern classification. In 1880 two papers written *Henry Faulds* and *W. J. Herschel* appeared in Nature recommending the use of fingerprints for personal identification. *Herschel* reported actually using this method of identification in India. *Faulds* reported his interest in fingerprints dated from finding impressions of them on ancient Japanese pottery. [Cam98] In the early twentieth century *Harris Hawthorne Wilder*, pioneered comprehensive studies of the methodology, inheritance and racial variation of palmer and planter papillary ridge patterns as well as fingerprints. He began to publish a series of papers on these subjects in 1902 and continued publication through 1916. [Cam98]

The second quarter of the twentieth century, the field was dominated by *Harold Cummins*, sometime professor of Microscopic Anatomy at Tulane University.

To improve the performance of matching, various approaches have been proposed: *Chen et. al.* [Che91] adopted the concept of tree matching in their approach; *Isenor et. al.* [Wan06] use graph matching to solve the problem of elastic deformation; *Jain* and *Lee et. al.* [Jai00-Lee99] represents the fingerprint with texture information extracted by Gabor filters and the matching is based on Euclidian distance. Recently, *Jain et. al.* [Jai01] combined texture features and minutiae feature in matching; *Vajna* [Vaj00] combined triangular matching and dynamic time warping to tolerate nonlinear deformation; the use of vector field computation and core detection came into existence [Alf93] which is still used by different researcher for fingerprint recognition. The error based propagation

matching was carried on with some accepted result, some problem remained unsolved like Minutiae similarities, template similarities [Hao99]. During 2004 the work on fingerprint recognition using Phase Only Correlation was carried out by some Japanese and it assured genuine fingerprint acceptance up to 97.5%, but before than the acceptance ratio was 81.7% [Ito04]. In the following year , same researchers came with another model which worked for fingerprints obtained from low quality sensors, from dirty fingers using phase based finger print matching [Ito05] with satisfactory results, using the same techniques as in the previous paper. The use of triangle matching i.e. Delaunay Triangulation for minutiae matching is found in [Wan06].

In 2009 *Ravi J. et.al*. came with a model which use FRMSM (Fingerprint Recognition using Minutiae Score Matching) model. This model follows the sequence of steps like binarization, thinning, Data matrix computing and finally matching with the template image, and guarantees the better matching ratio with decrease in FMR (False Matching Ratio) and FNMR (False Non Matching Ratio). [Rav09]

Similarly *L.H. Thai* and *et. al.* came with FRSFM (Fingerprint Recognition using Standardized Fingerprint model). This uses genetic algorithm for transformation on an image and the process of matching guarantees the highest degree of matching. [Tha10]

2.2 Fingerprint Representation

Fingerprint representation (template) is a machine readable form of a fingertip epidermis. It influences the system's accuracy, time in both approaches of recognition i.e. verification as well as identification and the design of the system.

There are mainly three different kinds of fingerprint representations that are used by today's fingerprint recognition systems, each with some advantages as well as drawbacks [Jea05].

2.2.1 Image-based Representation

This is the simplest way of fingerprint representation. In this representation, the fingerprint image itself is used as a template. There is no need for a specific feature

extracting algorithm, and the raw intensity pixel values are directly used. This representation preserves the most information about a fingerprint. However, image-based representation requires tremendous storage space. For example, a 0.8mm X 1.0mm (400 X 500 pixels) fingerprint image obtained by a scanner at 500 dots per inch (DPI) with 8 bits gray-scale resolution. The resulting fingerprint image is 400 X 500 = 0.2 Megabytes. A system with large amount of fingerprint data may have difficulty storing all the templates. However, some traditional compression techniques, such as JPEG and WSQ can reduce the size of image to 20 Kilobytes without discriminating information [Bri95].

In this research work the same approach is used but only after conversion of the original RGB image into gray-scale image.

2.2.2 Global Ridge Pattern

This representation relies on the ridge structure, global landmarks and ridge pattern characteristics, such as the singular points, ridge orientation map, and the ridge frequency map. This representation is sensitive to the quality of fingerprint images [Jai97]. However, the discriminative abilities of this representation are limited due to absence of singular points.

2.2.3 Local Ridge Detail

This is the most widely used and studied fingerprint representation. Local ridge details are the discontinuities of local ridge structures referred to as minutiae. Sir Francis Galton (1822-1922) was the first person who observed the structures and permanence of minutiae. Therefore, minutiae are also called “Galton details”.

There are about 150 different types of minutiae [Jai97] categorized based on their configuration. Among these minutia types, “ridge ending” and ”ridge bifurcation” are the most used, since all other types of minutiae can be seen as combination of “ridge ending” and “ridge bifurcation” (Figure 2.1).

The American National Standards Institute-National Institute of Standard and Technology (ANSI-NIST) proposed a minutiae-based fingerprint representation. It

includes minutiae location and orientation [ANSI93]. The minutia orientation is defined as the direction of the underlying ridge at the minutia location (Figure 2.2). Minutiae-based fingerprint representation also has an advantage in helping privacy issues since one cannot reconstruct the original image from using only minutiae information. Minutia is relatively stable and robust to contrast, image resolutions, and global distortion when compared to other representations. However, to extract the minutiae from a poor quality image is not an easy task.

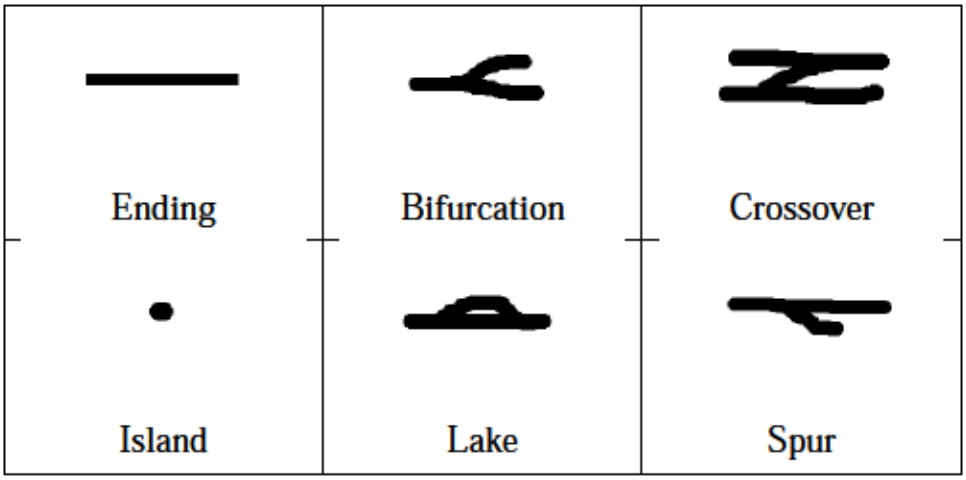


Fig. 2.1: Some of the common minutiae types

Today, most of the automatic fingerprint recognition systems are designed to use minutiae as their fingerprint representation.

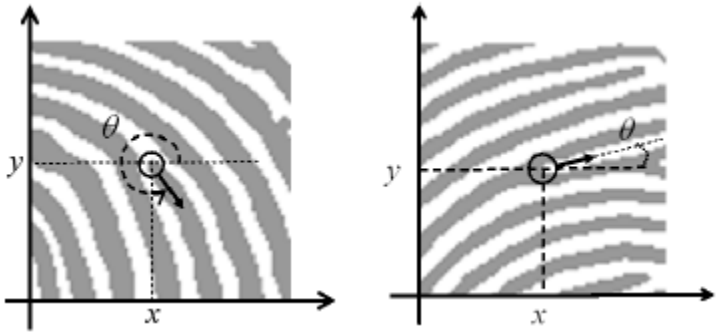


Figure 2.2: (a) A ridge ending minutia: (x, y) are the minutia coordinates; θ is the minutia's orientation; (b) A ridge bifurcation minutia: (x, y) are the minutia coordinates; θ is the minutia's orientation.

2.2.4 Intra-ridge Detail

On every ridge of the finger epidermis, there are many tiny sweat pores (Figure 2.3). Pores are considered to be highly distinctive in terms of their number, positions and shapes. However, extracting pores is pores is feasible only in high-resolution fingerprint images (for example 1000 DPI) and with good image quality. Therefore, this kind of representation is not practical for most applications.

2.3 Fingerprint Recognition

2.3.1 Minutiae – Based

Minutiae – based fingerprint representation and matching are widely used by both machine and human experts. Minutiae representation has several advantages compared to other fingerprint representation (Section 2.2.3). Minutiae have been (historically) used as key features recognition tasks. Its configuration is highly distinctive and several theoretical models [Sto86, Lee01, Pan02] use it to provide an approximation of the individuality of fingerprints. Minutiae-based systems are more accurate than correlation based systems [Mai97] and the template size of minutiae-based fingerprint representation is small. Forensic experts use this representation which has how become part of several standards [ANSI93] for exchange of information between different systems across the world.

2.3.2 Correlation Based

Each minutia is represented by a fixed number of attributes such as the location, orientation, type and other local information. A hard decision is made on the match between a pair of minutiae based on the similarity of these attributes. Correlation based has two main advantages. Since the gray level values of the pixels around a minutia point retain most of the local information, spatial correlation provides an accurate measure of the similarity between minutia regions. Secondly, no hard decision is made on the correspondence between a minutia pair. Instead the quality of all the minutiae matches is

accumulated to arrive at the final matching score between the template and query fingerprint impressions. [Jai04]

2.4 Minutiae Extraction

In automatic fingerprint recognition, the reliability of minutia features. Generally the minutiae representation of a fingerprint consists of simply a list of minutiae points associated with their spatial coordinates and orientation. Some methods also include the types [Lee02, Pra03] and quality of minutiae in the representation.

Minutiae extraction algorithms are of two types:

i. Binarization-based Extraction
ii. Gray-scale based extraction

2.4.1 Binarization Based Minutiae Extraction

Most of the proposed minutiae extraction methods are binarization-based approaches. They require conversion of the gray-scale fingerprint image (8 bits per pixel, 256 gray levels) into a binary form (2 bits per pixel, black and white). Various binarization techniques have been represented in the image processing literature [Tri95]. One intuitive approach is to use a global threshold t and assign each pixel a value according to following equation:

$$I_B(x,y) = \begin{cases} 1 \; if \; I(x,y) > \; t \\ 0 \text{ if } \mathrm{I(x,y)} \leq \; \mathrm{t} \end{cases} \qquad (2.1)$$

Where I(x, y) is the intensity value of the pixel at (x, y) in a gray-scale image. Otsu's method [Ots79] describes a technique to obtain the global threshold t from a statistical viewpoint.

2.4.2 Gray-scale Minutiae Extraction

Approaches that works directly on gray-scale images to extract minutiae are proposed overcome some of the problems caused by fingerprint binarization and thinning. Most of these algorithms are based on ridge tracing. Given a starting point (x_0, y_0) and a direction

θ_0, the method of *Maio et al*. [Mai97] tracks the ridges in the gray-scale image by sailing according to the local orientation of the ridge pattern.

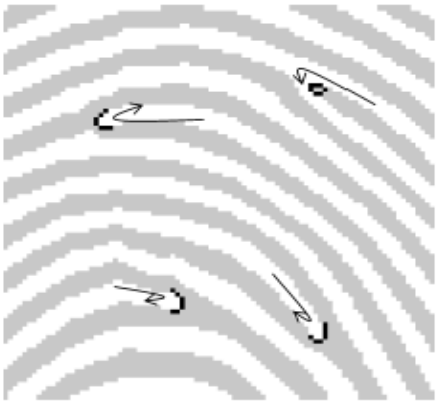


Figure 2.3: Orientation of ridge patterns

2.5 Matching Techniques

The task of comparing a test fingerprint that is provided by the user, to a template fingerprint that is provided earlier, during enrollment is known as matching. Basically there are three types of techniques used in fingerprint matching, namely

- Minutiae Based
- Radial Structure Based
- Correlation Based

2.5.1 Minutiae Based

Most fingerprint matching systems are based on the minutiae. A minutiae-based fingerprint matching system roughly consists of two stages. In the minutiae extraction stage, the minutiae are extracted from the gray-scale fingerprint, while in the minutiae matching stage; two sets of minutiae are compared in order to decide whether the fingerprints are of same finger.

In minutiae matching, two stages can be distinguished. First, registration aligns both fingerprints as far as possible. Most algorithms use a combination of translation, rotation and scaling for this task. After registration, the matching score is determined by counting the corresponding minutiae pairs that are present in both fingerprints. Two minutiae

correspond if a minutia from the test set is located within a bounding box or tolerance zone around a minutia from the template set. The matching score, which is a number in the range from 0 to 1, is calculated as the number of matched minutiae divided by the total number of minutiae.

2.5.2 Radial Structure Based

The radial structure is defined as follows. Figure 2.4 (The correlation factor of the radial structure) shows an example of the radial structure. A set $P = \{p_1, ..., p_n\}$ for $n \in Z$ is a fingerprint image consisting of n minutia points. The radial structure of point $p_i \in P$, denoted $R(p_i)$, is defined to the set of all neighborhood minutiae sharing the edge with minutia pi. We call the center minutia of $R(p_i)$ as c_i, and the neighborhood minutiae of $R(p_i)$ as n_i out of order. So, an arbitrary $R(p_i)$ is defined as

$$R(p_i) = \{c_i, n_1, n_2, ..., n_j\} \text{ for } 2 \leq j \leq n \quad (2.2)$$

Figure 2.5 (radial structure) shows that a radial structure consisting of center minutia and its neighbors in a fingerprint image after Voronoi diagram is constructed. The minutiae are extracted in the fingerprint image and the radial structures are formed for each $R(p_i)$ is saved with a text file with a form of following set to perform the matching stage.

$$R(p_i) = \{(c_{ix}, c_{iy}, c_{iO}, c_{iT}), (n_{1x}, n_{1y}, n_{1\theta}, n_{iO}, n_{1T}), ..., (n_{jx}, n_{jy}, n_{j\theta}, n_{iO}, n_{jT})\} \quad (2.3)$$

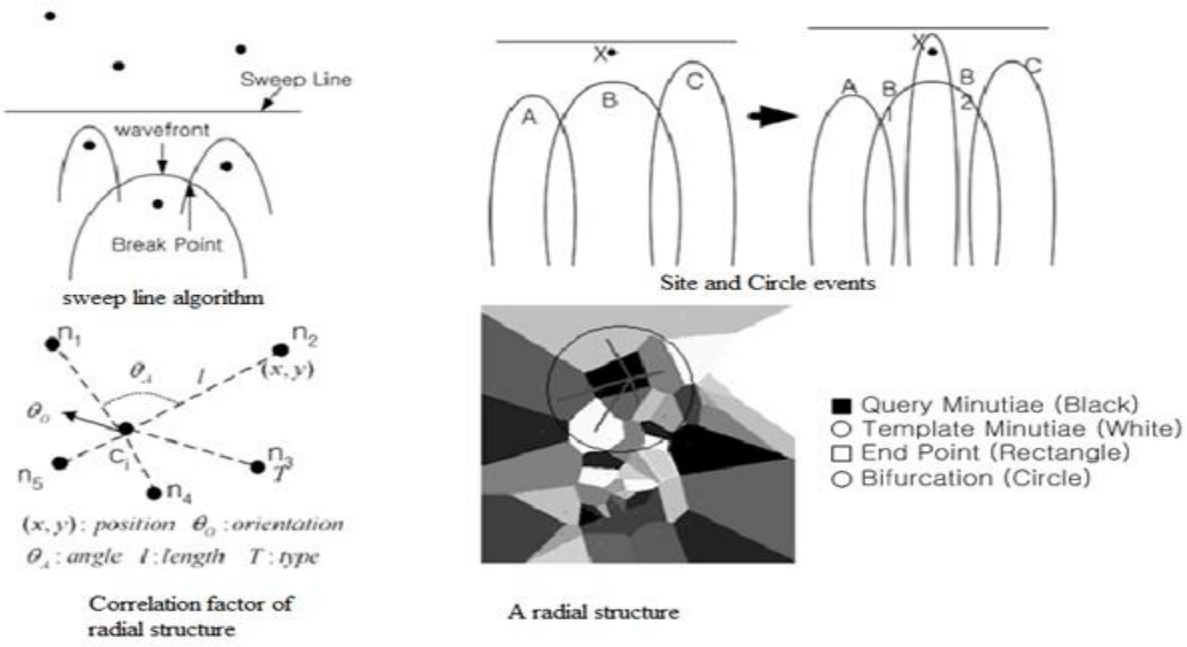


Figure 2.4: Radial structure details [Jea05]

2.5.3 Correlation Based

The minutiae extraction algorithm is applied to the template and the query fingerprint images and the minutia points and the associated ridge points are extracted. The query image is rotated using the estimated rotation. The template and the rotated query images are enhanced using a modified version of the Gabor filter-based enhancement technique. This method is based on a Markov random field model and generates a smooth orientation field in blocks of size $b \times b$ pixels. There is a trade-off in the selection of an appropriate value of b. If b is small, the orientation estimation is more accurate, but is more susceptible to noise. When b is large, the estimation of the orientation field in the regions of high curvature is not very accurate. An optimum value of b must be chosen according to the nature of the fingerprint images in the database. If the variation of the filtered values across the different orientations is small in the entire block, it indicates that the particular block is not oriented along any specific direction. This can occur if the block comes from the background region or is too noisy and hence, unrecoverable. Therefore, such blocks are marked as background.

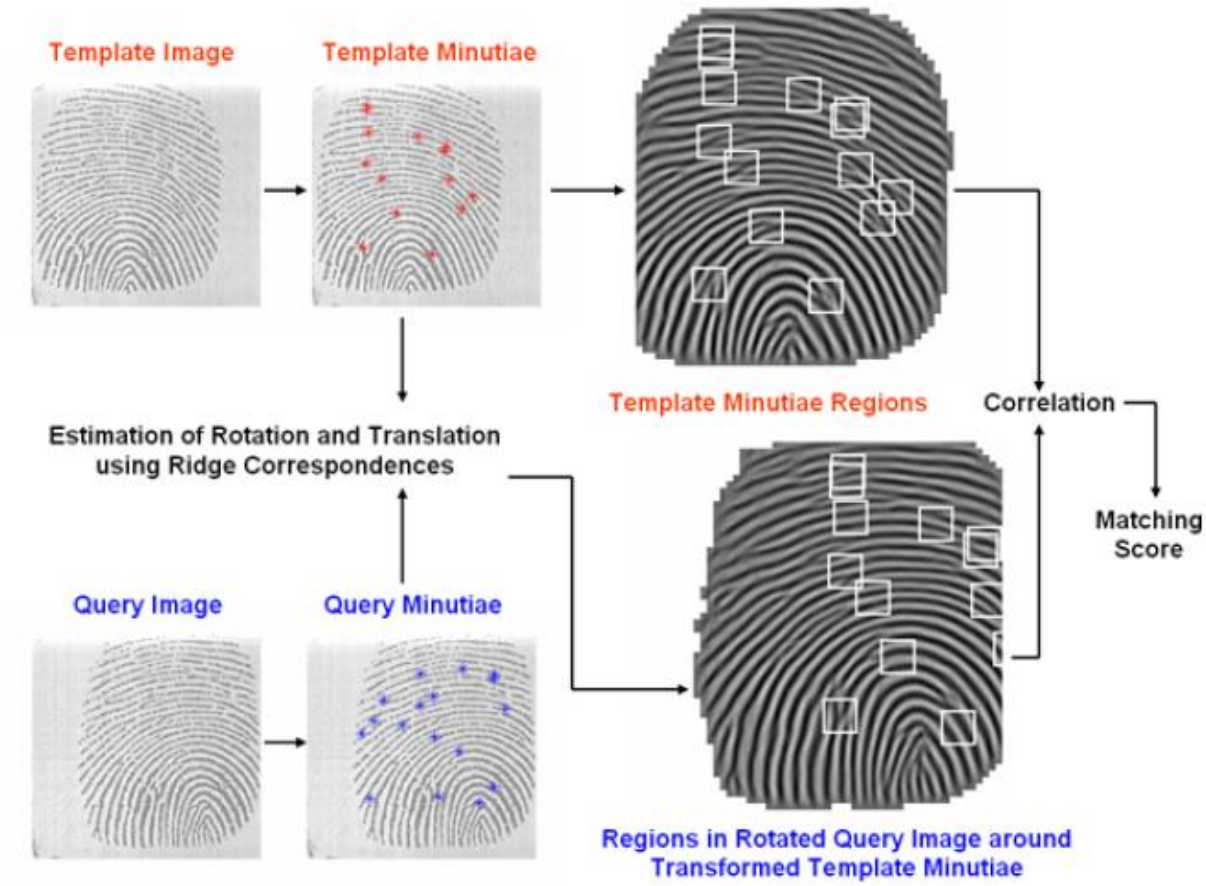


Figure 2.5: Algorithm for correlation based matching [Jea05]

Chapter 3

METHODOLOGY AND DESCRIPTION

All the algorithms of proposed fingerprint recognition system are implemented in MATLAB version 7.12.0.635 (R2011a). MATLAB is installed on a Intel(R) Core(TM) i5 CPU M 430 @ 2.27 GHz 2.27 GHz processor. The machine has total main memory of 4 gigabyte and 64-bit Microsoft Windows 7 Ultimate installed in it.

3.1 MATLAB

The name MATLAB stands for MATrix LABoratory. MATLAB is high-level technical computing language having high-performance computation. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses of the MATLAB are given below.

- Mathematical and scientific computation.
- Algorithm development.
- Application development, including Graphical User Interface building.
- Data analysis, exploration and visualization.
- Modeling, simulation and prototyping.
- Graphics, in 2D and 3D, including colour, lighting and animation.

The reason behind using MATLAB for my research work experimentation is its wide variety and flexible toolboxes. Toolboxes are comprehensive collections of MATLAB functions (Mfiles) that extend the MATLAB environment to solve particular classes of problems. In this research work, mainly image processing toolbox and neural network toolbox are used along with other basis functionality of the MATLAB.

3.2 Image Processing Toolbox

Image Processing Toolbox (IPT) is very rich MATLAB toolbox for working with images. It includes many powerful and very efficient image processing functions. It provides a comprehensive set of standard algorithms and graphical tools for image processing, analysis, visualization and algorithm development. IPT can be used to

perform complex image processing tasks like image enhancement, image de-blurring, feature detection, noise reduction, image segmentation, geometric transformations, image registration, etc. Many toolbox functions are multithreaded to take advantage of multi-core and multiprocessor computers.

Here, IPT is used for image preprocessing techniques like noise removal, binarization, transformation, etc. and for some feature extraction techniques (eg. Crossing Number). It is also used for visualizing and analyzing the results of image preprocessing and feature extraction steps.

3.3 Data collection

As it is well known about fingerprints, it is possible to collect 20 different samples of fingerprint from the individual. The process of data collection is pretty simple the data from 6 persons are taken including phalanges as well as tarsal and also a database FVC2000 is taken for the completion of this research work.

3.3.1 Image Acquisition

The fingerprints are first taken as the inked impression of finger on the A4 paper, then images are acquired by scanning the fingerprint documents using digital scanners or by taking photograph of the fingerprint document using digital camera or by optical sensor. CANON LIDE scanner is used to scan the fingerprints from different persons. Each image is scanned at 300dpi in RGB color mode. Figure 3.1 shows the slicing of individual images from fingerprint samples and Figure 3.2 shows the final stage of the image acquisition.

Fig 3.1: Image for fingerprint document before slicing

Fig 3.2: Image for fingerprint document after slicing

3.3.2 Database Creation

Datasets are created by taking fingerprint samples from different persons. The overall procedure for creating fingerprint datasets is described in the Algorithm 3.1.

Algorithm 3.1 Handwritten Dataset Creation

1: Take fingerprint samples from different persons in A4 paper.

2: Scan the sample documents using digital scanner.

3: Slice and crop individual images from each documents.

4: Make a directory.

5: Drag individual image in the directory.

7: Datasets are ready.

3.4 System Overview

The fingerprint recognition system is divided into following three sub-sections, Preprocessing, Minutiae Extraction and Recognition as in figure 3.3

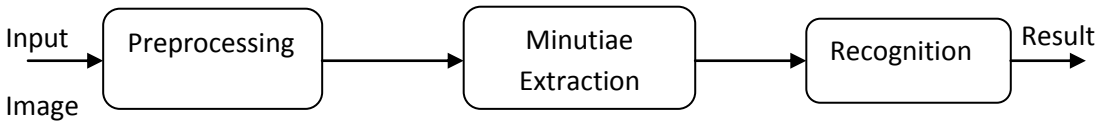


Figure 3.3: System overview

3.4.1 Image Preprocessing

Pre-processing is done prior to feature extraction algorithms. The raw images are subjected to a number of preliminary processing steps to make it usable in the descriptive stages of fingerprint analysis. Pre-processing aims to produce clean document images that are easy for the Recognition systems to operate accurately.

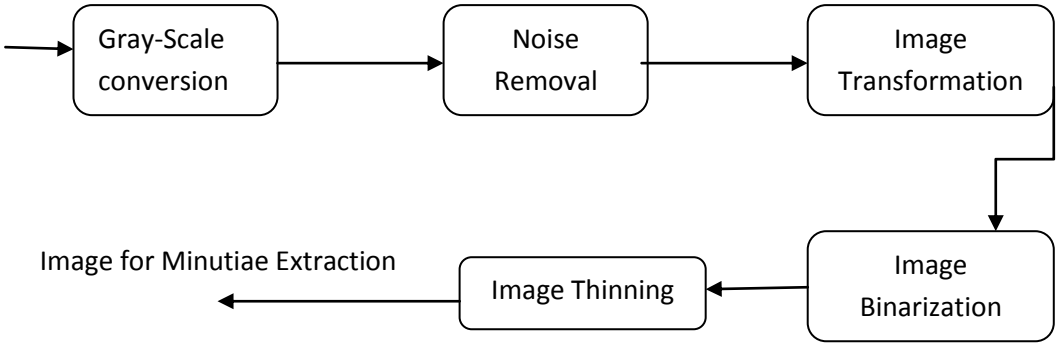


Figure 3.4: Preprocessing Steps

Algorithm 3.2 Image Preprocessing

1. Read image.
2. Convert RGB images to gray scale image.
3. Remove noise using median filter.
4. Transform the input image.
5. Binarize the input image.
6. Thinning of image is completed to obtain thinned ridges.

Following subsections describe each of these steps in detail. The organization of subsections are as follows: Color normalization method is described in section 3.4.1.1, noise removal is described in section 3.4.1.2, image transformation procedure is described in the section 3.4.1.3, image binarization is given in section 3.4.1.4, thinning technique is presented in the section 3.4.1.5.

3.4.1.1 RGB to Grayscale Conversion

An RGB colour image is an M × N × 3 array of colour pixels, where each colour pixel is a triplet corresponding to the red, green, blue component of the RGB image at a specific spatial location. The range of values is [0, 255] or [0, 65535] for RGB images of unit 8 or unit 16 respectively. Given 24 bit true colour RGB image is converted into 8 bit grayscale image. Grayscale component is calculated by taking weighted summation of R, G and B components of RGB image. Weights are selected same as the NTSC color space selects for the luminance i.e. the grayscale signal used to display pictures on monochrome televisions. For the RGB image *f(x,y)*, corresponding gray-scale image is given by,

$$g(x, y) = 0.2989*R(f(x, y)) + 0.5870*G(f(x, y)) + 0.1140*B(f(x, y)) \quad (3.1)$$

where, *R(f(x; y))* is red component of the RGB image f(x; y), and so on.

Figure 4.1 shows the RGB to grayscale conversion of input image.

Figure 3.5: Grayscale of input image

3.4.1.2 Noise Removal

Noise removal is one of the important steps of image preprocessing. Any unnecessary pixels are removed with the help of filtering. Median filter is an effective method of noise removal which can suppress isolated noise without blurring sharp edges. Median filter replaces a pixel with the median value of its neighborhood. For the digital image f(x, y), median filtered image is obtained as,

$$g(x, y) = \text{median}\{f(I, j) \mid (I, j) \in \omega\} \quad (3.2)$$

where, w is the neighborhood centered around location (x, y) in the image. Figure 3.6 shows the denoised image corresponding to given gray-scaled image.

Figure 3.6: Image after noise removal

3.4.1.3 Image Transformation

The Gray-scale image is further divided into small processing blocks (32 by 32 pixels) and Fourier transform is performed according to:

$$F(u,v) = \sum_{x-0}^{M-1} \sum_{y-0}^{N-1} f(x,y) \times exp\left\{-j2\pi \times \left(\frac{ux}{M} + \frac{vy}{N}\right)\right\} \quad (3.3)$$

for u = 0, 1, 2, ..., 31 and v = 0, 1, 2, ..., 31.

In order to enhance a specific block by its dominant frequencies, FFT of the block is multiplied by its magnitude a set of times. Where the magnitude of the original FFT = abs(F(u,v)) = |F(u,v)|.

Get the enhanced block according to

$$g(x,y) = F^{-1}\{F(u,v) \times |F(u,v)|^{k}\} \quad (3.4)$$

where $F^{-1}(F(u,v))$ is done by:

$$f(x,y) = \frac{1}{MN}\sum_{x-0}^{M-1}\sum_{y-0}^{N-1} F(u,v) \times exp\left\{j2\pi \times \left(\frac{ux}{M} + \frac{vy}{N}\right)\right\} \quad (3.5)$$

for x = 0, 1, 2, ..., 31 and y = 0, 1, 2, ..., 31.

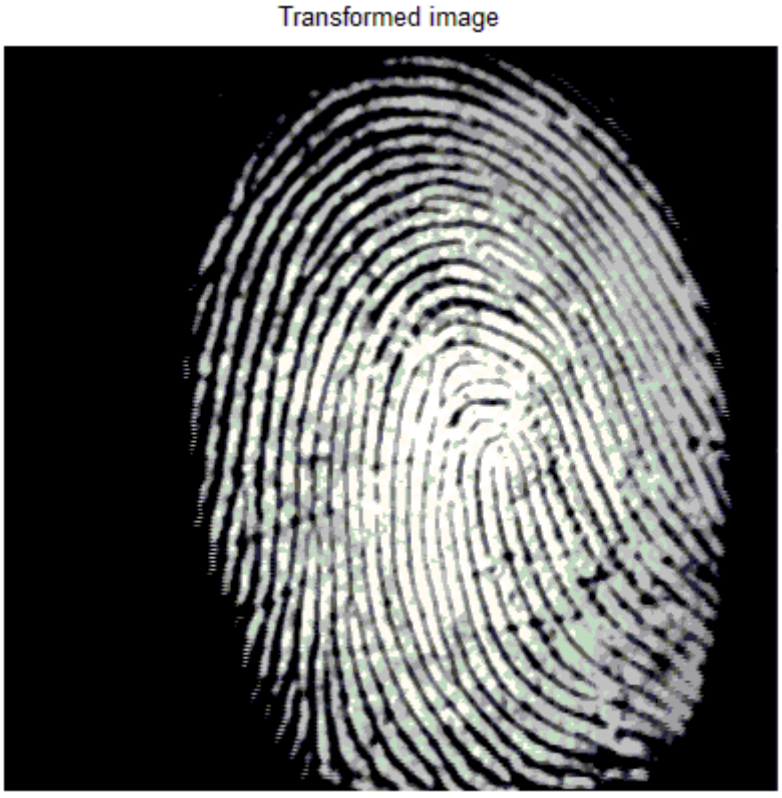


Figure 3.7: Transformed Image (FFT)

The k in formula (3.4) is an experimentally determined constant, which is chose as k= 0.45 to calculate. While having a higher "k" improves the appearance of the ridges, filling up small holes in ridges, having too high a "k" can result in false joining of ridges. Thus a termination might become a bifurcation. Figure 3.7 presents the image after FFT enhancement.

The enhanced image after FFT has the improvements to connect some falsely broken points on ridges and to remove some spurious connections between ridges. The side effect of each block is obvious but it has no harm to the further operations because it is found that the image after consecutive binarization operation is pretty good as long as the side effect is not too severe.

3.4.1.4 Image Binarization

Fingerprint Image Binarization is to transform the 8-bit Gray fingerprint image to a 1-bit image with 0-value for ridges and 1-value for furrows. After the operation, ridges in the fingerprint are highlighted with black color while furrows are white.

A locally adaptive binarization method is performed to binarize the fingerprint image. Such a named method comes from the mechanism of transforming a pixel value to 1 if the value is larger than the mean intensity value of the current block (16x16) to which the pixel belongs [Figure 3.8].

Algorithm 3.3 Image Binarization

1: Compute histogram and probability of each intensity level i , i = 1 _ _ _ L.

2: Compute between class variance for black and white pixels, by taking threshold

t = 1 --- L,

3: $\sigma^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2$

where,

4: $\omega_1(t) = \sum_1^t p(i)$ is the class probability for black pixels,

5: $\omega_2(t) = \sum_{t+1}^L p(i)$ is the class probability for white pixels,

6: $\mu_1(t) = \sum_1^t \frac{ip(i)}{\omega_1(t)}$ is the class mean for black pixels,

7: $\mu_2(t) = \sum_{t+1}^L \frac{ip(i)}{\omega_2(t)}$ is the class mean for white pixels,

8: and, p(i) is the probability of intensity level i.

9: Select t for which $\sigma_b^2(t)$ is maximum.

10: Finally, binarize the grayscale image f(x, y) using the threshold t as,

11: $$g(x,y) = \begin{cases} 1 \; if \; f(x,y) \geq t \\ 0 \; if \; f(x,y) < t \end{cases}$$

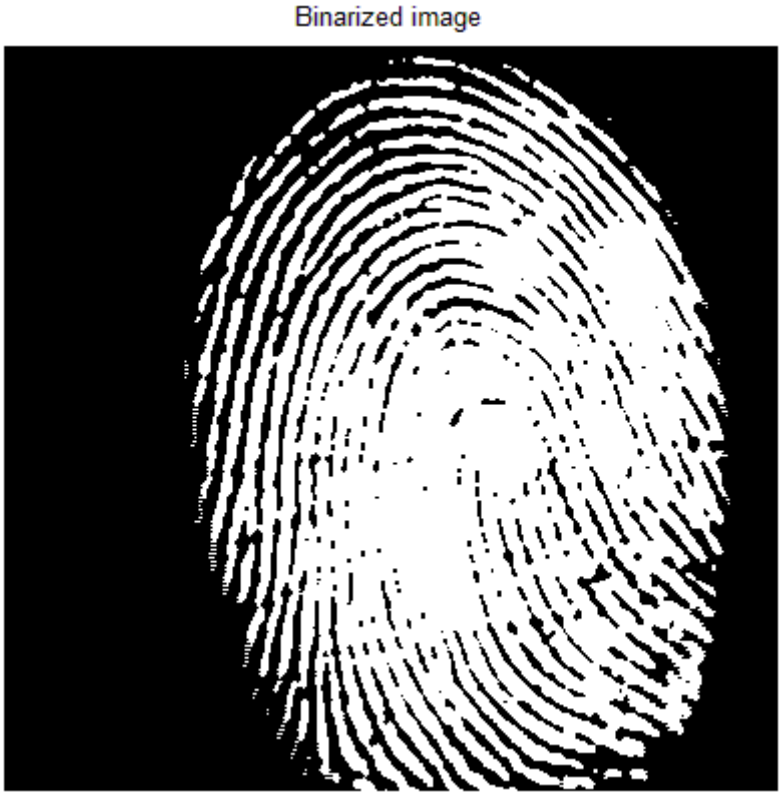


Figure 3.8 Binarized image

3.4.1.5 Image Thinning

The approach iteratively deletes edge points from the region until just the skeleton remains. For a point to be deleted, the following conditions must hold

1. The point is not an endpoint.
2. The removal does not break the connectedness of the skeleton.
3. The removal does not cause excessive erosion of the region.

The method was first suggested by Zhang and Suen in 1984. The method consists of successive passes of two basics steps that are applied to the contour points of the region. A contour point p1 is defined to have at least one pixel in its 8-pixel neighborhood that belongs to the background. The 8-pixel neighborhood is shown in Figure 3.9. A contour point is marked for deletion in the first pass if all of the following conditions hold for its 8-pixel neighborhood

a) The number of neighbors that belong to the region must be between 2 and 6.
b) All neighbors that belong to the background must be connected.
c) p2, p4 or p6 must belong to the background.
d) p4, p6 or p8 must belong to the background.

If all conditions are met the point is marked for deletion. However, the point is not deleted until all region points are processed to ensure that the data structure is not changed during the execution of the step.

For a point to be marked for deletion in the second pass of the algorithm the first two conditions still must hold, but condition (b) and (c) are changed as follows

(c') p2, p4 or p8 must belong to the background.

(d') p2, p6 or p8 must belong to the background.

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Figure 3.9: The 8-pixel neighborhood used in thinning algorithm.

As shown in Figure 3.10, the algorithm produces a well centered 1-pixel wide ridge skeleton when executed on a binary fingerprint image.

Thinning also removes some H breaks, isolated points and spikes.

3.4.2 Minutiae Extraction

After ridge thinning, marking minutia points is relatively easy. But it is still not a trivial task as most literatures declared because at least one special case evokes my caution during the minutia marking stage. Minutiae extraction step plays one of the most important roles in the recognition. It is the heart of recognition system. Good minutiae set should contain maximum number of minutiae extracted from the sample image. Here Binarization Based Minutiae Extraction technique is used.

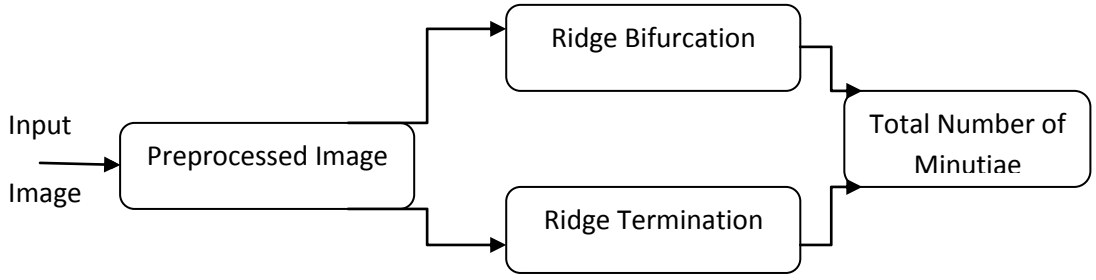


Figure 3.10: Minutiae Extraction Steps

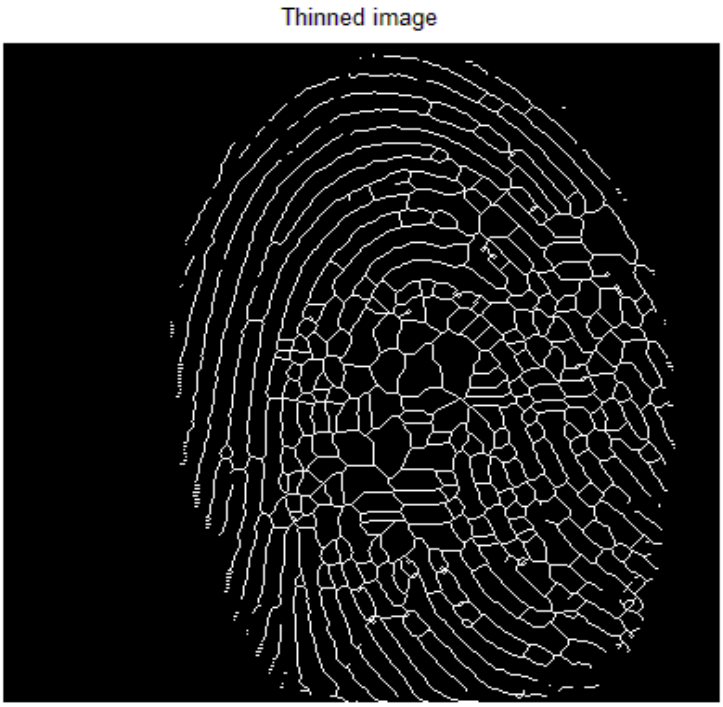


Figure 3.11: Thinned image.

In general, for each 3x3 window, if the central pixel is 1 and has exactly 3 one-value neighbors, then the central pixel is a ridge branch [Figure 3.12]. If the central pixel is 1 and has only 1 one-value neighbor, then the central pixel is a ridge ending [Figure3.13].

0	1	0
0	1	0
1	0	1

Figure 3.12Bifurcation

0	0	0
0	1	0
0	0	1

Figure3.13Termination

0	1	0
0	1	1
1	0	0

Figure3.14Triple counting branch

Figure 3.14 illustrates a special case that a genuine branch is triple counted. Suppose both the uppermost pixel with value 1 and the rightmost pixel with value 1 have another neighbor outside the 3x3 window, so the two pixels will be marked as branches too.

The minutiae extraction for FRMSM is completed here, whereas for FRSFM genetic algorithm for spur minutiae removal is done.

Also the average inter-ridge width D is estimated at this stage. The average inter-ridge width refers to the average distance between two neighboring ridges. The way to approximate the D value is simple. Scan a row of the thinned ridge image and sum up all pixels in the row whose value is one. Then divide the row length with the above summation to get an inter-ridge width. For more accuracy, such kind of row scan is performed upon several other rows and column scans are also conducted, finally all the inter-ridge widths are averaged to get the D.

Together with the minutia marking, all thinned ridges in the fingerprint image are labeled with a unique ID for further operation. The labeling operation is realized by using the Morphological operations. Figure 3.15 shows the context after minutiae marking.

4.2.1 False Minutiae Removal

The preprocessing stage does not totally heal the fingerprint image. For example, false ridge breaks due to insufficient amount of ink and ridge cross-connections due to over inking are not totally eliminated. Actually all the earlier stages themselves occasionally introduce some artifacts which later lead to spurious minutia. These false minutiae will significantly affect the accuracy of matching if they are simply regarded as genuine minutia. So some mechanisms of removing false minutia are essential to keep the fingerprint verification system effective.

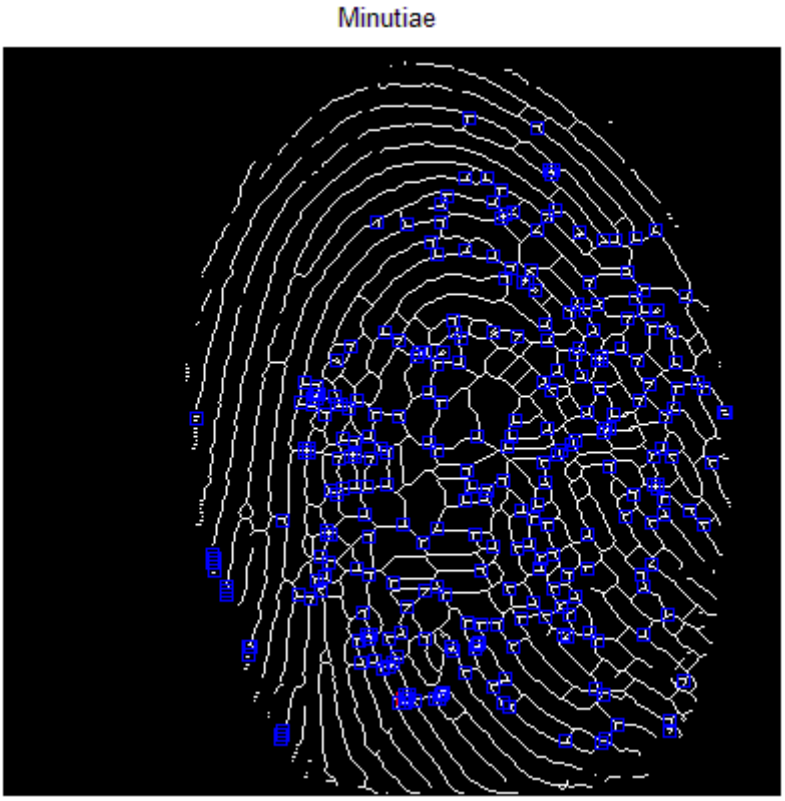


Figure 3.15: Image after minutiae extraction

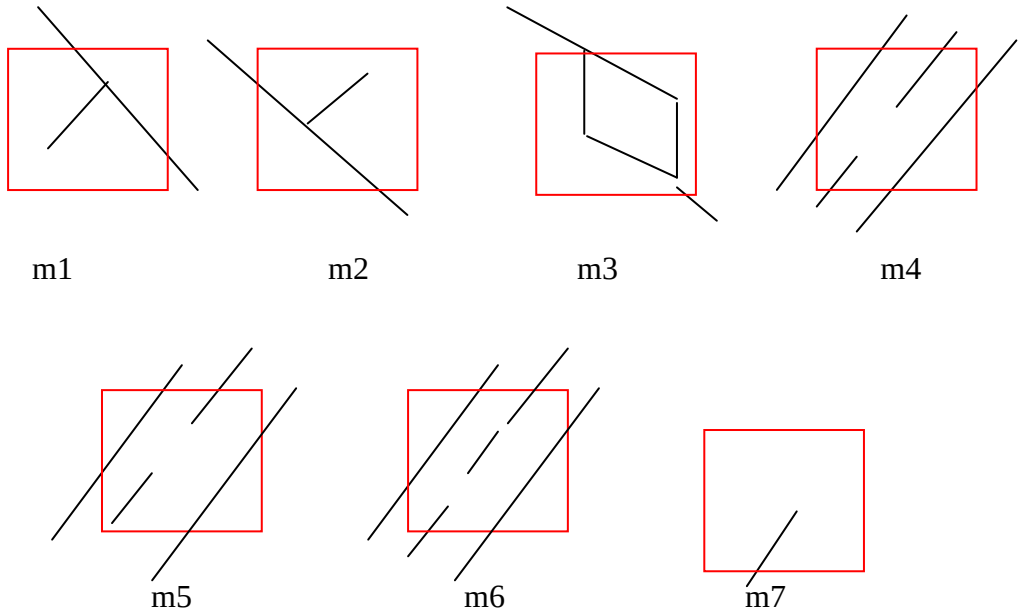


Figure3.16: False Minutia Structures. [Wuh02]

Figure 3.16 shows seven different types of false minutiae. m_1 is a spike piercing into a valley. In the m_2 case a spike falsely connects two ridges. m_3 has two near bifurcations located in the same ridge. The two ridge broken points in the m_4 case

have nearly the same orientation and a short distance. m_5 is alike the m4 case with the exception that one part of the broken ridge is so short that another termination is generated. m_6 extends the m_4 case but with the extra property that a third ridge is found in the middle of the two parts of the broken ridge. m_7 has only one short ridge found in the threshold window.

[Rat95] Only handles the case m_1, m_4, m_5 and m_6. [Mai97] have not false minutia removal by simply assuming the image quality is fairly good. [Hal99] has not a systematic healing method to remove those spurious minutias although it lists all types of false minutia shown in Figure 3.16 except the m_3 case.

Procedures in removing false minutia are:

1. If the distance between one bifurcation and one termination is less than D and the two minutia are in the same ridge(m_1 case) . Remove both of them. Where D is the average inter-ridge width representing the average distance between two parallel neighboring ridges.
2. If the distance between two bifurcations is less than D and they are in the same ridge, remove the two bifurcations. (m_2, m_3 cases).
3. If two terminations are within a distance D and their directions are coincident with a small angle variation. And they suffice the condition that no any other termination is located between the two terminations. Then the two terminations are regarded as false minutia derived from a broken ridge and are removed. (case m_4,m_5, m_6).
4. If two terminations are located in a short ridge with length less than D, remove the two terminations (m_7).

This proposed procedure in removing false minutia has two advantages. One is that the ridge ID is used to distinguish minutia and the seven types of false minutia are strictly defined comparing with those loosely defined by other methods. The second advantage is that the order of removal procedures is well considered to reduce the computation complexity. It surpasses the way adopted by [Mai97] that does not utilize the relations among the false minutia types. For example, the procedure 3 solves the m_4, m_5 and m_6 cases in a single check routine. And after procedure 3, the number of false minutia satisfying the m_7 case is significantly reduced.

3.4.3 Recognition

The process of identification or verification is because of recognition engine. i.e. either the fingerprint matches or mis-matches is determined with the help of it. Here the recognition engine of the system consist of two minutiae based algorithms implemented in it. The system carries out the matching process with a Boolean result if matches the result is true with value between zero and one else the result is false. The overview of recognition engine is shown in figure 3.17.

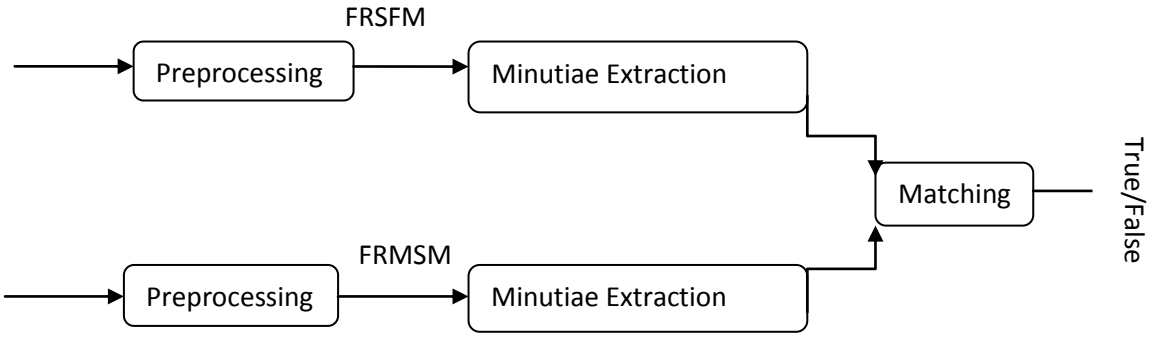


Figure3.17: Recognition Process

3.4.3 .1 Minutiae Matching

Given two set of minutia of two fingerprint images, the minutia match algorithm determines whether the two minutia sets are from the same finger or not.

An alignment-based match algorithm partially derived from the [Hon98] is used here. It includes two consecutive stages: one is alignment stage and the second is match stage.

1. Alignment stage. Given two fingerprint images to be matched, choose any one minutia from each image; calculate the similarity of the two ridges associated with the two referenced minutia points. If the similarity is larger than a threshold, transform each set of minutia to a new coordination system whose origin is at the referenced point and whose x-axis is coincident with the direction of the referenced point.

2. Match stage: After obtaining two set of transformed minutia points, the elastic match algorithm is used to count the matched minutia pairs by assuming two minutia having nearly the same position and direction are identical.

3.4.3.1.1 Alignment Stage

The ridge associated with each minutia is represented as a series of x-coordinates (x_1, $x_2 \ldots x_n$) of the points on the ridge. A point is sampled per ridge length L starting from the minutia point, where the L is the average inter-ridge length. And n is set to 10 unless the total ridge length is less than 10*L.

So the similarity of correlating the two ridges is derived from:

$$S = \sum^{m}_{i=0} x_i X_i / [\sum^{m}_{i=0} x_i^2 X_i^2]^{\wedge}0.5 \qquad (3.6)$$

where ($x_i \sim x_n$) and ($X_i \sim X_N$) are the set of minutia for each fingerprint image respectively. And m is minimal one of the n and N value. If the similarity score is larger than 0.8, then the next step is executed else continue to match the next pair of ridges.

For each fingerprint, translate and rotate all other minutia with respect to the reference minutia according to the following formula:

$$\begin{pmatrix} xi_{new} \\ yi_{new} \\ \theta i_{new} \end{pmatrix} = \mathrm{TM} * \begin{pmatrix} (\mathrm{xi} - \mathrm{x}) \\ (\mathrm{yi} - \mathrm{y}) \\ (\theta \mathrm{i} - \theta) \end{pmatrix} \qquad (3.7)$$

where (x, y, θ) is the parameters of the reference minutia, and TM is

$$TM = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (3.8)$$

3.4.3.1.2 Match Stage

The matching algorithm for the aligned minutia patterns needs to be elastic since the strict match requiring that all parameters (x, y, θ) are the same for two identical minutia is impossible due to the slight deformations and inexact quantization of minutiae.

The elastic match minutia is achieved by placing a bounding box around each template minutia. If the minutia to be matched is within the rectangle box and the direction discrepancy between them is very small, then the two Minutiae are regarded as a matched minutia pair. Each minutia in the template image either has no matched minutia or has only one corresponding minutia.

The final match ratio for two fingerprints is the number of total matched pair over the number of minutia of the template fingerprint. The score is 10*ratio and ranges from 0 to 1. If the score is larger than a pre-specified threshold, the two fingerprints are from th`e same finger.

Chapter 4

DISCUSSION

4.1 Experimentation and Results

The aim of this research work is to perform the comparative analysis of SFM and MSM fingerprint recognition algorithms on various bases. Experiment is carried on with FVC2000 database. The experimentation for various factors of fingerprint recognition is carried out several times for both the recognition engines. The efficiency, accuracy, time invariant, FMR and FNMR are the average of all the experimentation results for the dataset. The comparative results for SFM and MSM are described in this chapter with a detailed summary.

4.1.1 Analysis

Under this section the efficiency or recognition system is analyzed, in terms of time and matching ratio. This section is further divided into 2 subsections. Subsection 4.1.1.1 deals with accuracy of the system where as subsection 4.1.1.2 deals with time invariants.

4.1.1.1 Accuracy Analysis

The False Rejection Rate (FRR - portion of matching pairs resulting in false rejection) and the other is False Acceptance Rate (FAR - portion of non-matching pairs resulting in false acceptance) together categorize the efficiency as well as accuracy of the recognition system for a given decision threshold. It is found that varying the threshold trades FAR off against FRR as in

Figure: 5.1. In more general framework, two errors can be expressed as False Matching Ratio (FMR) and False Non-Matching Ratio (FNMR).

The Equal Error Ratio (EER) corresponds to a point at some threshold, where FRR = FAR.

FNMR- derives the number of times; someone who should be identified positively is instead rejected.

(%) FNMR = (FR/N)*100

FR=number of incidents of false rejections

Threshold Value	FRFSM	FRMSM
0.5	0.050%	0.06%
0.6	0	0
0.7	0	0.0001%
0.8	0.005%	0.009%
0.9	0.025%	0.029%
1.0	0.035%	0.045%

Table 4.1: False Non-Matching Ratio

FMR-describes the number of times, fingerprint is inaccurately positively matched.

(%) FMR = (FA/N)*100

FA= number of incidents of false acceptance

N=total number of samples

Threshold Value	FRFSM	FRMSM
0.5	0.070%	0.08%
0.6	0	0
0.7	0	0.0001%
0.8	0.0008%	0.0075%
0.9	0.002%	0.024%
1.0	0.005%	0.058%

Table 4.2: False Matching Ratio

Table 4.1 shows performance matrices for FRSFM and FRMSM for fingerprint recognition systems on various datasets. The best recognition result is obtained with FRSFM based recognition. The details of false non-matching ratio and false matching ratio are explained in tables 4.1.

Figure 4.1 shows the recognition system accuracy based on FMR and FNMR.

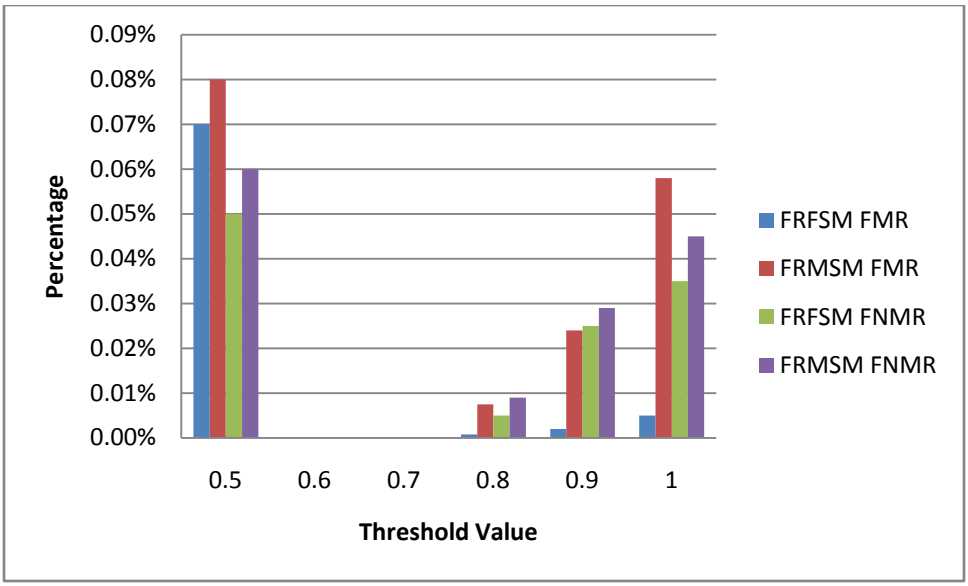


Figure 4.1: FNMR and FMR for FRSFM and FRMSM

4.1.1.2 Time Analysis

The time taken by each of the algorithms namely FRSFM and FRMSM, under different threshold values is shown in the Table 4.3 and Table 4.4. The timing constraint is further divided into two parts for better analysis of the results, the time taken for Minutiae extraction and the time taken for matching the fingerprint.

The table 4.3 describes the time factor for minutiae extraction under various threshold values for both of the algorithms named FRSFM and FRMSM, whereas Table 4.4 describes the time variation for matching the fingerprints using both algorithms under different threshold values.

Threshold value	FRSFM Minutiae Extraction Time (Seconds) For Images										FRMSM Minutiae Extraction Time (Seconds) For Images									
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
0.5	1.234065	1.270809	1.322300	1.457085	2.116754	2.053125	2.015058	1.929770	2.172303	2.164321	1.134052	1.170800	1.192230	1.449082	1.916659	1.854127	1.814058	1.729776	1.872407	1.784320
0.55	1.334054	1.470811	1.392133	1.449589	1.656657	1.738145	1.492820	1.470151	1.810381	1.930101	1.034052	1.170800	1.103056	1.409082	1.876659	1.838145	1.749188	1.777152	1.853382	1.730100
0.6	1.490736	1.454740	1.692285	1.377828	1.595102	1.463766	1.396397	1.375210	1.522239	1.76301	1.004746	1.104746	1.003283	1.157828	1.590102	1.703767	1.636397	1.595209	1.624239	1.605301
0.65	1.548380	1.439866	1.513290	1.292813	1.208490	1.362334	1.406184	1.409308	1.431100	1.040000	0.965089	0.997861	0.981291	0.970803	1.008490	1.262330	1.306182	1.109308	1.131124	1.340059
0.7	1.462987	1.670105	1.616371	1.900512	1.791497	2.044606	2.186160	2.379005	2.400000	2.400130	1.262	1.272105	1.212371	1.500512	1.791497	1.844606	1.786160	1.792050	1.997442	1.838139
0.75	1.431145	1.423035	1.510312	1.746334	2.016300	2.208795	2.338026	2.407597	2.501901	2.489456	1.251149	1.223047	1.210312	1.446334	1.816311	1.848796	1.838027	1.875986	1.871903	1.849476
0.8	1.513146	1.495362	1.657345	1.759090	2.008925	2.140029	2.209591	2.308415	2.093769	2.456778	1.217141	1.215361	1.259347	1.456091	1.848929	1.840029	1.919591	1.798418	1.873760	1.856774
0.85	1.500855	1.580080	2.505643	2.087161	2.554202	2.454402	2.607037	2.697936	2.562774	2.700003	1.200845	1.180084	1.205640	1.487161	1.754237	1.854437	1.907037	1.897936	1.862779	1.792807
0.9	1.964576	2.008411	2.421717	2.170600	2.564066	2.417594	2.638211	2.075554	2.775740	2.800592	1.264576	1.258411	1.217175	1.470634	1.764066	1.807592	1.838255	1.855549	1.895743	1.893592

0.95	2.190062	2.322995	2.554200	2.454407	2.607037	2.697936	2.562774	2.700007	2.854200	2.904407	1.210062	1.322995	1.228160	1.560421	1.753375	1.893470	2.116264	1.925548	1.961308	1.908354
1.0	2.247822	2.232881	2.454407	2.600913	2.697936	2.566708	2.710440	2.834109	2.902410	1.860436	1.237854	1.238883	1.242835	1.519124	1.748801	1.886699	1.841304	1.843237	1.870960	1.860436

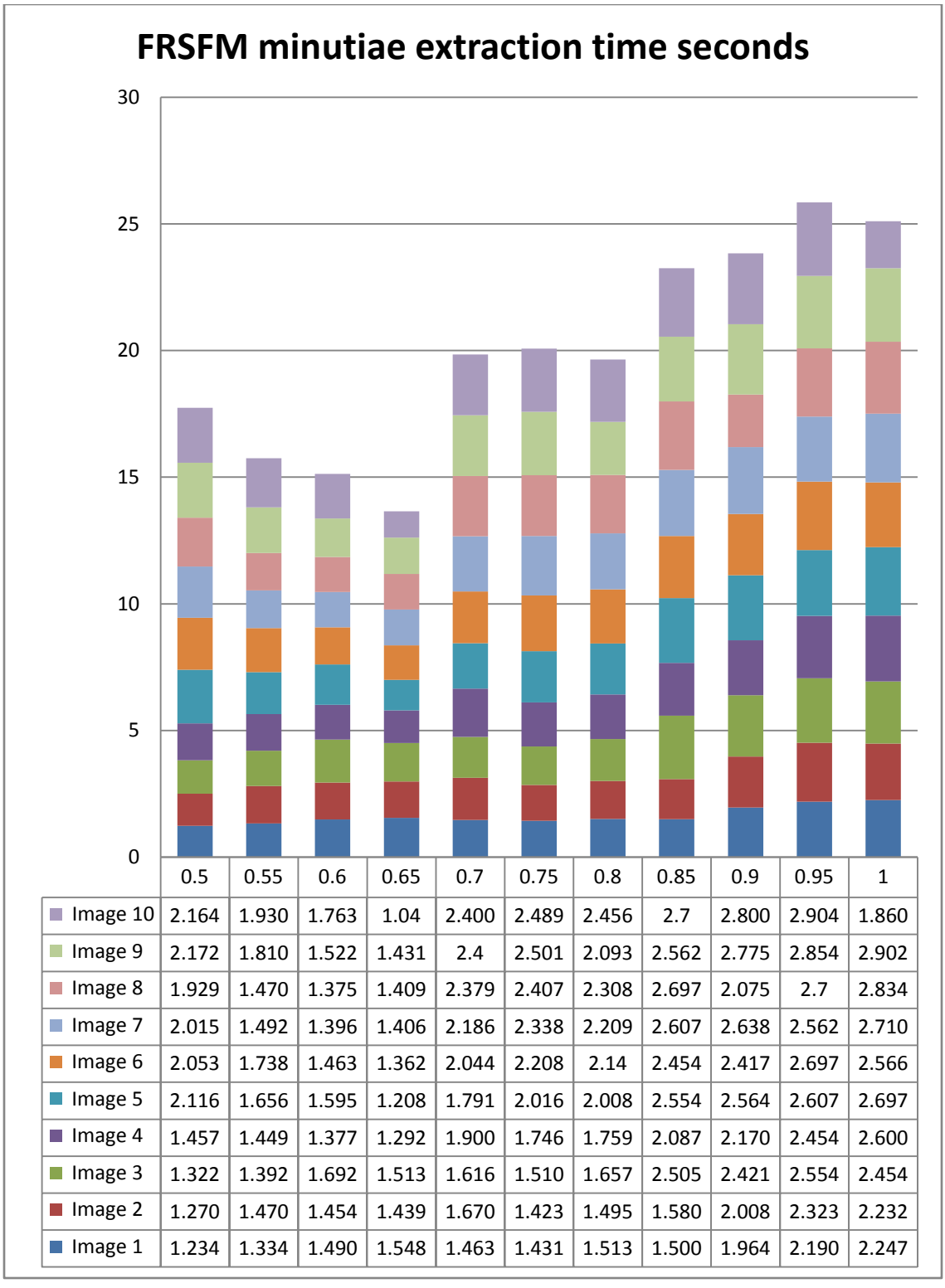
Table 4.3: Minutiae Extraction Time Variants

Threshold value	**FRSFM Minutiae Matching Time (Seconds) For Images**										**FRMSM Matching Time (Seconds) For Images**									
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
0.5	1.450032	1.321021	1.150320	1.240792	1.301910	1.197645	1.208756	1.312405	1.175694	1.097824	2.290057	2.370263	2.300245	2.270263	2.260250	2.240241	2.310252	2.370631	2.870453	2.370244
0.55	1.410865	1.294103	1.113052	1.463027	1.284039	1.105938	1.358100	1.330021	1.199604	1.120330	4.082834	3.391150	4.058050	3.064609	3.391150	3.64904	2.884682	3.198263	3.822147	3.278106
0.6	1.382507	1.319433	1.108829	1.440026	1.279217	1.109880	1.389059	1.327091	1.166073	1.159034	2.282881	2.338794	2.282168	2.163399	2.134641	2.141433	2.041447	2.173382	2.352131	2.601530
0.65	1.421759	1.300967	1.088954	1.251288	1.290014	1.104933	1.451142	1.392006	1.201104	1.179022	2.720247	2.430936	2.603640	2.681394	2.971125	2.895616	2.592858	2.670379	2.871678	2.750209
0.7	1.409950	1.324503	1.110066	1.277041	1.321028	1.124013	1.450032	1.380031	1.210945	1.183019	3.651044	3.779774	3.820712	3.860764	3.529675	3.743406	3.031367	2.350538	2.173448	2.163400

1.0	0.95	0.9	0.85	0.8	0.75
1.000348	1.450032	1.387002	1.420567	1.432081	1.446093
0.998407	0.986590	0.997821	1.275610	1.301190	1.321021
1.023490	1.150320	1.149870	1.304578	1.290219	1.270459
1.302145	1.304597	1.392752	1.450032	1.469038	1.439032
1.098312	1.084120	1.321021	1.332077	1.324915	1.372981
1.002340	0.986509	0.975698	1.112067	1.130972	1.158270
1.113902	1.130854	1.277552	1.197402	1.270418	1.430457
1.012459	1.207845	1.309633	1.308041	1.309948	1.316915
1.121315	1.210754	1.330751	1.138759	1.150320	1.163844
1.010331	1.150320	1.340096	1.199883	1.208718	1.227903
3.709299	3.519427	3.691570	3.253642	3.825973	3.663451
3.323564	3.759249	2.849614	3.408243	3.890621	3.919709
3.847176	3.473487	3.819054	4.085256	3.508213	3.222789
3.484227	3.031626	3.804841	4.144399	3.989405	3.148725
3.324759	3.740090	4.095408	3.285450	3.563222	3.691992
2.486084	3.384277	3.938370	2.409661	2.592019	3.759494
2.188132	3.736332	4.093288	2.451676	3.255630	3.787873
3.073413	3.451395	3.588226	3.644701	2.779167	3.922889
3.042926	3.792538	3.601939	2.911459	3.707895	3.937403
3.667597	3.769910	4.003870	3.918776	3.646160	3.651754

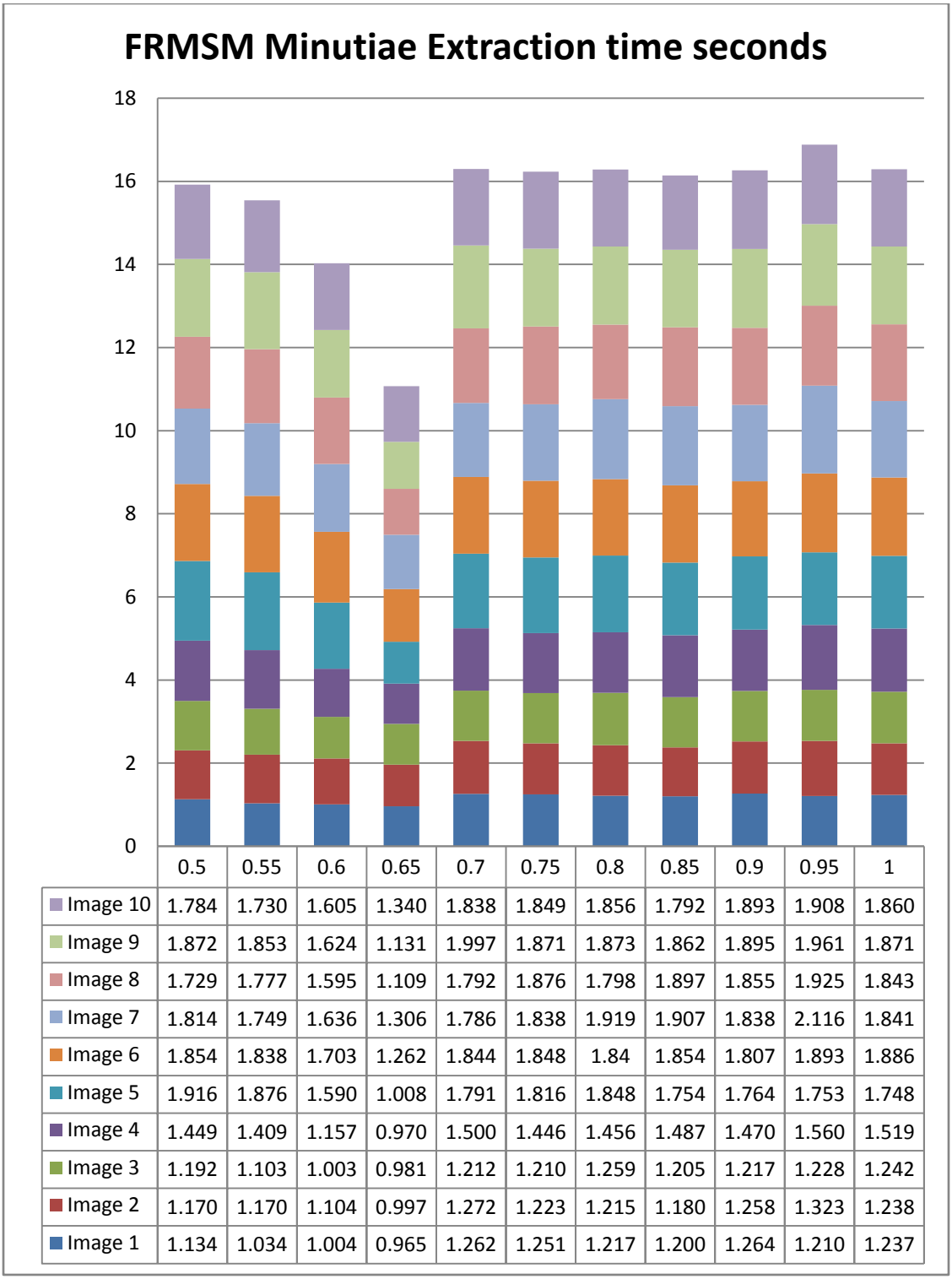
Table 4.4: Matching Time Variants

Figure 4.2, 4.3, 4.4, 4.5 shows the inter relationship between different threshold values along with time taken for completion of the operation.



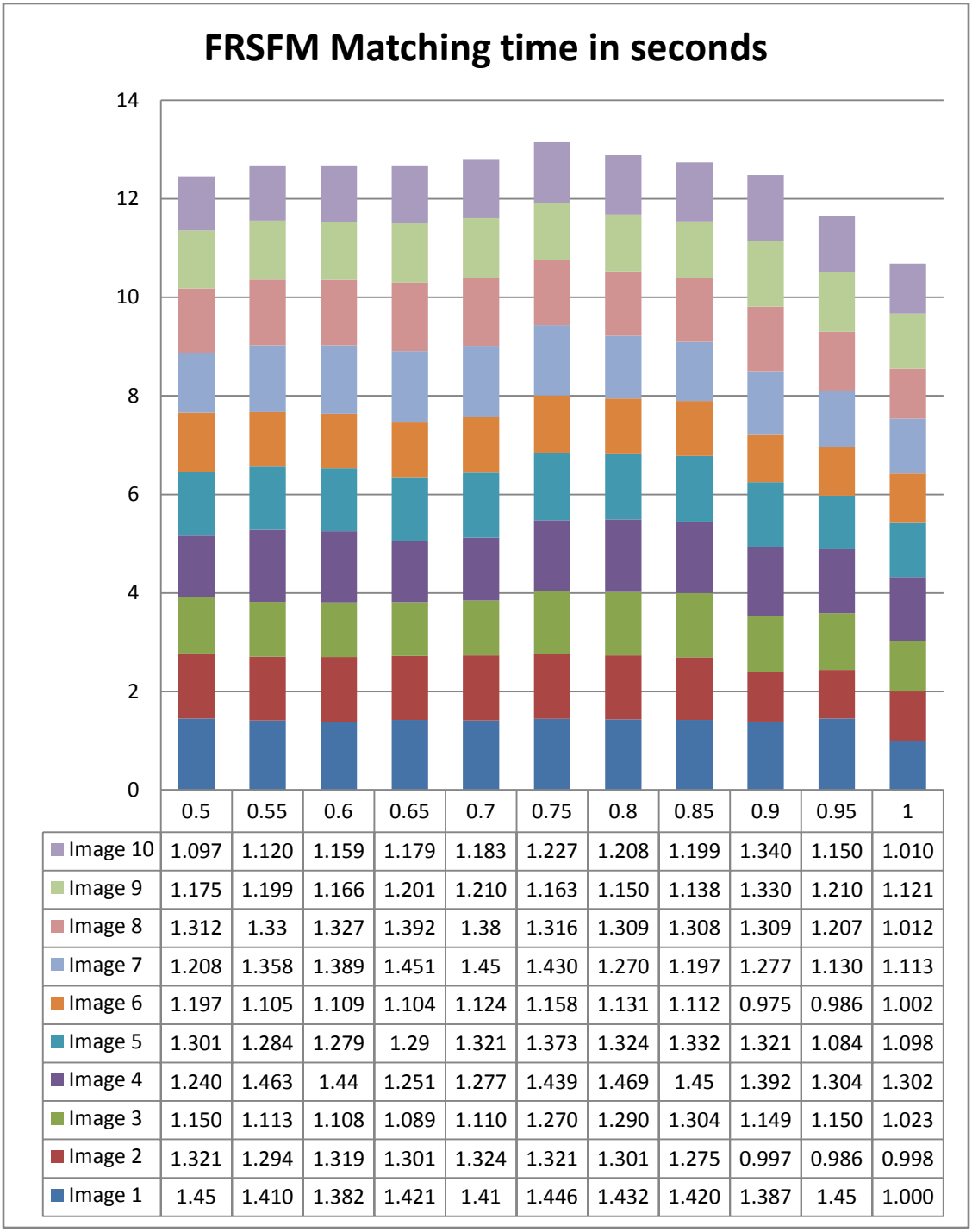
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
Image 10	2.164	1.930	1.763	1.04	2.400	2.489	2.456	2.7	2.800	2.904	1.860
Image 9	2.172	1.810	1.522	1.431	2.4	2.501	2.093	2.562	2.775	2.854	2.902
Image 8	1.929	1.470	1.375	1.409	2.379	2.407	2.308	2.697	2.075	2.7	2.834
Image 7	2.015	1.492	1.396	1.406	2.186	2.338	2.209	2.607	2.638	2.562	2.710
Image 6	2.053	1.738	1.463	1.362	2.044	2.208	2.14	2.454	2.417	2.697	2.566
Image 5	2.116	1.656	1.595	1.208	1.791	2.016	2.008	2.554	2.564	2.607	2.697
Image 4	1.457	1.449	1.377	1.292	1.900	1.746	1.759	2.087	2.170	2.454	2.600
Image 3	1.322	1.392	1.692	1.513	1.616	1.510	1.657	2.505	2.421	2.554	2.454
Image 2	1.270	1.470	1.454	1.439	1.670	1.423	1.495	1.580	2.008	2.323	2.232
Image 1	1.234	1.334	1.490	1.548	1.463	1.431	1.513	1.500	1.964	2.190	2.247

Figure 4.2 FRSFM Minutiae Extraction Time



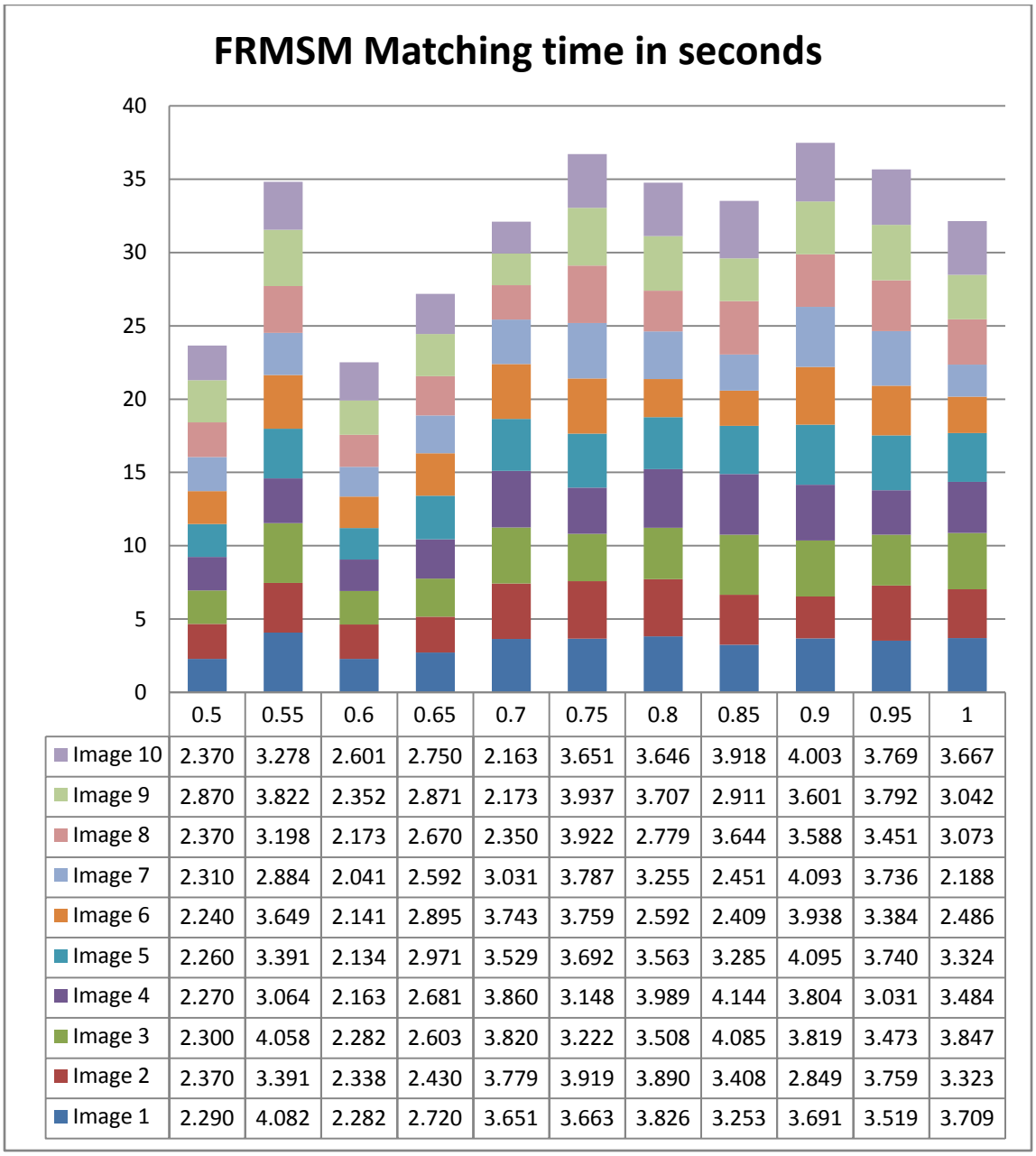
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
Image 10	1.784	1.730	1.605	1.340	1.838	1.849	1.856	1.792	1.893	1.908	1.860
Image 9	1.872	1.853	1.624	1.131	1.997	1.871	1.873	1.862	1.895	1.961	1.871
Image 8	1.729	1.777	1.595	1.109	1.792	1.876	1.798	1.897	1.855	1.925	1.843
Image 7	1.814	1.749	1.636	1.306	1.786	1.838	1.919	1.907	1.838	2.116	1.841
Image 6	1.854	1.838	1.703	1.262	1.844	1.848	1.84	1.854	1.807	1.893	1.886
Image 5	1.916	1.876	1.590	1.008	1.791	1.816	1.848	1.754	1.764	1.753	1.748
Image 4	1.449	1.409	1.157	0.970	1.500	1.446	1.456	1.487	1.470	1.560	1.519
Image 3	1.192	1.103	1.003	0.981	1.212	1.210	1.259	1.205	1.217	1.228	1.242
Image 2	1.170	1.170	1.104	0.997	1.272	1.223	1.215	1.180	1.258	1.323	1.238
Image 1	1.134	1.034	1.004	0.965	1.262	1.251	1.217	1.200	1.264	1.210	1.237

Figure 4.3: FRMSM Minutiae Extraction Time



	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
Image 10	1.097	1.120	1.159	1.179	1.183	1.227	1.208	1.199	1.340	1.150	1.010
Image 9	1.175	1.199	1.166	1.201	1.210	1.163	1.150	1.138	1.330	1.210	1.121
Image 8	1.312	1.33	1.327	1.392	1.38	1.316	1.309	1.308	1.309	1.207	1.012
Image 7	1.208	1.358	1.389	1.451	1.45	1.430	1.270	1.197	1.277	1.130	1.113
Image 6	1.197	1.105	1.109	1.104	1.124	1.158	1.131	1.112	0.975	0.986	1.002
Image 5	1.301	1.284	1.279	1.29	1.321	1.373	1.324	1.332	1.321	1.084	1.098
Image 4	1.240	1.463	1.44	1.251	1.277	1.439	1.469	1.45	1.392	1.304	1.302
Image 3	1.150	1.113	1.108	1.089	1.110	1.270	1.290	1.304	1.149	1.150	1.023
Image 2	1.321	1.294	1.319	1.301	1.324	1.321	1.301	1.275	0.997	0.986	0.998
Image 1	1.45	1.410	1.382	1.421	1.41	1.446	1.432	1.420	1.387	1.45	1.000

Figure 4.4 FRSFM Matching time



	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
Image 10	2.370	3.278	2.601	2.750	2.163	3.651	3.646	3.918	4.003	3.769	3.667
Image 9	2.870	3.822	2.352	2.871	2.173	3.937	3.707	2.911	3.601	3.792	3.042
Image 8	2.370	3.198	2.173	2.670	2.350	3.922	2.779	3.644	3.588	3.451	3.073
Image 7	2.310	2.884	2.041	2.592	3.031	3.787	3.255	2.451	4.093	3.736	2.188
Image 6	2.240	3.649	2.141	2.895	3.743	3.759	2.592	2.409	3.938	3.384	2.486
Image 5	2.260	3.391	2.134	2.971	3.529	3.692	3.563	3.285	4.095	3.740	3.324
Image 4	2.270	3.064	2.163	2.681	3.860	3.148	3.989	4.144	3.804	3.031	3.484
Image 3	2.300	4.058	2.282	2.603	3.820	3.222	3.508	4.085	3.819	3.473	3.847
Image 2	2.370	3.391	2.338	2.430	3.779	3.919	3.890	3.408	2.849	3.759	3.323
Image 1	2.290	4.082	2.282	2.720	3.651	3.663	3.826	3.253	3.691	3.519	3.709

Figure 4.5 FRMSM Matching Time

Table 4.5 describes the matching score of FRSFM and FRMSM with different threshold values. Also figure 4.6 shows the graphical representation of matching scores with threshold values for both of the algorithms.

Threshold Value	FRSFM Matching Scores	FRMSM Matching Scores
0.5	0.83	0.75
0.55	0.9	0.84
0.6	0.97	0.93
0.65	0.92	0.87
0.7	0.87	0.85
0.75	0.86	0.82
0.8	0.83	0.79
0.85	0.82	0.77
0.9	0.81	0.81
0.95	0.79	0.78
1	0.73	0.7

Table 4.5 Matching Scores

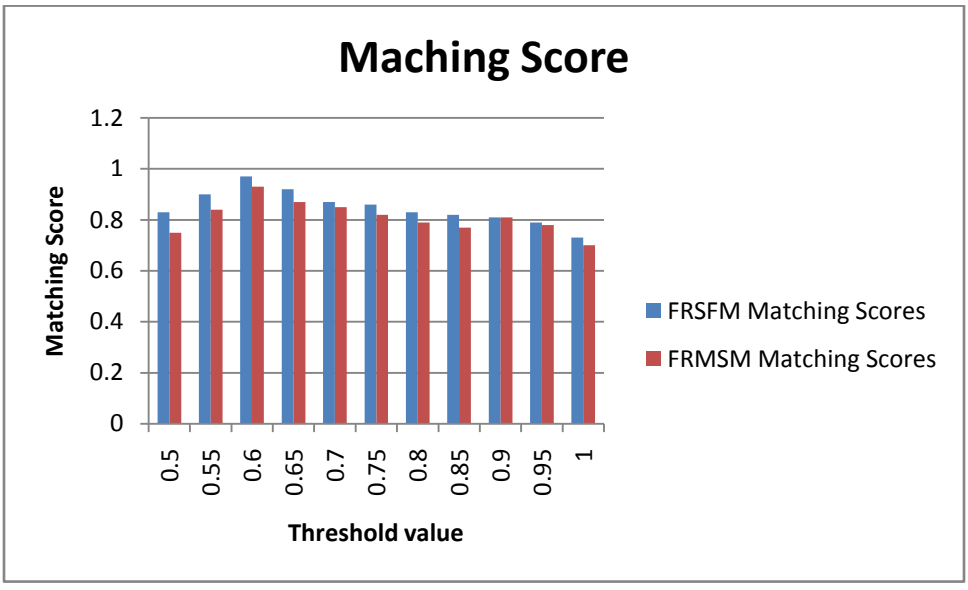


Figure 4.6: Matching Score

4.2 Result

For the experimentation with the fingerprint recognition system FVC2000 database and a self created database is used. The fingerprint database consists of 300 grayscale fingerprint images. For the experiment in fingerprint recognition, the FRMSM operates its preprocessing at an average of 1.523405 seconds; whereas the matching time for this approach is nearly 2.390256 seconds. Its average FMR is 0.02827% and FNMR is 0.024%.

The FRFSM operates its preprocessing at an average of 2.3250093 seconds; whereas the matching time for this approach is nearly 1.030045 seconds. The FMR is 0.00247% and FNMR is 0.0183%.

The total recognition time required for FRFSM is 1.57752715 seconds and that for FRMSM is 1.9568305 seconds. The FAR and FRR, both are found less in FRFSM with a value 0.2802% and 0.0056% respectively. Similarly the matching score for FRFSM is found to be 0.8509 and that for FRMSM is 0.813636.

With reference from various tests, it is found that best extraction as well as matching time is achieved at threshold value 0.6.

Chapter 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

Both the FRSFM and FRMSM algorithms were implemented, along with various preprocessing techniques involves fingerprint conversion, noise removal, binarization, transformation, false minutiae extraction. The post processing techniques for SFM includes false minutiae removal in this stage the H points as well as the spikes are also removed. The matching techniques involve a two stage matching namely alignment stage for maintaining the orientation angle of the fingerprint image as well as the match stage for finding out the matching score. Tests are run on different fingerprint images shows that FRSFM has lower FAR, lower FRR as well as lower average matching with higher matching ratio as compared to FRMSM. Similarly it is also found that the result of fingerprint matching on the basis of parameters like time and matching score varies on the threshold value.

Among both algorithms the FRSFM is also capable of matching low quality images where as the efficiency of FRMSM is degraded in this scenario.

Further it can be asserted that the threshold value recommended by this research work will lead to better accuracy and time efficiency.

5.2 Further Recommendations

With the help of various set of tests it is found that, the best result for fingerprint recognition is achieved between the threshold value 0.6 and 0.72 under problem domain minutiae extraction and matching, the further improvements on algorithms FRSFM and FRMSM can be done based on the above mentioned threshold value. Since the matching score as well as the time efficiency is result of threshold value as well as nature of the fingerprint image the clean and fine fingerprint image guarantee the better approximation result on fingerprint matching.

REFERENCES

[Jai01] A. Jain, A. Ross, S.Prabhakar, "*Fingerprint Matching Using Minutiae and Texture Features*", Proc. OfICIP, pp. 282-285, 2001.

[Jai04] A. Jain, N.K. Karthik, "*Local Correlation-based Fingerprint Matching"*, ICVGIP, Kolkata, 2004.

[Jai97] A. Jain, Lin Hong, and R. Bolle, "*On-line fingerprint verification*". IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(4):302-314, 1997.

[Jai00] A. K. Jain, S.Prabhakar, L. Hong, S.Pankanti, "*Filterbank-Based Fingerprint Matching*", IEEE Trans. OnImage Processing, Vol.9, No.5, pp. 846- 859, 2000.

[ANSI93] American National Standards Institute, "*American national standard for information system, data format for the interchange of fingerprint information*, DOC# ANSI/NIST-CSL 1-1993.

[Lee99] C. J. Lee, S. D. Wang, "*A Gabor Filter-Based Approach to Fingerprint Recognition*", IEEE Workshop on Signal Processing Systems (SiPS 99), pp.371 –378, 1999.

[Alf93] C. L. Alfredo, R. L. Ricardo, C. Q. Reinaldo, "*Fingerprint Recognition*", Electrical Engineering Department, Polytechnic University

[Wan06] C. Wang, M. Gavrilova, Y. Luo, J. Rokne, "*An efficient algorithm for finger print matching",* Proceedings of the 18th International Conference on Pattern Recognition., 2006

[Lee02] D. Lee, K. Choi, and J. Kim, "*A robust fingerprint matching algorithm using local alignment*", In 16th International conference on pattern recognition, volume 3, pages 803-806, 2002.

[Sto86] D.A. Stoney and J.I. Thornton,"*A critical analysis of quantitative fingerprint individuality models*", Journal of Forensic Sciences, 31(4):1187-1216, 1986.

[Ise86] D. K. Isenor, S. G. Zaky, "*Fingerprint Identification Using Graph Matching*", Pattern Recognition, Vol. 19, pp. 111-112, 1986.

[Mai97] D. Maio and D. Maltoni,"*Direct gray scale minutia detection in fingerprints*", IEEE Transactions on Pattern analysis and machine intelligence, 19(1), 1997.

[Dia02] D.Maio, D. Maltoni, R. Cappelli, J.L. Wayman, and A.K. Jain, FVC2002: "*Second fingerprint verification competition*", In Int. Conf. on Pattern Recognition (16^{th}), volume 3, pages 811-814, 2002.

[Cam98] E.D.Campbell, "*Fingerprints & Palmar Dermatoglyphics"*, 1998

[Lee01] H.C. Lee and R.E. Gaensslen, "*Advance in Fingerprint Technology*",Elsevier, New York, 2^{nd} edition, 2001.

[Rav09] J. Ravi. J, K. B. Raja, K. R.Venugopal, "*Fingerprint Recognition Using Minutiae Score Matching*", International Journal of Engineering Science and Technology Vol. 1 (2),2009.

[Ito04] K. Ito, A. Morita, T. Aoki, T. Higuchi, H. Nakajima, K. Kobayashi, "*IEICE TRANS. FUNDAMENTAL*", VOL. E87-A, NO. 3 March 2004

[Ito05] K. Ito, A. Morita, T. Aoki, T. Higuchi, H. Nakajima, K. Kobayashi, "*A Fingerprint Recognition Algorithm Using Phase-Based Image Matching for Low-Quality Fingerprints*", IEEE, 2005

[Hal99] L.C. Jain, U.Halici, I. Hayashi, S.B. Lee and S.Tsutsui. "*Intelligent biometric techniques in fingerprint and face recognition*", the CRC Press, 1999

[Hon98] Lin Hong. "*Automatic Personal Identification Using Fingerprints*", Ph.D. Thesis, 1998.

[Tha10] L. H. Thai, N. H. Tam, "*Fingerprint Recognition using Standardized Fingerprint Model*", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 7, 2010

[Bri95] M. Brislawn and N. Bradly, "*The FBI compression standard for digitized fingerprint images*". In preprocessing of SPIE, volume 2847, pages 344- 355, 1995.

[Ots79] N. Otsu, "*A threshold selection method from gray level histograms*", IEEE Transactions on systems, Man and Cybernetics, 9: 62-66, 1979.

[Rat95] N. Ratha, S. Chen and A.K. Jain, "*Adaptive Flow Orientation Based Feature Extraction in Fingerprint Images*", Pattern Recognition, Vol. 28, pp. 1657-1672, 1995.

[Tri95] O. D. Trier and T. Taxt, "*Evaluation of Binarization methods for document images*", IEEE Transactions on pattern Analysis and Machine Intelligence, 17(3): 312-315, 1995.

[Pan02] S. Pankanti, P. Salil, and A.K. Jain,"*On the individuality of fingerprint*", IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(8): 1010-1025, 2002.

[Pra03] S. pravakar, A. Jain and S. Pankanti, "*Learning fingerprint minutiae location and type*", pattern recognition, 36(8): 1847-1857, 2003.

[Xue06]T.Xuejun, B. Bir, "*Fingerprint matching by genetic algorithms*", Pattern Recognition, vol. 39, 2006, pp. 465 –477.

[Hao99] Y. Hao, T. Tan, Y. Wang, "*An effective algorithm for fingerprint matching*", National Lab of Pattern Recognition, CAS, Institute of Automation, Beijing, P. R. China, 100080,1999

[Che91] Z. Chen, C. H. Kuo, "*A Topology-Based Matching Algorithm for Fingerprint Authentication*", Proc. of 25thAnnual IEEE International Carnahan Conference on Security Technology, pp. 84-87, 1991.

[Vaj00] Z.Miklós K.Vajna, "*A Fingerprint Verification System Based on Triangular Matching and Dynamic Time Warping*", IEEE Tran. On PAMI, Vol. 22, No. 11, 2000.

[Wuh02] Wuzhill, "*Fingerprint Recognition",* Hong Kong Baptist University, 2002

APPENDIX

Read Image

```
[image , pathname]=
uigetfile('*.bmp;*.BMP;*.tif;*.TIF;*.jpg','Open Fingerprint
image');
if image ~= 0
    cd(pathname);
    image1=imread(char(image));
    end;
figure;imshow(image1);title('Input image');
```

Grayscale Conversion

```
figure('units','normalized','outerposition',[0 0 0.5 0.5]);
subplot(1,2,2),imshow(image);
title('Original Image','FontSize',14);
imag_gray=rgb2gray(image);
image_gray=dither(imag_gray);
subplot(1,2,2),imshow(image_gray);
title('Gray Scale Image','FontSize',14);
```

Noise Removal

```
image_denoised=medfilt2(image1);
image_denoised=image_denoised(2:end-1,2:end-1);%remove corner pixels
figure;imshow(image_denoised);title('Denoised image');
```

Image Transformation

```
%Image Transformation by discrete fourier transform
gamma=1;low_in=0;high_in=1;low_out=1;high_out=0;
image_transformed = imadjust(image_denoised, [low_in high_in], [low_out
high_out], gamma);
figure;imshow(image_transformed);title('Transformed image');
```

Image Binarization

```
threshhold=graythresh(image_transformed);%otsu's method for finding
global threshhold%
image_binarized= im2bw(image_transformed, threshhold);
figure;imshow(image_binarized);title('Binarized image');
```

Block Filter

```
image_thinned=bwmorph(image_binarized,'thin','Inf');
figure;imshow(image_thinned);title('Thinned image');
```

Minutiae Extraction and Matching for FRMSM

```
function [ridge_x,ridge_y,bifurcation_x,bifurcation_y]= MatchingMSM(~)
tic
%Minutiae extraction
%thinned_image=loadimage
image1=imread('thin.tif');
s=size(image1);
N=3;%window size
n=(N-1)/2;
r=s(1)+2*n;
c=s(2)+2*n;
double temp(r,c);
temp=zeros(r,c);bifurcation=zeros(r,c);ridge=zeros(r,c);
temp((n+1):(end-n),(n+1):(end-n))=image1(:,:);
outImg=zeros(r,c,3);%For Display
outImg(:,:,1) = temp .* 255;
outImg(:,:,2) = temp .* 255;
outImg(:,:,3) = temp .* 255;
for x=(n+1+10):(s(1)+n-10)
    for y=(n+1+10):(s(2)+n-10)
        e=1;
        for k=x-n:x+n
            f=1;
            for l=y-n:y+n
                mat(e,f)=temp(k,l);
                f=f+1;
            end
            e=e+1;
        end;
         if(mat(2,2)==0)
            ridge(x,y)=sum(sum(~mat));
            bifurcation(x,y)=sum(sum(~mat));
         end
    end;
end;

% RIDGE END FINDING
[ridge_x ridge_y]=find(ridge==2);
len=length(ridge_x);
c1=0;
%For Display
for i=1:len
    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)-3),2:3)=0;
    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)+3),2:3)=0;
    outImg((ridge_x(i)-3),(ridge_y(i)-3):(ridge_y(i)+3),2:3)=0;
    outImg((ridge_x(i)+3),(ridge_y(i)-3):(ridge_y(i)+3),2:3)=0;

    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)-3),1)=255;
    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)+3),1)=255;
    outImg((ridge_x(i)-3),(ridge_y(i)-3):(ridge_y(i)+3),1)=255;
    outImg((ridge_x(i)+3),(ridge_y(i)-3):(ridge_y(i)+3),1)=255;
    c1=len;
end

%BIFURCATION FINDING
```

```
[bifurcation_x bifurcation_y]=find(bifurcation==4);
len1=length(bifurcation_x);
%For Display
for i=1:len1
    outImg((bifurcation_x(i)-3):(bifurcation_x(i)+3),(bifurcation_y(i)-
3),1:2)=0;
    outImg((bifurcation_x(i)-
3):(bifurcation_x(i)+3),(bifurcation_y(i)+3),1:2)=0;
    outImg((bifurcation_x(i)-3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),1:2)=0;
    outImg((bifurcation_x(i)+3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),1:2)=0;

    outImg((bifurcation_x(i)-3):(bifurcation_x(i)+3),(bifurcation_y(i)-
3),3)=255;
    outImg((bifurcation_x(i)-
3):(bifurcation_x(i)+3),(bifurcation_y(i)+3),3)=255;
    outImg((bifurcation_x(i)-3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),3)=255;
    outImg((bifurcation_x(i)+3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),3)=255;
    c1=c1+len1;
end

%figure;imshow(outImg);title('Minutiae');

image2=imread('thin.tif');
s=size(image2);
N=3;%window size
n=(N-1)/2;
r=s(1)+2*n;
c=s(2)+2*n;
double temp(r,c);
temp=zeros(r,c);bifurcation=zeros(r,c);ridge=zeros(r,c);
temp((n+1):(end-n),(n+1):(end-n))=image2(:,:);
outImg=zeros(r,c,3);%For Display
outImg(:,:,1) = temp .* 255;
outImg(:,:,2) = temp .* 255;
outImg(:,:,3) = temp .* 255;
for x=(n+1+10):(s(1)+n-10)
    for y=(n+1+10):(s(2)+n-10)
        e=1;
        for k=x-n:x+n
            f=1;
            for l=y-n:y+n
                mat(e,f)=temp(k,l);
                f=f+1;
            end
            e=e+1;
        end;
         if(mat(2,2)==0)
            ridge(x,y)=sum(sum(~mat));
            bifurcation(x,y)=sum(sum(~mat));
         end
    end;
end;
```

```
% RIDGE END FINDING
[ridge_x ridge_y]=find(ridge==2);
len=length(ridge_x);
c2=0;
%For Display
for i=1:len
    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)-3),2:3)=0;
    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)+3),2:3)=0;
    outImg((ridge_x(i)-3),(ridge_y(i)-3):(ridge_y(i)+3),2:3)=0;
    outImg((ridge_x(i)+3),(ridge_y(i)-3):(ridge_y(i)+3),2:3)=0;

    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)-3),1)=255;
    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)+3),1)=255;
    outImg((ridge_x(i)-3),(ridge_y(i)-3):(ridge_y(i)+3),1)=255;
    outImg((ridge_x(i)+3),(ridge_y(i)-3):(ridge_y(i)+3),1)=255;
    c2=len;
end

%BIFURCATION FINDING
[bifurcation_x bifurcation_y]=find(bifurcation==4);
len1=length(bifurcation_x);
%For Display
for i=1:len1
    outImg((bifurcation_x(i)-3):(bifurcation_x(i)+3),(bifurcation_y(i)-
3),1:2)=0;
    outImg((bifurcation_x(i)-
3):(bifurcation_x(i)+3),(bifurcation_y(i)+3),1:2)=0;
    outImg((bifurcation_x(i)-3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),1:2)=0;
    outImg((bifurcation_x(i)+3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),1:2)=0;

    outImg((bifurcation_x(i)-3):(bifurcation_x(i)+3),(bifurcation_y(i)-
3),3)=255;
    outImg((bifurcation_x(i)-
3):(bifurcation_x(i)+3),(bifurcation_y(i)+3),3)=255;
    outImg((bifurcation_x(i)-3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),3)=255;
    outImg((bifurcation_x(i)+3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),3)=255;
    c2=c2+len1;

end
if(c1==c2)
    fprintf('matched\n');
else
    fprintf('unmatched\n');
end;
toc
end
```

False Minutiae Removal

```
function [res] = filter_minutia(img, index, m_list,  path_len)
```

```
   iax = 0; iay = 0; ibx = 0; iby = 0; icx = 0; icy = 0;
   x = m_list(index, 1);
   y = m_list(index, 2);
   CN = m_list(index, 3);
   progress = 1;

   res = index

   pax = 0; pay=0; pbx = 0; pby=0; pcx = 0; pcy=0;
   pao = 0; pbo = 0; pco = 0;      %1-8 position of 3x3 square
around minutia

   for i = 1:8
         [ta, xa, ya] = p(img, x, y, i);
         [tb, xb, yb] = p(img, x, y, i+1);
         if (ta > tb)
           if pao == 0
              if i < 5
                 pao = 4 + i;
              else
                 pao = mod(4 + i, 9) + 1;
              end
              pax = xa;
              pay = ya;
           elseif pbo == 0
              if i < 5
                 pbo = 4 + i;
              else
                 pbo = mod(4 + i, 9) + 1;
              end
              pbx = xa;
              pby = ya;
           else
              if i < 5
                 pco = 4 + i;
              else
                 pco = mod(4 + i, 9) + 1;
              end
              pcx = xa;
              pcy = ya;
           end
         end
   end


   while ( progress < path_len)
      if ( iax  & ibx )
         break;
      end

      if ( iax & icx )
         break;
```

```
end

if ( ibx  & icx  )
   break;
end


if (CN == 1 & (iax  | ibx ))
   break;
end

progress = progress + 1;

if iax == 0
  iax = find(m_list(:,1) == pax);
  if iax
     m_list(iax, 2)
     iay = find(m_list(iax, 2) == pay)
     if iay == 0
         iax = 0;
     else
         iax = iax(iay);
     end
  else
     iax = 0;
  end
end

if ibx == 0
  ibx = find(m_list(:,1) == pbx);
  if ibx
     iby = find(m_list(ibx, 2) == pby);
     if iby == 0
         ibx = 0;
     else
         ibx = ibx(iby);
     end
  else
       ibx = 0;
  end
end

if icx == 0
  icx = find(m_list(:,1) == pcx);
  if icx
     icy = find(m_list(icx, 2) == pcy);
     if icy == 0
         icx = 0;
     else
         icx = icx(icy);
     end
  else
```

```
     icx = 0;
  end
end


if pao ~= 0
  if iax == 0
    for i = 1:8
       if i == pao
          continue;
       end

       [ta, xa, ya] = p(img, pax, pay, i);

       if ta == 1
          if i < 5
             pao = 4 + i;
          else
             pao = mod(4 + i, 9) + 1;
          end
          pax = xa;
          pay = ya;
          break;
       end
    end
  end
end

if pbo ~= 0
  if ibx == 0
    for i = 1:8
       if i == pbo
          continue;
       end

       [ta, xa, ya] = p(img, pbx, pby, i);

       if ta == 1
          if i < 5
             pbo = 4 + i;
          else
             pbo = mod(4 + i, 9) + 1;
          end
          pbx = xa;
          pby = ya;
          break;
       end
    end
  end
end

if pco ~= 0
```

```
        if icx == 0
          for i = 1:8
               if i == pco
                  continue;
               end

               [ta, xa, ya] = p(img, pcx, pcy, i);
               if ta == 1
                  if i < 5
                     pco = 4 + i;
                  else
                     pco = mod(4 + i, 9) + 1;
                  end
                   pcx = xa;
                  pcy = ya;
                  break;
               end
          end
       end
    end

end


if progress >= path_len
   res = 0;
else
   if iax > 0
     res = index;
   elseif ibx > 0
     res = index;
   elseif icx > 0
     res = index;
   else
     res = 0;
   end
end
```

Matching for FRSFM

```
function [sim, angle, sc_cost, E] = do_match(f1, f2)

%load template fingerprint
X=csvread( [char(f1) '.X']);
m1=csvread( [char(f1) '.m']);
oX=load( [char(f1) '.o']);
rX=load( [char(f1) '.r']);
nX = load( [char(f1) '.n']);
roX = load( [char(f1) '.ro']);
orient_img_x = [];%load( [char(f1) '.oi']);
or_x = [];%load( [char(f1) '.oi']);
```

```
%load test fingerprint
Y=csvread( [char(f2) '.X']);
m2=csvread( [char(f2) '.m']);
oY=load( [char(f2) '.o']);
rY=load( [char(f2) '.r']);
nY = load( [char(f2) '.n']);
roY = load( [char(f2) '.ro']);
orient_img_y = [];%load( [char(f2) '.oi']);
or_y = [];%load( [char(f2) '.oi']);


%Make sure all minutiae type are ridge endings, bifurcations, or
primary cores (pseudo minutia).
i1=find(m1(:,3)<7);
i2=find(m2(:,3)<7);
X=X(i1,:);
m1=m1(i1,:);
Y=Y(i2,:);
m2=m2(i2,:);
m1r=m1(:,4);
m2r=m2(:,4);
m1(:,4)=mod(m1(:,4),pi);
m2(:,4)=mod(m2(:,4),pi);

%Attempt to retrieve index of both test and template fingerprint core
indexes (provided they exist).
test = m1(:,3);
ic1  = find(test == 5);
c1o = -1;
c2o = -1;
test = m2(:,3);
ic2  = find(test == 5);


%Build fingerprint feature structure.
f1=struct('X', X, 'M', m1, 'O', oX, 'R', rX, 'N', nX, 'RO',roX, 'OIMG',
orient_img_x, 'OREL', or_x);
f2=struct('X', Y, 'M', m2, 'O', oY, 'R', rY, 'N', nY, 'RO',roY, 'OIMG',
orient_img_y, 'OREL', or_y);

%Bending energy
E=0;

%Detect if the core is near the edge of the fingerprint. Edge core
comparisons have their similarity
%reduced since they are more prone to error due to lack of distributed
coverage about the core point
%(i.e. much more possible alignment combinations are achieved when
minutiae only lie on one side
% of a core point since we have less rotational contraints).
edge_core = 0;
if numel(ic1) > 0 && numel(ic2) > 0
   a=max(f1.X(:,1));
   b=min(f1.X(:,1));
   if f1.X(ic1,1) > a-0.02 || f1.X(ic1,1) < b+0.02
```

```
      edge_core=1
   end
   a=max(f2.X(:,1));
   b=min(f2.X(:,1));
   if f2.X(ic2,1) > a-0.02 || f2.X(ic2,1) < b+0.02
      edge_core=1
   end
end

%Swap structures to make sure X represents the fingerprint with fewer
minutiae
if size(X,1) > size(Y,1)
  temp = f1;
  f1 = f2;
  f2 = temp;
  temp = ic1;
  ic1 = ic2;
  ic2 = temp;
  temp = m1r;
  m1r = m2r;
  m2r = temp;
end

display_flag=0;
GC=0;

mean_dist_global=[]; % use [] to estimate scale from the data

nbins_theta=19;
nbins_r=5;
eps_dum=1;
r=0.35; % annealing rate
beta_init=0.8;  % initial regularization parameter (normalized)

r_inner=1/8;
r_outer=1/2;

%Register fingerprints
[ft,res]=register(f1,f2);
angle=0;
nsamp1=size(f1.X,1);
nsamp2=size(f2.X,1);
out_vec_1=zeros(1,nsamp1);
out_vec_2=zeros(1,nsamp2);

d1=dist2(f1.X, f1.X);
d2=dist2(f2.X, f2.X);

o_res=[];
sc_res=[];
mc_res=[];
ro_res=[];

%Map out binding box for overlap region.
xt = min(ft.X(:,2));
```

```
xb = max(ft.X(:,2));
xl = min(ft.X(:,1));
xr = max(ft.X(:,1));
yt = min(f2.X(:,2));
yb = max(f2.X(:,2));
yl = min(f2.X(:,1));
yr = max(f2.X(:,1));
region_t=max(xt, yt);
region_b=min(xb, yb);
region_r=min(xr, yr);
region_l=max(xl, yl);
region_a = (region_b - region_t)*(region_r-region_l);

%Find all indices within bounding box
ind1=intersect(intersect(find(ft.X(:,1) > region_l), find(ft.X(:,1) <
region_r)), intersect(find(ft.X(:,2) > region_t), find(ft.X(:,2) <
region_b))  );
ind2=intersect(intersect(find(f2.X(:,1) > region_l), find(f2.X(:,1) <
region_r)), intersect(find(f2.X(:,2) > region_t), find(f2.X(:,2) <
region_b))  );

%get minutiae count for each image in overlap region
ng_samp1=numel(ind1);
ng_samp2=numel(ind2);

if numel(res.map) > 0
   if numel(ic1) > 0 && numel(ic2) > 0
      GC=1;
   end

   f1=ft;
   inda1=res.map1;
   inda2=res.map2;

   if GC ~=1 && ng_samp1*ng_samp2 > numel(inda1)*numel(inda2)
      % overlap region minutiae counts are set to nearest neighbour
minutiae index counts
      ng_samp1=numel(inda1);
      ng_samp2=numel(inda2);
   end

   % overlap region index sets have anchor minutiae indexes removed.
   inda1=setdiff(inda1(find(f1.M(inda1,3) < 5)),res.map(:,1));
   inda2=setdiff(inda2(find(f2.M(inda2,3) < 5)),res.map(:,2));

   y=0;
   redo=[];
   res.map=res.map(setdiff(1:size(res.map,1), redo),:);
   f1.M(:,4)=mod(m1r-res.angle,2*pi);

   orients=[];
   for i=1:numel(inda1)
       for j=1:numel(inda2)
           if f1.M(inda1(i),3) < 5 && f2.M(inda2(j),3) < 5
```

```
            orients(i,j) = calc_orient(f1.X(inda1(i),:),
f1.R(inda1(i),:), f2.X(inda2(j),:), f2.R(inda2(j),:), []);
         else
            orients(i,j)=0;
         end
      end
   end

   if numel(inda1) > 1 && numel(inda2 ) > 1
     if numel(inda1) > numel(inda2)
        orients=orients';
        t_res_map=res.map(:,[2,1]);
        [sc_cost3, E, cvec,angle] = tps_iter_match(f2.M, f1.M, f2.X,
f1.X, orients, nbins_theta, nbins_r,r_inner,r_outer,3, r, beta_init,
inda2, inda1, t_res_map);
        if numel(cvec) > 0
           xx=[inda1(cvec(:,2)); inda2(cvec(:,1))]'
123
           res.map=[res.map; xx];
        end
     else
        [sc_cost3, E, cvec,angle] = tps_iter_match(f1.M, f2.M, f1.X,
f2.X, orients, nbins_theta, nbins_r,r_inner,r_outer,3, r, beta_init,
inda1, inda2, res.map);

        if numel(cvec) > 0
           xx=[inda1(cvec(:,1)); inda2(cvec(:,2))]'
123
           res.map=[res.map; xx];
        end
     end
   end

   d1=sqrt(dist2(f1.X,f1.X));
   d2=sqrt(dist2(f2.X,f2.X));

   s1=numel(find(f1.M(:,3) < 5));
   s2=numel(find(f2.M(:,3) < 5));
else
   res.map=[];
   res.o_res=[];
end


nX=[];
nY=[];

ns=3;
n_weight=[];
same_type=[];


if numel(res.map) > 0
   for i=1:size(res.map,1)
       if res.map(i,1) == 0
```

```
            continue
        end
        x=m1(res.map(i,1),1);
        y=m1(res.map(i,1),2);

        bonus=1;
        a1=mod(m1r(res.map(i,1))-res.angle,2*pi);
        a2=m2r(res.map(i,2));

        bonus=bonus*exp(-min(abs(a1-a2), 2*pi-abs(a1-a2)));

        if GC==1 && f1.M(res.map(i,1),3) ~= f2.M(res.map(i,2),3)
            bonus=bonus-0.1;
            same_type(i)=0;
        else
            same_type(i)=1;
        end

        [dx,ii]=sort(d1(res.map(i,1),:));
        [dy,jj]=sort(d2(res.map(i,2),:));
        dd1=f1.N((res.map(i,1)-1)*(s1-1) + 1:(res.map(i,1)-1)*(s1-1) +
ns,3:9);
        dd2=f2.N((res.map(i,2)-1)*(s2-1) + 1:(res.map(i,2)-1)*(s2-1) +
ns,3:9);
        dd1(:,3)=mod(dd1(:,3)-res.angle,2*pi);
        dd2(:,3)=mod(dd2(:,3),2*pi);
        dd1(:,7)=mod(dd1(:,3),pi);
        dd2(:,7)=mod(dd2(:,3),pi);

        used=[];
        m_score = 0;
        t=1;
        for x=1:ns
            for y=1:ns
                    if numel(find(used==y))
                        continue
                    end
                    a_diff = min(abs(dd1(x,3)-dd2(y,3)), 2*pi-
abs(dd1(x,3)-dd2(y,3)) );
                    dist_diff =  abs(dd1(x,1) - dd2(y,1));
                    lo_diff =  abs(dd1(x,6) - dd2(y,6));
                    o_diff=min(abs(dd1(x,7)-dd2(y,7)), pi-abs(dd1(x,7)-
dd2(y,7)));
                    if dist_diff < 0.05 && a_diff < pi/2
                        m_score=m_score + exp(-o_diff)*exp(-
dist_diff)*exp(-a_diff);
                        used(t)=y; t=t+1;
                    end
            end
         end
         if m_score == 0
%           continue
         end

         n_weight(i)=m_score + bonus;
```

```
        rox=f1.RO(res.map(i,1),:);
        roy=f2.RO(res.map(i,2),:);
        mox=f1.M(res.map(i,1),4);
        moy=f2.M(res.map(i,2),4);
        z=min([abs(mox-moy), abs(pi - abs(mox-moy))]) * 2/pi;

        t1=find(rox < 0);
        t2=find(roy < 0);
        t=[t1 t2 9];
        if numel(min(t)) > 0
           rox=rox(1:min(t)-1);
           roy=roy(1:min(t)-1);
           ro_res(i)=max(abs(rox-roy));
           if ro_res(i) < 0.1 && min(t)>4
              n_weight(i)=n_weight(i)+0.2;
           end
        else
           ro_res(i)=0;
        end

        [o_res(i), ih]= calc_orient(f1.O(res.map(i,1),:),
f1.R(res.map(i,1),:), f2.O(res.map(i,2),:), f2.R(res.map(i,2),:),
[1:1]');

        mc_res(i)=z;
        sc_res(i)=1;%costmat(res.map(i,1),res.map(i,2));
        res.map(setdiff(i,find(res.map(:,1)==res.map(i,1))), 1)=0;
        res.map(setdiff(i,find(res.map(:,2)==res.map(i,2))), 1)=0;
    end
end

sc_cost=100;

if numel(o_res) > 1
    inda1=union(inda1,res.map(:,1));
    inda2=union(inda2,res.map(:,2));

    A=f1.X(inda1,:);
    B=f2.X(inda2,:);
    out_vec_1=zeros(1,size(A,1));
    out_vec_2=zeros(1,size(B,1));


[BH1,mean_dist_1]=sc_compute(A',f1.M(inda1,4)',mean_dist_global,nbins_t
heta,nbins_r,r_inner,r_outer,out_vec_1);

[BH2,mean_dist_2]=sc_compute(B',f2.M(inda2,4)',mean_dist_1,nbins_theta,
nbins_r,r_inner,r_outer,out_vec_2);
    %compute pairwise cost between all shape contexts
    costmat=hist_cost_2(BH1,BH2,r_inner, r_outer, nbins_theta, nbins_r);
    sc_vals=[];

    for i=1:numel(res.map)/2
```

```
       if numel(costmat(find(inda1==res.map(i,1)),
find(inda2==res.map(i,2)))) > 0
          sc_vals(i)=costmat(find(inda1==res.map(i,1)),
find(inda2==res.map(i,2)));
       else
          sc_vals(i)=10;
       end
   end
   sc_cost=mean(sc_vals);%/exp(-(0.7-max(sqrt(res.area),0)))


end

unknown_c=numel(find(o_res == -1));

ind=intersect(find(o_res > 0.25), find(ro_res < 0.897));

for i=1:numel(ind)
    dd1=f1.N((res.map(ind(i),1)-1)*(s1-1) + 1:(res.map(ind(i),1)-
1)*(s1-1) + ns,3:9);
    dd2=f2.N((res.map(ind(i),2)-1)*(s2-1) + 1:(res.map(ind(i),2)-
1)*(s2-1) + ns,3:9);
end

o_res=o_res(ind);
ro_res=ro_res(ind);
mc_res=mc_res(ind);
sc_res=sc_res(ind);
n_weight=n_weight(ind);
same_type=same_type(ind);
%o_res=o_res.*n_weight;

res.sc/res.area;
res.sc;
ro_res; %=1/(sum(ro_res)/numel(ro_res))
o_res ;%=sum(o_res)/numel(o_res)
n_weight;
mc_res; %=1/sum(mc_res)/numel(mc_res)
sc_res;

nX=intersect(nX,ind1);
nY=intersect(nY,ind2);
numel(ind);
size(res.map,1);

ns1=numel(nX);
ns2=numel(nY);

ng_samp1=ng_samp1-unknown_c        %=ns1; %ng_samp1+(ns1/2);
ng_samp2=ng_samp2-unknown_c        %=ns2; %ng_samp2+(ns2/2);

ic1  = find(f1.M(:,3) == 5);
ic2  = find(f2.M(:,3) == 5);
```

```
o_a=zeros(numel(ind), numel(ind));
o_b=zeros(numel(ind), numel(ind));
for i=1:numel(ind)
    for j=i+1:numel(ind)
        [o_a(i,j),ia,ra]=calc_orient(f1.O(res.map(i,1),:),
f1.R(res.map(i,1),:), f1.O(res.map(j,1),:), f1.R(res.map(j,1),:),[]);
        [o_b(i,j),ib,rb]=calc_orient(f2.O(res.map(i,2),:),
f2.R(res.map(i,2),:), f2.O(res.map(j,2),:), f2.R(res.map(j,2),:), []);
        o_a=min(o_a, 1);
        o_b=min(o_b, 1);

        if (numel(intersect(ra,rb)) < 120)
           o_a(i,j)=0;
           o_b(i,j)=0;
        end

        o_a(j,i)=o_a(i,j);
        o_b(j,i)=o_b(i,j);
    end
end


vv=1;
if (numel(mc_res) >= 2) && res.area > -1
         hold on
%         figure(1)
         subplot(2,2,4);
         plot(f1.X(:,1),f1.X(:,2),'b+',f2.X(:,1),f2.X(:,2),'ro')
         if GC == 1
         end
         title(['final'])
         hold off
         drawnow

   v=max(abs(o_a-o_b));
   vv=median(v)

sim=(numel(mc_res)^2)*sqrt(max(o_res))*mean(n_weight)/max((ng_samp1*ng_
samp2),1);
   if edge_core == 1
      sim=sim*0.5;
   end
else
   sim=0;
end
sc_cost
sim
```