Fingerprint Recognition using MATLAB

Graduation project

Prepared by:
Zain S. Barham
Supervised by:
Dr. Allam Mousa

5/17/2011

Acknowledgement

I would like to extend my heartiest gratitude to Dr. Allam A. Mousa, my project supervisor, for his invaluable guidance, inspirations and timely suggestions which facilitated the entire process of bringing out this project on “fingerprint recognition using Matlab”.

It would have been really hard to complete this project successfully without the directions of Dr. Mousa.

Also, I would like to express my deepest thanks to the dear Miss Nuha Odeh, for her help which was a huge contribution to the completion of this project; I really couldn’t have done it without her.

Contents

Abstract

Human fingerprints are rich in details called minutiae, which can be used as identification marks for fingerprint verification. The goal of this project is to develop a complete system for fingerprint verification through extracting and matching minutiae. To achieve good minutiae extraction in fingerprints with varying quality, preprocessing in form of image enhancement and binarization is first applied on fingerprints before they are evaluated. Many methods have been combined to build a minutia extractor and a minutia matcher. Minutia-marking with false minutiae removal methods are used in the work. An alignment-based elastic matching algorithm has been developed for minutia matching. This algorithm is capable of finding the correspondences between input minutia pattern and the stored template minutia pattern without resorting to exhaustive search. Performance of the developed system is then evaluated on a database with fingerprints from different people.

Chapter one: Introduction

1.1 INTRODUCTION

Personal identification is to associate a particular individual with an identity. It plays a critical role in our society, in which questions related to identity of an individual such as “Is this the person who he or she claims to be?”, “Has this applicant been here before?”, “Should this individual be given access to our system?” “Does this employee have authorization to perform this transaction?” etc are asked millions of times every day by hundreds of thousands of organizations in financial services, health care, electronic commerce, telecommunication, government, etc. With the rapid evolution of information technology, people are becoming even more and more electronically connected. As a result, the ability to achieve highly accurate automatic personal identification is becoming more critical.

A wide variety of systems require reliable personal authentication schemes to either confirm or determine the identity of individuals requesting their services. The purpose of such schemes is to ensure that the rendered services are accessed by a legitimate user, and not anyone else. Examples of these systems include secure access to buildings, computer systems, laptops, cellular phones and ATMs. In the absence of robust authentication schemes, these systems are vulnerable to the wiles of an impostor.

Traditionally, passwords (knowledge-based security) and ID cards (token-based security) have been used to restrict access to systems. The major advantages of this traditional personal identification are that

(i) They are very simple

(ii) They can be easily integrated into different systems with a low cost.

However these approaches are not based on any inherent attributes of an individual to make a personal identification thus having number of disadvantages like tokens may be lost, stolen, forgotten, or misplaced; PIN may be forgotten or guessed by impostors. Security can be easily breached in these systems when a password is divulged to an unauthorized user or a card is stolen by an impostor; further, simple passwords are easy to guess (by an impostor) and difficult passwords may be hard to recall (by a legitimate user).Therefore they are unable to satisfy the security requirements of our electronically interconnected information society. The emergence of biometrics has addressed the problems that plague traditional verification.

1.2 Biometrics

In the world of computer security, biometrics refers to authentication techniques that rely on measurable physiological and individual characteristics that can be automatically verified. In other words, we all have unique personal attributes that can be used for distinctive identification purposes, including a fingerprint, the pattern of a retina, and voice characteristics. Strong or two-factor authentication—identifying oneself by two of the three methods of something you know (for example, a password), have (for example, a swipe card), or is (for example, a fingerprint)—is becoming more of a genuine standard in secure computing environments. Some personal computers today can include a fingerprint scanner where you place your index finger to provide authentication. The computer analyzes your fingerprint to determine who you are and, based on your identity followed by a pass code or pass phrase, allows you different levels of access. Access levels can include the ability to open sensitive files, to use credit card information to make electronic purchases, and so on.

1.3 Biometrics Authentication Techniques

A biometric authentication is essentially a pattern-recognition that makes a personal identification by determining the authenticity of a specific physiological or behavioral characteristic possessed by the user. An important issue is designing a practical approach to determine how an individual is identified. An authentication can be divided into two modules:

a.) Enrollment module

b.) Identification or Verification module

1.4 How Biometric Technologies Work

The enrollment module is responsible for enrolling individuals into the biometric system. During the enrollment phase, the biometric characteristic of an individual is first scanned by a biometric reader to produce a raw digital representation of the characteristic. In order to

facilitate matching, the raw digital representation is usually further processed by feature extractor to generate a compact but expensive representation, called a template.

Depending on the application, the template may be stored in the central database. Depending on the application, biometrics can be used in one of two modes: verification or identification. Verification—also called authentication—is used to verify a person's identity—that is, to authenticate that individuals are who they say they are. Identification is used to establish a person's identity—that is, to determine who a person is. Although biometric technologies measure different characteristics in substantially different ways, all biometric systems start with an enrollment stage followed by a matching stage that can use either verification or identification.

1.4.1 Enrollment

In enrollment, a biometric system is trained to identify a specific person. The person first provides an identifier, such as an identity card. The biometric is linked to the identity specified on the identification document. He or she then presents the biometric (e.g., fingertips, hand, or iris) to an acquisition device. The distinctive features are located and one or more samples are extracted, encoded, and stored as a reference template for future comparisons. Depending on the technology, the biometric sample may be collected as an image, a recording, or a record of related dynamic measurements. How biometric systems extract features and encode and store information in the template is based on the system vendor's proprietary algorithms. Template size varies depending on the vendor and the technology. Templates can be stored remotely in a central database or within a biometric reader device itself; their small size also allows for storage on smart cards or tokens.

Minute changes in positioning, distance, pressure, environment, and other factors influence the generation of a template. Consequently, each time an individual's biometric data are captured, the new template is likely to be unique. Depending on the biometric system, a person may need to present biometric data several times in order to enroll.

Either the reference template may then represent an amalgam of the captured data or several enrollment templates may be stored. The quality of the template or templates is critical in the overall success of the biometric application. Because biometric features can change over time, people may have to reenroll to update their reference template.

Some technologies can update the reference template during matching operations. The enrollment process also depends on the quality of the identifier the enrollee presents. The reference template is linked to the identity specified on the identification document. If the identification document does not specify the individual's true identity, the reference template will be linked to a false identity.

1.4.2 Verification

In verification systems, the step after enrollment is to verify that a person is who he or she claims to be (i.e., the person who enrolled). After the individual provides an identifier, the biometric is presented, which the biometric system captures, generating a trial template that is based on the vendor's algorithm. The system then compares the trial biometric template with this person's reference template, which was stored in the system during enrollment, to determine whether the individual's trial and stored templates match.

Verification is often referred to as 1:1 (one-to-one) matching. Verification systems can contain databases ranging from dozens to millions of enrolled templates but are always predicated on matching an individual's presented biometric against his or her reference template. Nearly all verification systems can render a match–no-match decision in less than a second.

One of the most common applications of verification is a system that requires employees to authenticate their claimed identities before granting them access to secure buildings or to computers.

1.4.3 Identification

In identification systems, the step after enrollment is to identify who the person is. Unlike verification systems, no identifier is provided. To find a match, instead of locating and comparing the person's reference template against his or her presented biometric, the trial template is compared against the stored reference templates of all individuals enrolled in the system. Identification systems are referred to as 1: M (one-to-M, or one-to-many) matching because an individual's biometric is compared against multiple biometric templates in the system's database. There are two types of identification systems: positive and negative. Positive identification systems are designed to ensure that an individual's biometric is enrolled in the database. The anticipated result of a search is a match. A typical positive identification system controls

access to a secure building or secure computer by checking anyone who seeks access against a database of enrolled employees. The goal is to determine whether a person seeking access can be identified as having been enrolled in the system. Negative identification systems are designed to ensure that a person's biometric information is not present in a database. The anticipated result of a search is a no match. Comparing a person's biometric information against a database of all who are registered in a public benefits program, for example, can ensure that this person is not "double dipping" by using fraudulent documentation to register under multiple identities. Another type of negative identification system is a watch list system. Such systems are designed to identify people on the watch list and alert authorities for appropriate action. For all other people, the system is to check that they are not on the watch list and allow them normal passage. The people whose biometrics is in the database in these systems may not have provided them voluntarily. For instance, for a surveillance system, the biometric may be faces captured from mug shots provided by a law enforcement agency.

1.4.4 Matches Are Based on Threshold Settings

No match is ever perfect in either verification or identification system, because every time a biometric is captured, the template is likely to be unique. Therefore, biometric systems can be configured to make a match or no-match decision, based on a predefined number, referred to as a threshold, which establishes the acceptable degree of similarity between the trial template and the enrolled reference template. After the comparison, a score representing the degree of similarity is generated, and this score is compared to the threshold to make a match or no-match decision. Depending on the setting of the threshold in identification systems, sometimes several reference templates can be considered matches to the trial template, with the better scores corresponding to better matches.

1.5 Leading Biometric Technologies

A growing number of biometric technologies have been proposed over the past several years, but only in the past 5 years have the leading ones become more widely deployed.

Some technologies are better suited to specific applications than others, and some are more acceptable to users. We describe seven leading biometric technologies:

- Facial Recognition
- Fingerprint Recognition
- Hand Geometry
- Iris Recognition
- Signature Recognition
- Speaker Recognition

Fingerprint Recognition

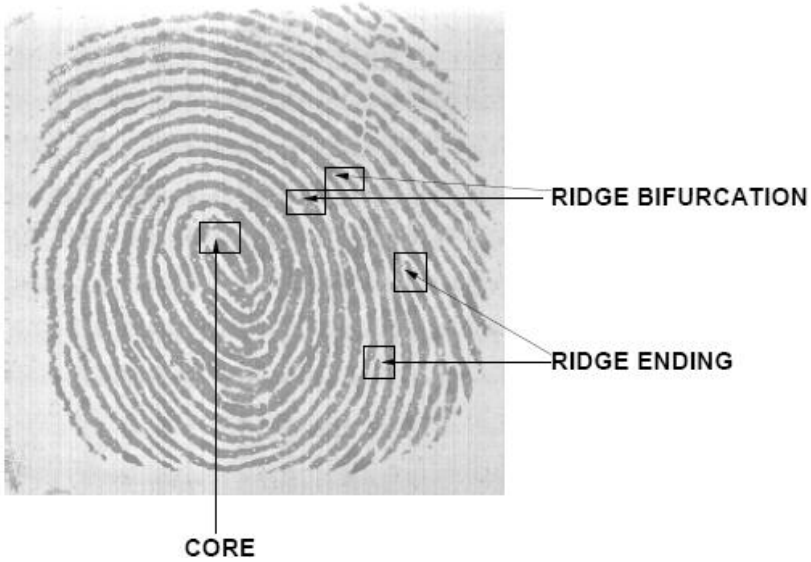
Fingerprint recognition is one of the best known and most widely used biometric technologies. Automated systems have been commercially available since the early 1970s, and at the time of our study, we found there were more than 75 fingerprint recognition technology companies. Until recently, fingerprint recognition was used primarily in law enforcement applications.

Fingerprint recognition technology extracts features from impressions made by the distinct ridges on the fingertips. The fingerprints can be either flat or rolled. A flat print captures only an impression of the central area between the fingertip and the first knuckle; a rolled print captures ridges on both sides of the finger.

An image of the fingerprint is captured by a scanner, enhanced, and converted into a template. Scanner technologies can be optical, silicon, or ultrasound technologies. Ultrasound, while potentially the most accurate, has not been demonstrated in widespread use. In 2002, we found that optical scanners were the most commonly used. During enhancement, “noise” caused by such things as dirt, cuts, scars, and creases or dry, wet or worn fingerprints is reduced, and the definition of the ridges is enhanced. Approximately 80 percent of vendors base their algorithms on the extraction of minutiae points relating to breaks in the ridges of the fingertips. Other algorithms are based on extracting ridge patterns.

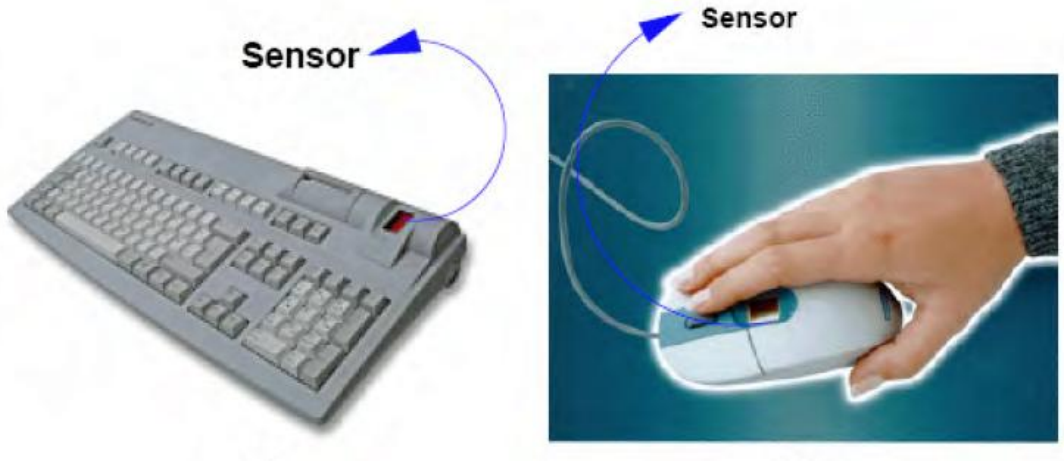
1.6 Fingerprints as a Biometric

Among all biometric traits, fingerprints have one of the highest levels of reliability and have been extensively used by forensic experts in criminal investigations. A fingerprint refers to the flow of ridge patterns in the tip of the finger. The ridge flow exhibits anomalies in local regions of the fingertip (Figure), and it is the position and orientation of these anomalies that are used to represent and match fingerprints.



Although not scientifically established, fingerprints are believed to be unique across individuals, and across fingers of the same individual. Even identical twins having similar DNA, are believed to have different fingerprints. Traditionally, fingerprint patterns have been extracted by creating an inked impression of the fingertip on paper.

The electronic era has ushered in a range of compact sensors that provide digital images of these patterns. These sensors can be easily incorporated into existing computer peripherals like the mouse or the keyboard (figure), thereby making this mode of identification a very attractive proposition. This has led to the increased use of automatic fingerprint-based authentication systems in both civilian and law enforcement applications.



1.6.1 Fingerprint Representation

The uniqueness of a fingerprint is determined by the topographic relief of its ridge structure and the presence of certain ridge anomalies termed as minutiae points.

Typically, the global configuration defined by the ridge structure is used to determine the class of the fingerprint, while the distribution of minutiae points is used to match and establish the similarity between two fingerprints.

Automatic fingerprint identification systems, that match a query print against a large database of prints (which can consist of millions of prints), rely on the pattern of ridges in the query image to narrow their search in the database (fingerprint indexing), and on the minutiae points to determine an exact match (fingerprint matching). The ridge flow pattern itself is rarely used for matching fingerprints.

1.6.2 Minutiae

Minutiae, in fingerprinting terms, are the points of interest in a fingerprint, such as bifurcations (a ridge splitting into two) and ridge endings. Examples are:

a.) ridge endings - a ridge that ends abruptly

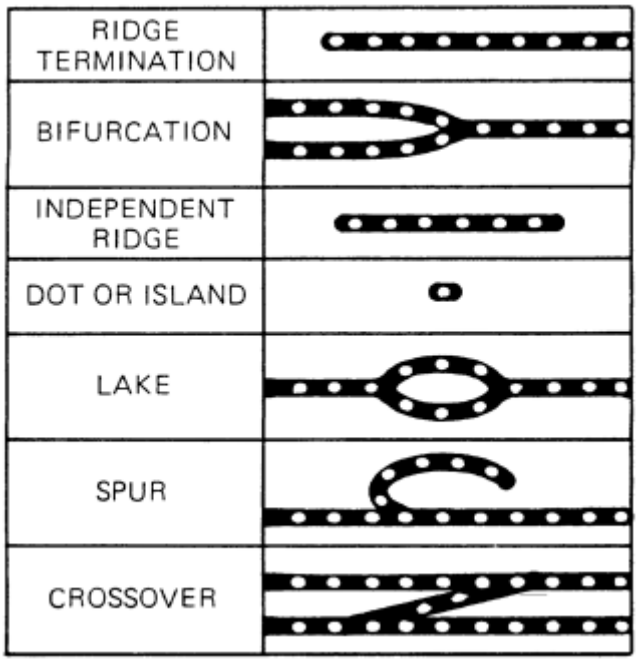
b.) ridge bifurcation - a single ridge that divides into two ridges

c.) short ridges, island or independent ridge - a ridge that commences, travels a short distance and then ends

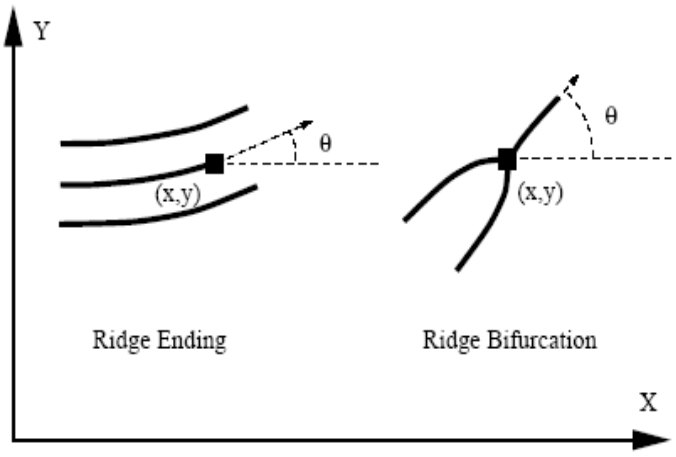
d.) ridge enclosures - a single ridge that bifurcates and reunites shortly afterward to continue as a single ridge

e.) spur - a bifurcation with a short ridge branching off a longer ridge

f.) crossover or bridge - a short ridge that runs between two parallel ridges



Minutiae also refer to any small or otherwise incidental details.
But the focus when matching is only on the 2 main minutiae; ridge ending and ridge bifurcation.



Chapter two: Motivation for the project

2.1 Problem Definition

We propose a simple and effective approach for Biometric fingerprint image enhancement and minutiae extraction based on the frequency and orientation of the local ridges and thereby extracting correct minutiae points.

Automatic and reliable extraction of minutiae from fingerprint images is a critical step in fingerprint matching. The quality of input fingerprint images plays an important role in the performance of automatic identification and verification algorithms. In this project we presents a fast fingerprint enhancement and minutiae extraction algorithm which improves the clarity of the ridge and valley structures of the input fingerprint images based on the frequency and orientation of the local ridges and thereby extracting correct minutiae.

Fingerprint based identification has been one of the most successful biometric techniques used for personal identification. Each individual has unique fingerprints. A fingerprint is the pattern of ridges and valleys on the finger tip. A fingerprint is thus defined by the uniqueness of the local ridge characteristics and their relationships. Minutiae points are these local ridge characteristics that occur either at a ridge ending or a ridge bifurcation. A ridge ending is defined as the point where the ridge ends abruptly and the ridge bifurcation is the point where the ridge splits into two or more branches. Automatic minutiae detection becomes a difficult task in low quality fingerprint images where noise and contrast deficiency result in pixel configurations similar to that of minutiae. This is an important aspect that has been taken into consideration in this presentation for extraction of the minutiae with a minimum error in a particular location. A complete minutiae extraction scheme for automatic fingerprint recognition systems is presented. The proposed method uses improving alternatives for the image enhancement process, leading consequently to an increase of the reliability in the minutiae extraction task.

2.2 Motivation for the Project

Accurate automatic personal identification is critical in wide range of application domains such as national ID cards, electronic commerce and automatic banking. Biometrics, which refers to automatic identification of a person based on his or her personal physiological or behavioral characteristics, is inherently more reliable and more capable in differentiating between a reliable person and a fraudulent impostor than traditional methods such as PIN and passwords. Automatic fingerprint identification is one of the most reliable biometric technology among the different major biometric technologies which are either currently available or under investigation. The objective of our project is to implement the image enhancement and minutiae extraction algorithm which is capable of doing the matching between different digitized fingerprints of standard image file formats namely; BMP, JPEG with high level of accuracy and confidence.

2.3 About the Project

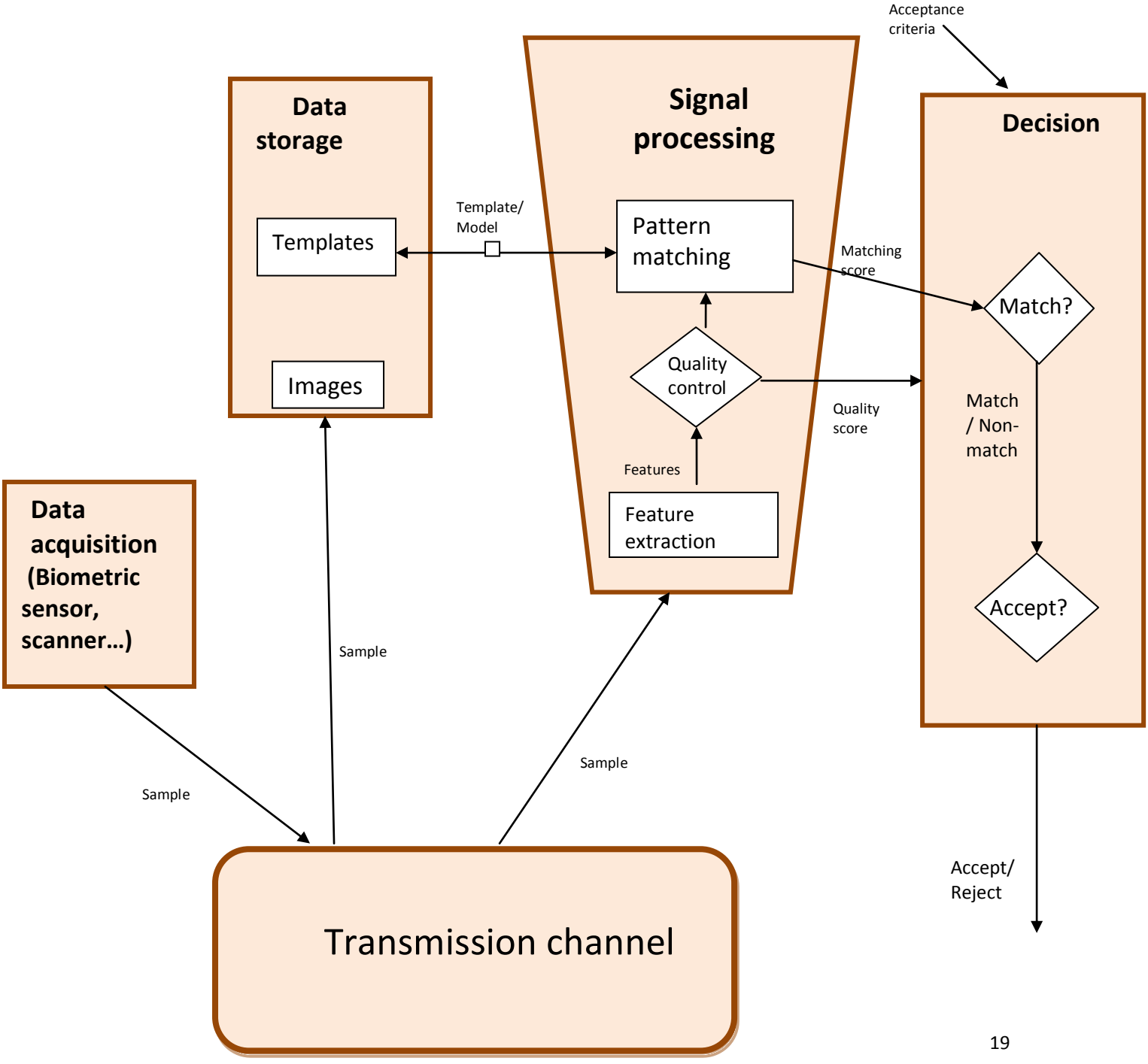
We propose a simple and effective approach for fingerprint image enhancement and minutiae extraction based on the frequency and orientation of the local ridges and thereby extracting correct minutiae.

Automatic and reliable extraction of minutiae from fingerprint images is a critical step in fingerprint matching. The quality of input fingerprint images plays an important role in the performance of automatic identification and verification algorithms. In this project we presents a fast fingerprint enhancement and minutiae extraction algorithm which improves the clarity of the ridge and valley structures of the input fingerprint images based on the frequency and orientation of the local ridges and thereby extracting correct minutiae.

Fingerprint based identification has been one of the most successful biometric techniques used for personal identification. Each individual has unique fingerprints. A fingerprint is the pattern of ridges and valleys on the finger tip. A fingerprint is thus defined by the uniqueness of the local ridge characteristics and their relationships. Minutiae points are these local ridge characteristics that occur either at a ridge ending or a ridge bifurcation.

A ridge ending is defined as the point where the ridge ends abruptly and the ridge bifurcation is the point where the ridge splits into two or more branches. Automatic minutiae detection becomes a difficult task in low quality fingerprint images where noise and contrast deficiency result in pixel configurations similar to that of minutiae. This is an important aspect that has been taken into consideration in this presentation for extraction of the minutiae with a minimum error in a particular location.

A complete minutiae extraction scheme for automatic fingerprint recognition systems is presented. The proposed method uses improving alternatives for the image enhancement process, leading consequently to an increase of the reliability in the minutiae extraction task.



Chapter three: System design

2.1 System Level Design

A fingerprint recognition system constitutes of fingerprint acquiring device, minutia extractor and minutia matcher.

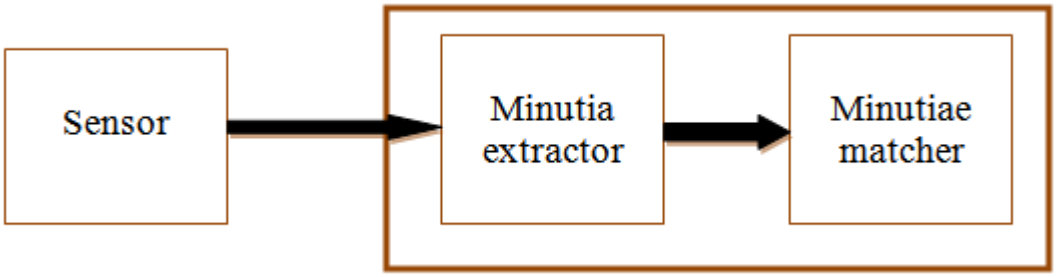


Figure 2.1.1 Simplified Fingerprint Recognition System

For fingerprint acquisition, optical or semi-conduct sensors are widely used. They have high efficiency and acceptable accuracy except for some cases that the user's finger is too dirty or dry. However, the testing database for my project consists of scanned fingerprints using the ink and paper technique because this method introduces a high level of noise to the image and the goal of designing a recognition system is to work with the worst conditions to get the best results.

The minutia extractor and minutia matcher modules are explained in detail later on in this paper.

2.2 Algorithm Level Design

To implement a minutia extractor, a three-stage approach is widely used by researchers. They are preprocessing, minutia extraction and post-processing stage.

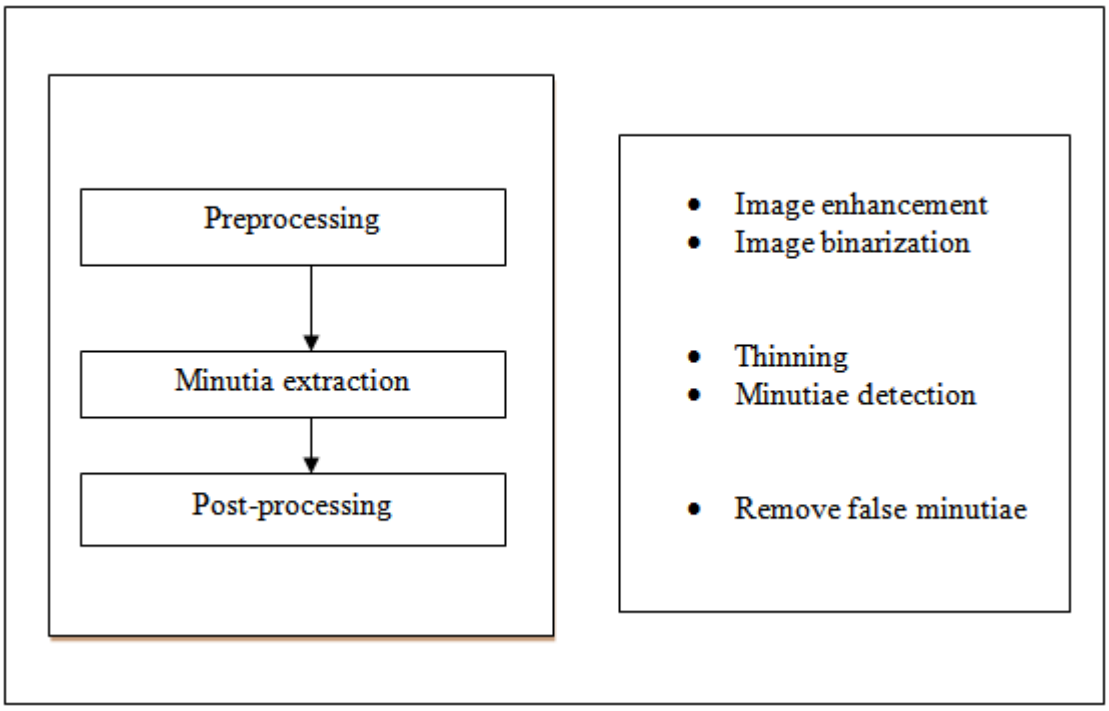


Figure 2.2.1 Minutia Extractor

For the fingerprint image preprocessing stage, Histogram Equalization and Fourier Transform are used to do image enhancement. And then the fingerprint image is binarized using the locally adaptive threshold method. The image segmentation task is fulfilled by a three-step approach: block direction

estimation, segmentation by direction intensity and Region of Interest extraction by Morphological operations.

For minutia extraction stage, iterative parallel thinning algorithm is used. The minutia marking is a relatively simple task. For the post-processing stage, a more rigorous algorithm is developed to remove false minutia.

The minutia matcher chooses any two minutiae as a reference minutia pair and then matches their associated ridges first. If the ridges match well, the two fingerprint images are aligned and matching is conducted for all the remaining minutiae.

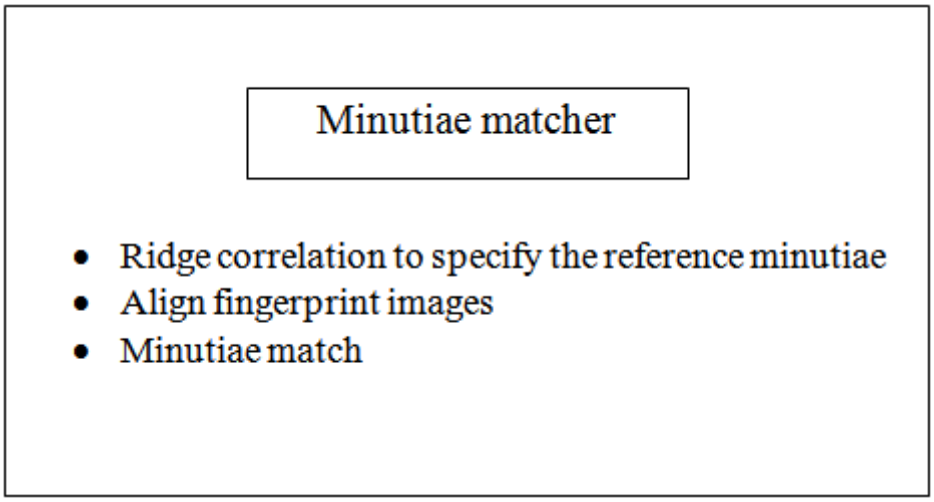


Figure2.2.2 Minutia Matcher

Chapter four: Fingerprint image preprocessing

4.1 Fingerprint Image Enhancement

Fingerprint Image enhancement is used to make the image clearer for easy further operations. Since the fingerprint images acquired from scanner or any other media are not assured with perfect quality, those enhancement methods, for increasing the contrast between ridges and valleys and for connecting the false broken points of ridges due to insufficient amount of ink, are very useful for keep a higher accuracy to fingerprint recognition.

Originally, the enhancement step was supposed to be done using the canny edge detector. But after trial, it turns out that the result of an edge detector is an image with the borders of the ridges highlighted. Using edge detection would require the use of an extra step to fill out the shapes which would consume more processing time and would increase the complexity of the code, as shown in figure [4.1.1].

So, for this part of the project, two Methods are adopted for image enhancement stage: the first one is Histogram Equalization; the next one is Fourier Transform.

4.1.1 Histogram Equalization:

Histogram equalization is to expand the pixel value distribution of an image so as to increase the perceptional information. The original histogram of a fingerprint image is shown in [Figure 4.1.1.1], the histogram after the histogram equalization is shown in [Figure 4.1.1.2].

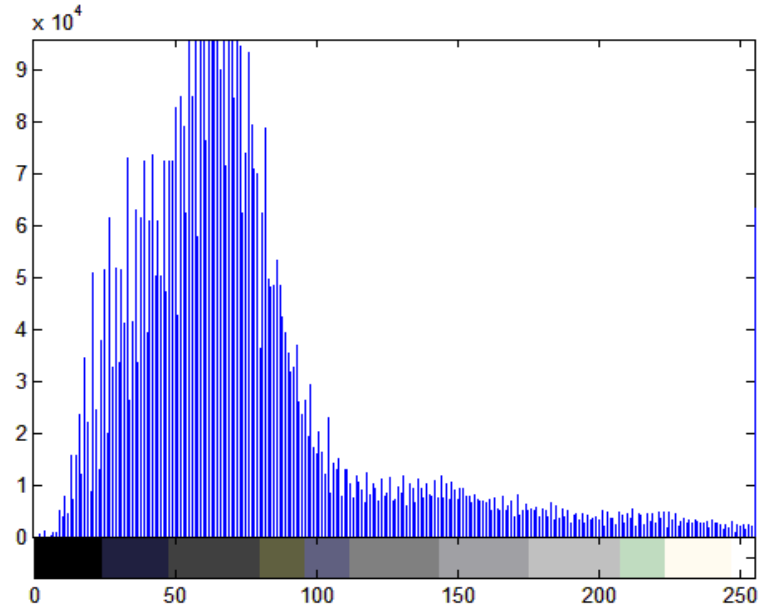


Figure [4.1.1.1]

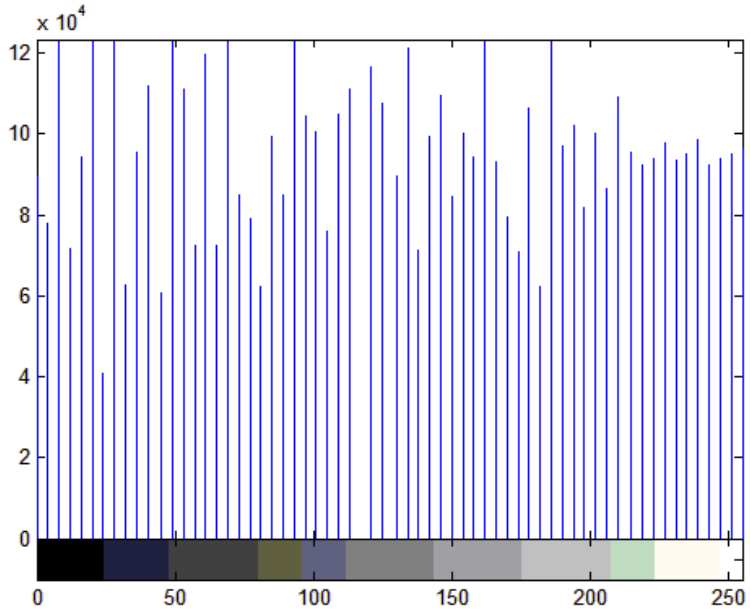


Figure [4.1.1.2]

The right side of the following figure [Figure 4.1.1.3] is the output after the histogram equalization

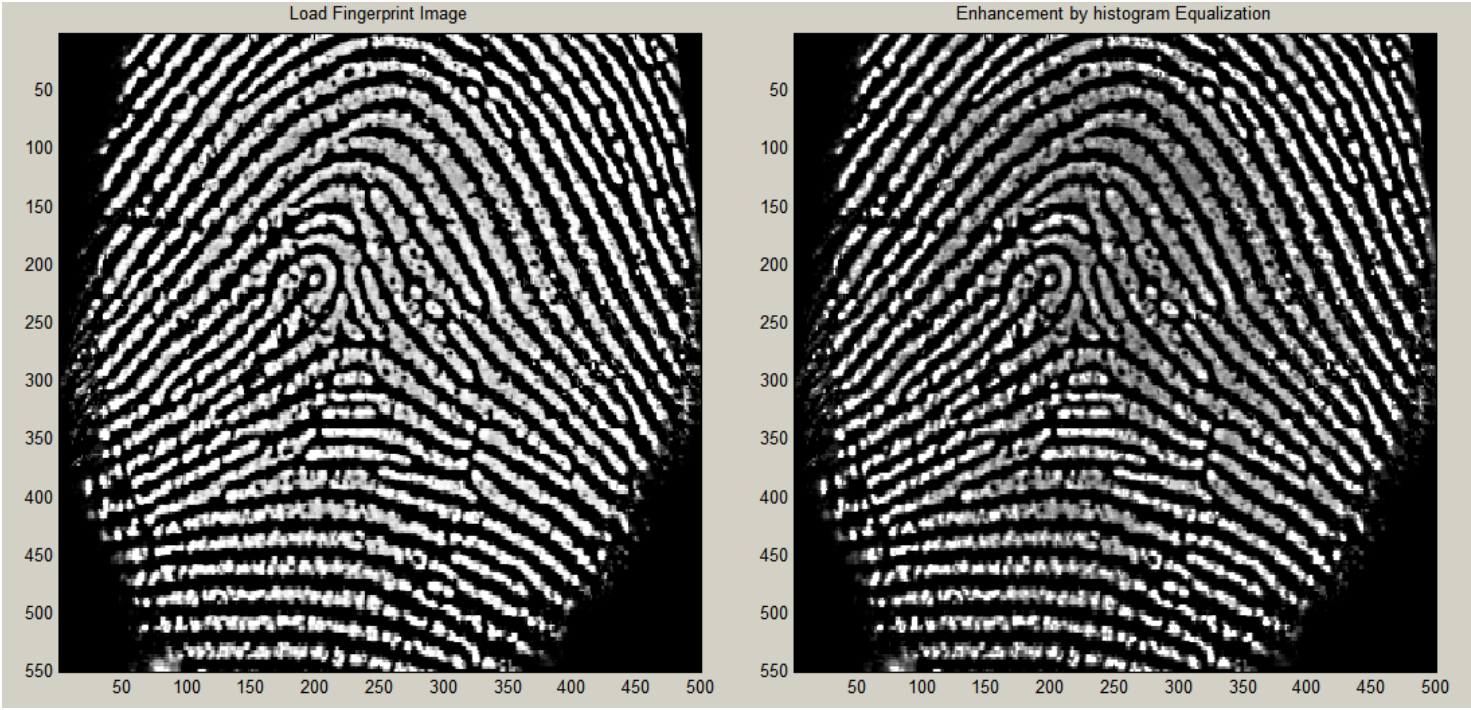


Figure 4.1.1.3 Histogram Enhancement.

Original Image (Left). Enhanced image (Right)

4.1.2 Fingerprint Enhancement by Fourier Transform

We divide the image into small processing blocks (32 by 32 pixels) and perform the Fourier transform according to:

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

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

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

We get the enhanced block according to

$$g(x,y) = F^{-1}\left\{F(u,v) \times |F(u,v)|^{K}\right\} \quad (2)$$

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

$$f(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u,v) \times \exp\left\{j2\pi \times \left(\frac{ux}{M} + \frac{vy}{N}\right)\right\} \quad (3)$$

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

The k in formula (2) is an experimentally determined constant, which we choose 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 [4.1.2.1] presents the image after FFT enhancement.

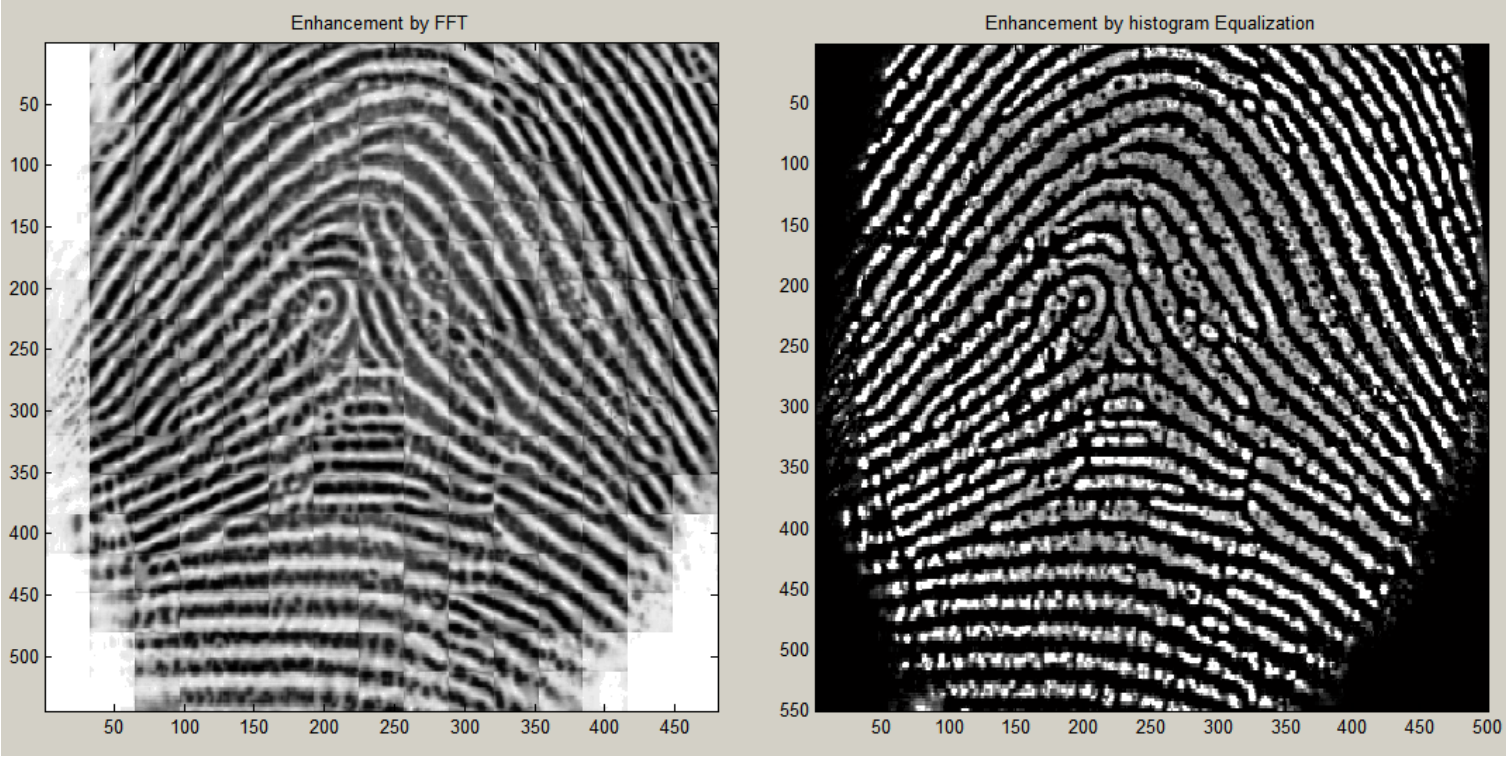


Figure 4.1.2.1 Fingerprint enhancement by FFT
Enhanced image (left), Original image (right)

The enhanced image after FFT has the improvements to connect some falsely broken points on ridges and to remove some false connections between ridges.

4.2 Fingerprint Image Binarization

The binarization step is basically stating the obvious, which is that the true information that could be extracted from a print is simply binary; ridges vs. valleys. But it is a really important step in the process of ridge extracting, since the prints are taken as grayscale images, so ridges, knowing that they're in fact ridges, still vary in intensity. So, binarization transforms the image from a 256-level image to a 2-level image that gives the same information.

Typically, an object pixel is given a value of "1" while a background pixel is given a value of "0." Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's label (black for 0, white for 1).

The difficulty in performing binarization is that not all the fingerprint images have the same contrast characteristics, so a single intensity threshold (global thresholding) cannot be chosen.

A locally adaptive binarization method is performed to binarize the fingerprint image. In this method, the image is divided into blocks (16x16), and the mean intensity value is calculated for each block, then each pixel is turned into 1 if its intensity value is larger than the mean intensity value of the current block to which the pixel belongs.

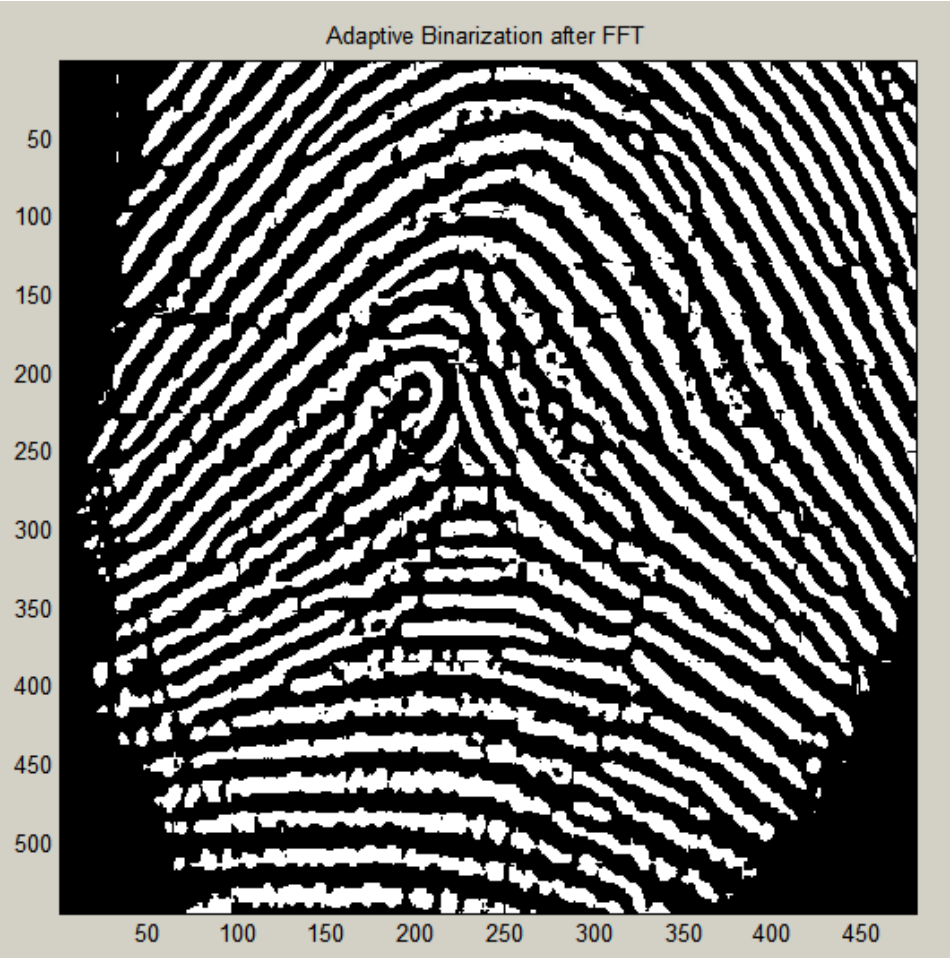


Figure 4.2.1 the Fingerprint image after adaptive binarization

4.3 Fingerprint Image Segmentation (orientation flow estimate)

In general, only a Region of Interest (ROI) is useful to be recognized for each fingerprint image. The image area without effective ridges is first discarded since it only holds background information and probably noise. Then the bound of the remaining effective area is sketched out since the minutiae in the bound region are confusing with those false minutiae that are generated when the ridges are out of the sensor.

To extract the ROI, a two-step method is used. The first step is block direction estimation and direction variety check, while the second is done using some Morphological methods.

4.3.1 Block direction estimation

1.1 Estimate the block direction for each block of the fingerprint image with WxW in size (W is 16 pixels by default). The algorithm does the following:

I. Calculates the gradient values along x-direction (g_x) and y-direction (g_y) for each pixel of the block. Two Sobel filters are used to fulfill the task.

II. For each block, it uses the following formula to get the Least Square approximation of the block direction.

$\tan 2ß = 2 \sum \sum (g_x * g_y) / \sum \sum (g_x^2 - g_y^2)$ for all the pixels in each block.

The formula is easy to understand by regarding gradient values along x-direction and y-direction as cosine value and sine value. So the tangent value of the block direction is estimated nearly the same as the way illustrated by the following formula.

$\tan 2\theta = 2\sin\theta \cos\theta / (\cos^2\theta - \sin^2\theta)$

1.2 After finishing with the estimation of each block direction, those blocks without significant information (ridges) are discarded based on the following formulas:

$E = \{2 \sum \sum (g_x * g_y) + \sum \sum (g_x^2 - g_y^2)\} / W * W * \sum \sum (g_x^2 + g_y^2)$

For each block, if its certainty level (E) is below a threshold, then the block is regarded as a background block.

The direction map is shown in the following diagram. We assume there is only one fingerprint in each image.

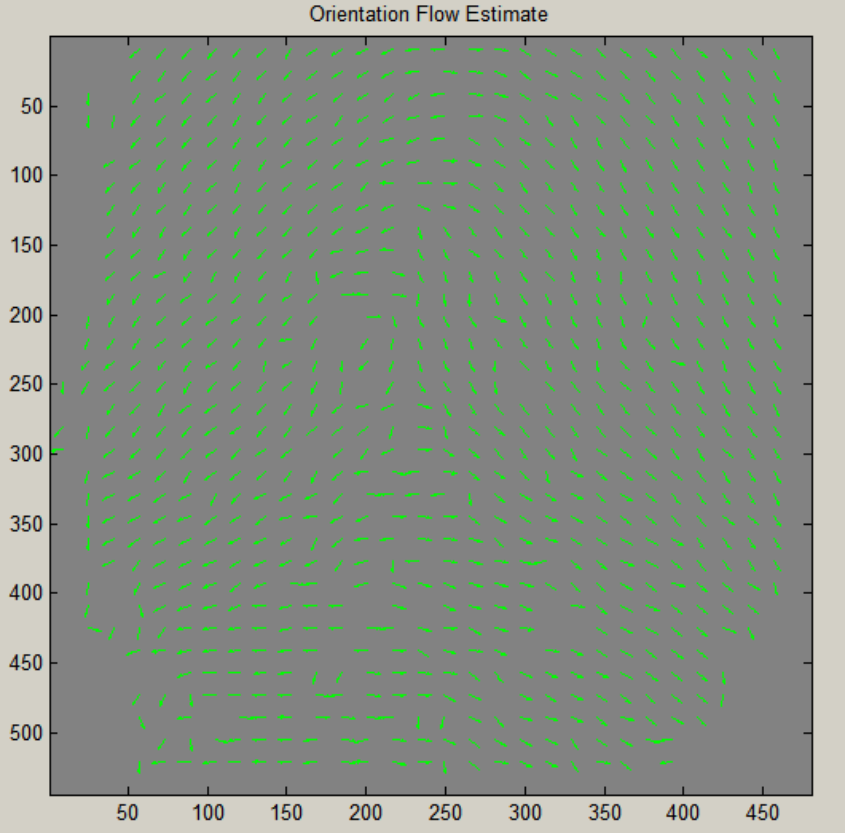


Figure 4.3.1.1 Direction map.

3.3.2 ROI extraction by Morphological operation

Two Morphological operations called ‘OPEN’ and ‘CLOSE’ are adopted. The ‘OPEN’ operation can expand images and remove peaks introduced by background noise. The ‘CLOSE’ operation can shrink images and eliminate small cavities.

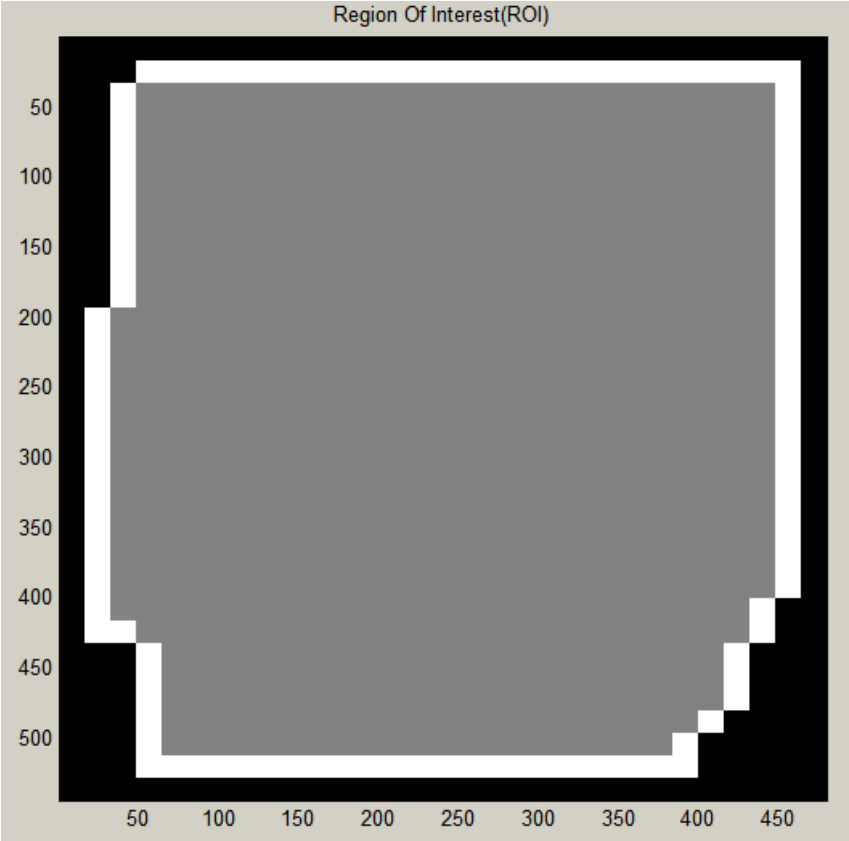


Figure 4.3.2.1 ROI + Bound

Figure [4.3.2.1] shows the interest fingerprint image area and its bound. The bound is the subtraction of the closed area from the opened area. Then the algorithm eliminates those leftmost, rightmost, uppermost and bottommost blocks out of the bound so as to get the tightly bounded region just containing the bound and inner area.

Chapter five: Minutiae extraction

5.1 Fingerprint Ridge Thinning

Ridge Thinning is to eliminate the redundant pixels of ridges till the ridges are just one pixel wide. An iterative, parallel thinning algorithm is used. In each scan of the full fingerprint image, the algorithm marks down redundant pixels in each small image window (3x3) and finally removes all those marked pixels after several scans. The thinned ridge map is then filtered by other Morphological operations to remove some H breaks, isolated points and spikes. In this step, any single points, whether they are single-point ridges or single-point breaks in a ridge are eliminated and considered processing noise.

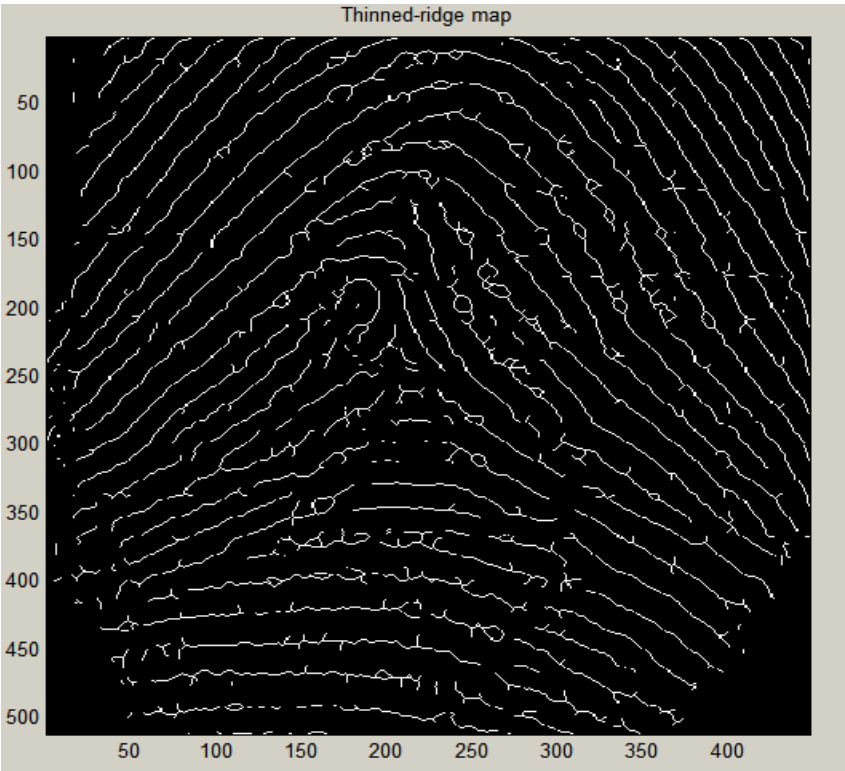


Figure 5.1.1 Thinned image

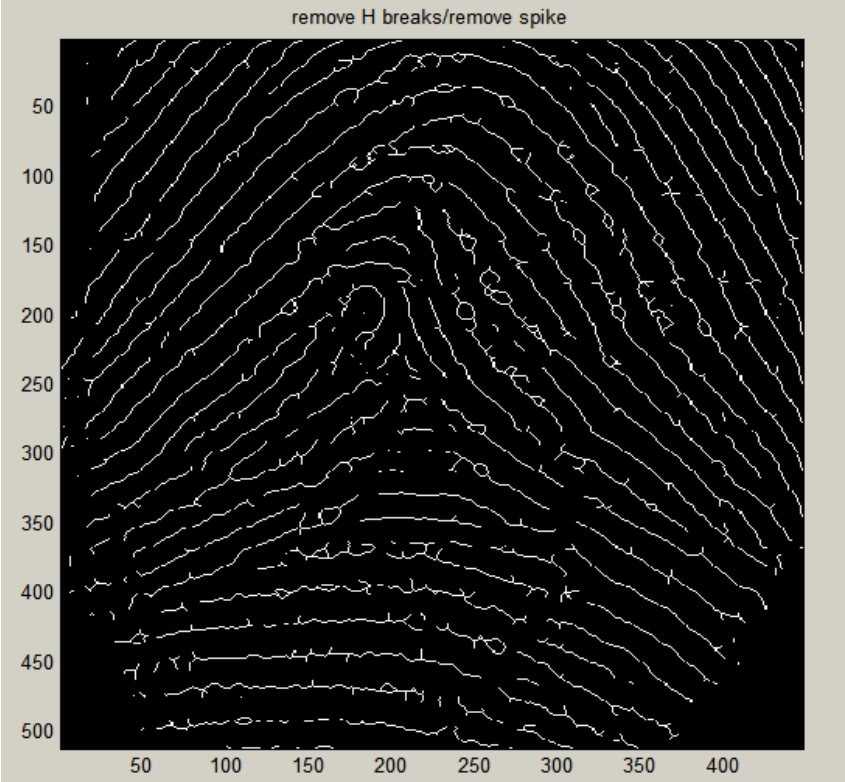


Figure 5.1.2 image after removing H-breaks and spikes

5.2 Minutia Marking

After the fingerprint ridge thinning, marking minutia points is relatively easy. The concept of Crossing Number (CN) is widely used for extracting the minutiae.

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 4.2.1]. If the central pixel is 1 and has only 1 one-value neighbor, then the central pixel is a ridge ending [Figure4.2.2], i.e., for a pixel P, if Cn(P) = = 1 it's a ridge end and if Cn(P) = = 3 it's a ridge bifurcation point.

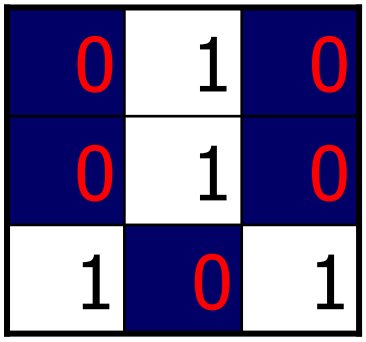


Figure 5.2.1 Bifurcation

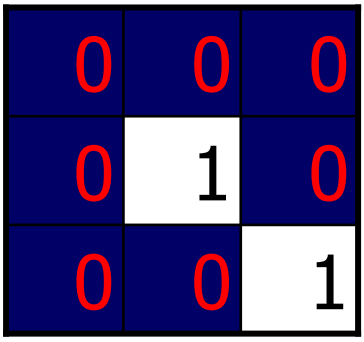


Figure 5.2.2 Termination

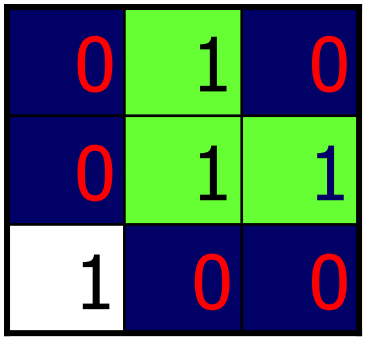


Figure 5.2.3 Triple counting branch

Figure 5.2.3 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, but actually only one branch is located in the small region. So a check routine requiring that none of the neighbors of a branch are branches is added.

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 values are one. Then divide the row length by 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.

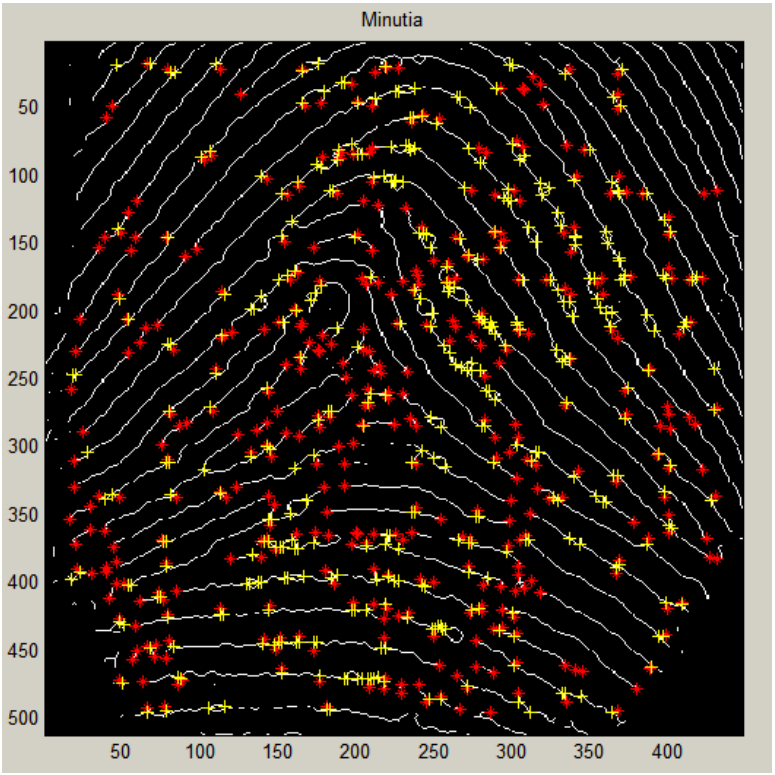


Figure 5.2.4 image after minutiae marking

Chapter six: Minutiae post-processing

False Minutia Removal

The preprocessing stage does not usually fix the fingerprint image in total. 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 minutiae. So some mechanisms of removing false minutia are essential to keep the fingerprint verification system effective.

Seven types of false minutia are specified in following diagrams:

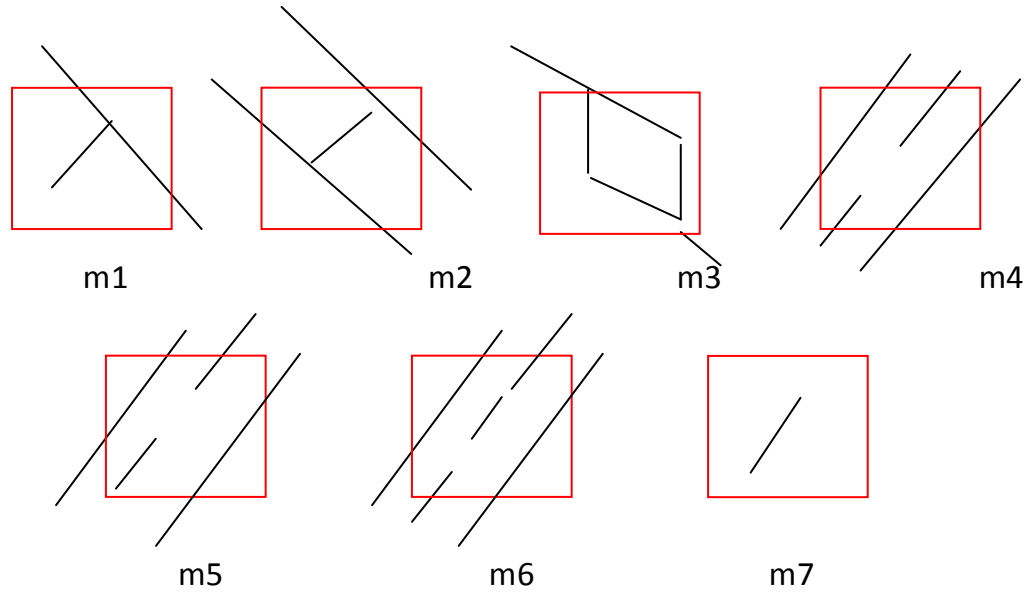


Figure 6.1 False Minutia Structures. m1 is a spike piercing into a valley. In the m2 case a spike falsely connects two ridges. m3 has two near bifurcations located in the same ridge. The two ridge broken points in the m4 case have nearly the same orientation and a short distance. m5 is alike the m4 case with the exception that one part of the broken ridge is so short that another termination is generated. m6 extends the m4 case but with the extra property that a third ridge is found in the middle of the two parts of the broken ridge. m7 has only one short ridge found in the threshold window.

The procedure for the removal of false minutia consists of the following steps:

1. If the distance between one bifurcation and one termination is less than D and the two minutiae are in the same ridge (m1 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. (m2, m3 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 minutiae derived from a broken ridge and are removed. (Cases m4,m5 & m6).
4. If two terminations are located in a short ridge with length less than D, remove the two terminations (m7).

This 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. The second advantage is that the order of removal procedures is well considered to reduce the computation complexity because it utilizes the relations among the false minutia types. For example, the procedure3

solves the m4, m5 and m6 cases in a single check routine. And after procedure 3, the number of false minutia satisfying the m7 case is significantly reduced.

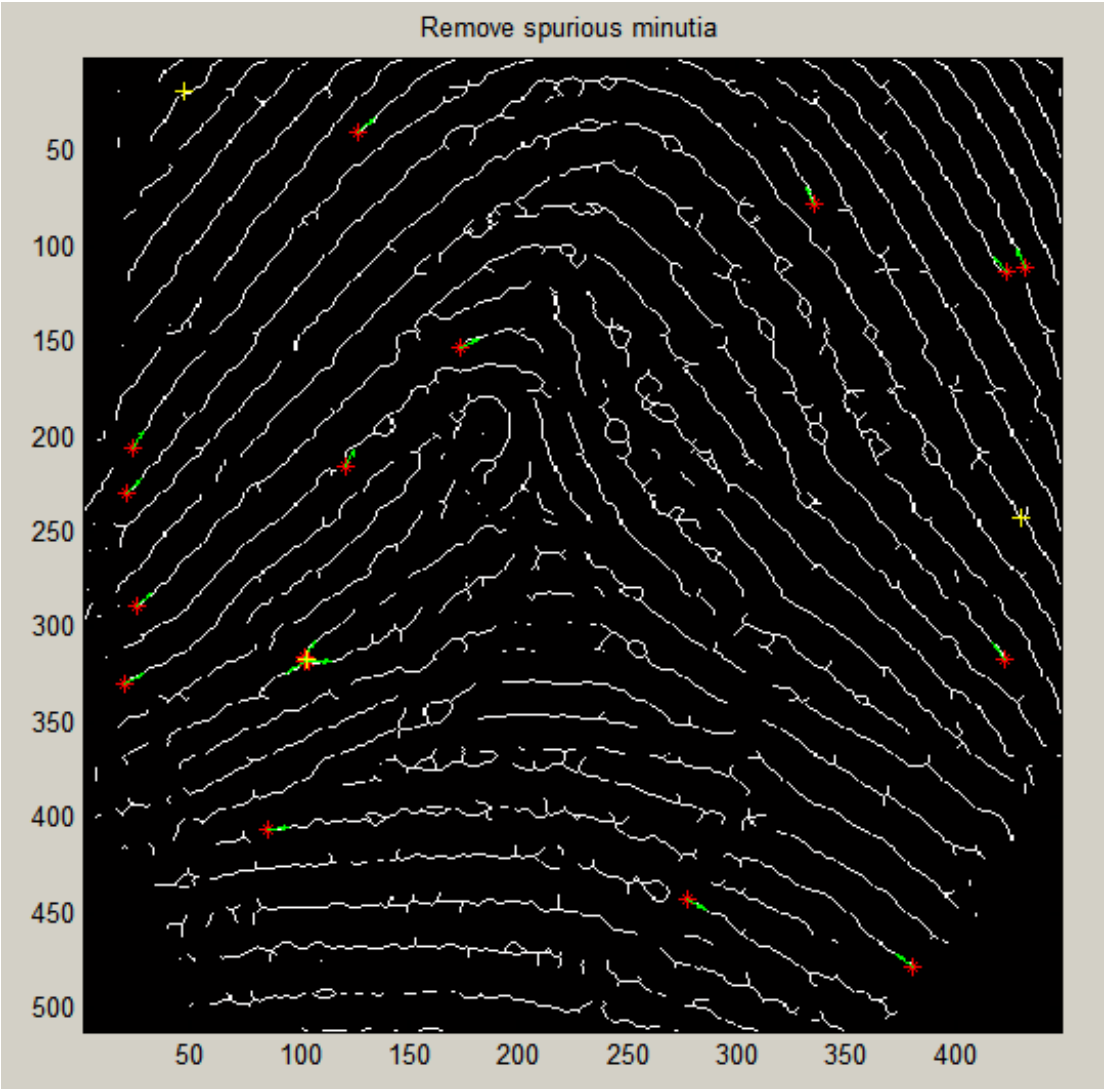


Figure 6.2 image after removing false minutiae

Chapter seven: Minutiae match

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 is used in my project. 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 reference point and whose x-axis is coincident with the direction of the referenced point.
2. **Match stage:** After we get two set of transformed minutia points, we use the elastic match algorithm to count the matched minutia pairs by assuming two minutia having nearly the same position and direction are identical.

6.1 Alignment Stage

1. The ridge associated with each minutia is represented as a series of x-coordinates (x_1, x_2…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:

$$S = \sum^{m}_{i=0} x_i X_i / [\sum^{m}_{i=0} x_i^2 X_i^2]^{0.5},$$

where (x_i~x_n) and (X_i~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 go to step 2, otherwise continue to match the next pair of ridges.

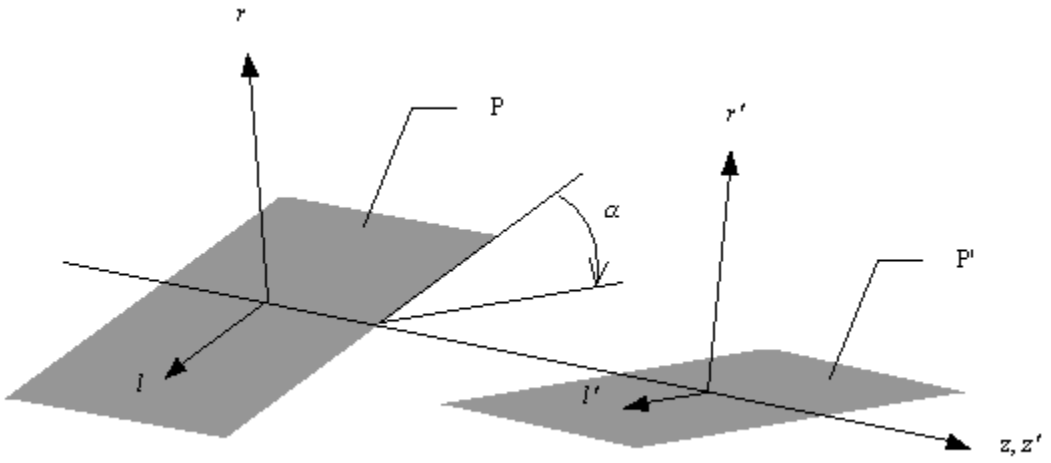
2. For each fingerprint, translate and rotate all other minutia with respect to the reference minutia according to the following formula:

$$\begin{pmatrix} xi\_new \\ yi\_new \\ \theta i\_new \end{pmatrix} = TM * \begin{bmatrix} (xi - x) \\ (yi - y) \\ (\theta i - \theta) \end{bmatrix}$$

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

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

The following diagram illustrates the rotation of the coordinate system according to the reference minutia's orientation:



This method uses the rotation angle calculated earlier by tracing a short ridge start from the minutia with length D. And since the rotation angle is already calculated and saved along with the coordinates of each minutiae, then this saves some processing time. The following step is to transform each minutia according to its own reference minutia and then match them in a unified x-y coordinate.

6.2 Match Stage

The matching algorithm for the aligned minutia patterns needs to be adaptive since the strict match requires that all parameters (x, y, θ) are the same for two identical minutiae which is impossible to get when using biometric-based matching.

This 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 difference 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 pairs divided by the number of minutia of the template fingerprint. The score is 100*ratio and ranges from 0 to 100. If the score is larger than a pre-specified threshold (typically 80%), the two fingerprints are from the same finger.

Chapter eight: System evaluation and conclusion

8.1 Evaluation of the system

As we can see in the graph shown below, when eliminating a step from the whole process or changing some of the parameters, the matching process is affected.

Observations:

1. When altering in such an important step such as the image enhancement part, the performance quality of the system drops rapidly as the noise in the image is increased. Because when working with a biometric identification system, obtaining clear and noise free images is a really hard thing, so this step is usually needed.
2. For the binarization step, as explained earlier, using global thresholding may introduce a few problems and may lead to the elimination of significant details by mistake. Here, I tried using global thresholding, with 2 different thresholds, once using an intensity threshold of 120 and the second time using a value of 80. As we can see from the graph, setting the threshold at 120 (although it's almost the average value for a gray-scale image) affected the system performance a lot and led to false non-match results, while setting a fixed threshold as low as 80 gave better results. Still, it remains better to use the adaptive threshold method because, although it consumes more processing time, it still guarantees the quality of the results.

3. If we try to remove the H-breaks step, the system wouldn't be greatly affected and the matching process wouldn't become harder, but it's considered a preprocessing step and it doesn't add much complexity to the system, so no harm in keeping the accuracy higher.

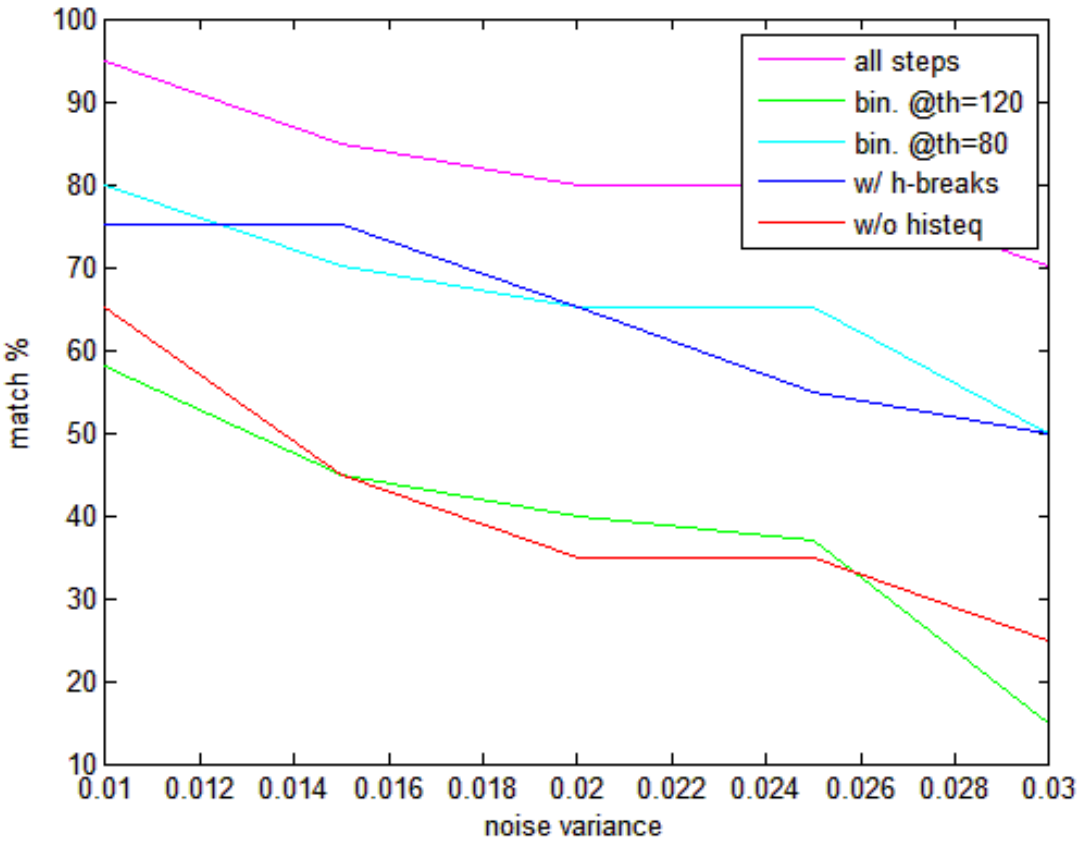


Figure 8.1.1 match percentage vs. noise variance

8.2 Conclusion

The reliability of any automatic fingerprint system strongly relies on the precision obtained in the minutia extraction process. A number of factors damage the correct location of minutia. Among them, poor image quality is the one with most influence.

The proposed alignment-based elastic matching algorithm is capable of finding the correspondences between minutiae without resorting to exhaustive research.

There is a scope of further improvement in terms of efficiency and accuracy which can be achieved by improving the hardware to capture the image or by improving the image enhancement techniques. So that the input image to the thinning stage could be made better, this could improve the future stages and the final outcome

Appendix

Program no.1

(Image Enhancement)

```
function [final]=fftenhance(image,f)
I = 255-double(image);
[w,h] = size(I);
%out = I;
w1=floor(w/32)*32;
h1=floor(h/32)*32;
inner = zeros(w1,h1);
for i=1:32:w1
  for j=1:32:h1
    a=i+31;
    b=j+31;
    F=fft2( I(i:a,j:b) );
    factor=abs(F).^f;
    block = abs(ifft2(F.*factor));
    larv=max(block(:));
    if larv==0
```

```
        larv=1;
      end;
      block= block./larv;
      inner(i:a,j:b) = block;
    end;
end;
final=inner*255;
final=histeq(uint8(final));
```

Program no.2

(Image Binarization)

```
function [o] = adaptiveThres(a,W,noShow);
%Adaptive thresholding is performed by segmenting image a
[w,h] = size(a);
o = zeros(w,h);
%seperate it to W block
%step to w with step length W
for i=1:W:w
for j=1:W:h
mean_thres = 0;
if i+W-1 <= w & j+W-1 <= h
    mean_thres = mean2(a(i:i+W-1,j:j+W-1));
      mean_thres = 0.8*mean_thres;
```

```
        o(i:i+W-1,j:j+W-1) = a(i:i+W-1,j:j+W-1) < mean_thres;
    end;
  end;
end;
if nargin == 2
imagesc(o);
colormap(gray);
end;
```

Program no.3

(for Block Direction Estimation)

```
function [p,z] = direction(image,blocksize,noShow)
%image=adaptiveThres(image,16,0);
[w,h] = size(image);
direct = zeros(w,h);
gradient_times_value = zeros(w,h);
gradient_sq_minus_value = zeros(w,h);
gradient_for_bg_under = zeros(w,h);
W = blocksize;
theta = 0;
sum_value = 1;
bg_certainty = 0;
blockIndex = zeros(ceil(w/W),ceil(h/W));
```

```
    %directionIndex = zeros(ceil(w/W),ceil(h/W));
    times_value = 0;
    minus_value = 0;
    center = [];
    filter_gradient = fspecial('sobel');
    %to get x gradient
    I_horizontal = filter2(filter_gradient,image);
    %to get y gradient
    filter_gradient = transpose(filter_gradient);
    I_vertical = filter2(filter_gradient,image);
    gradient_times_value=I_horizontal.*I_vertical;
    gradient_sq_minus_value=(I_vertical-
I_horizontal).*(I_vertical+I_horizontal);
    gradient_for_bg_under = (I_horizontal.*I_horizontal) +
(I_vertical.*I_vertical);
    for i=1:W:w
       for j=1:W:h
         if j+W-1 < h & i+W-1 < w
       times_value = sum(sum(gradient_times_value(i:i+W-1, j:j+W-1)));
           minus_value = sum(sum(gradient_sq_minus_value(i:i+W-1, j:j+W-
1)));
           sum_value = sum(sum(gradient_for_bg_under(i:i+W-1, j:j+W-1)));
```

```
         bg_certainty = 0;
        theta = 0;

        if sum_value ~= 0 & times_value ~=0
          %if sum_value ~= 0 & minus_value ~= 0 & times_value ~= 0
bg_certainty = (times_value*times_value +
minus_value*minus_value)/(W*W*sum_value);
                    if bg_certainty > 0.05
          blockIndex(ceil(i/W),ceil(j/W)) = 1;
         %tan_value = atan2(minus_value,2*times_value);
          tan_value = atan2(2*times_value,minus_value);
         theta = (tan_value)/2 ;
theta = theta+pi/2;
          center = [center;[round(i + (W-1)/2),round(j + (W-1)/2),theta]];
       end;
      end;
   end;
times_value = 0;
        minus_value = 0;
        sum_value = 0;
       end;
      end;
```

```
if nargin == 2
imagesc(direct);
        hold on;
[u,v] = pol2cart(center(:,3),8);
  quiver(center(:,2),center(:,1),u,v,0,'g');
  hold off;
end;
x = bwlabel(blockIndex,4);
y = bwmorph(x,'close');
z = bwmorph(y,'open');
p = bwperim(z);
```

Program no.4

(to extract ROI)

```
function [roiImg,roiBound,roiArea] = drawROI(in,inBound,inArea,noShow)
[iw,ih]=size(in);
tmplate = zeros(iw,ih);
[w,h] = size(inArea);
tmp=zeros(iw,ih);
left = 1;
right = h;
upper = 1;
bottom = w;
```

```
    le2ri = sum(inBound);
    roiColumn = find(le2ri>0);
    left = min(roiColumn);
    right = max(roiColumn);
    tr_bound = inBound';
    up2dw=sum(tr_bound);
    roiRow = find(up2dw>0);
    upper = min(roiRow);
    bottom = max(roiRow);
    %cut out the ROI region image
    %show background,bound,innerArea with different gray
intensity:0,100,200
    for i = upper:1:bottom
      for j = left:1:right
        if inBound(i,j) == 135
          tmplate(16*i-15:16*i,16*j-15:16*j) = 200;
          tmp(16*i-15:16*i,16*j-15:16*j) = 1;
    elseif inArea(i,j) == 1 & inBound(i,j) ~=1
          tmplate(16*i-15:16*i,16*j-15:16*j) = 100;
          tmp(16*i-15:16*i,16*j-15:16*j) = 1;
        end;
      end;
```

```
end;

in=in.*tmp;

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

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

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

%inner area

roiArea = im2double(roiArea) - im2double(roiBound);

if nargin == 3

colormap(gray);

imagesc(tmplate);

end;
```

Program no.5

(Ridge Thinning)

```
function edgeDistance =RidgeThin(image,inROI,blocksize)

[w,h] = size(image);

a=sum(inROI);

b=find(a>0);

c=min(b);

d=max(b);

i=round(w/5);

m=0;

for k=1:4
```

```
    m=m+sum(image(k*i,16*c:16*d));
  end;
  e=(64*(d-c))/m;
  a=sum(inROI,2);
  b=find(a>0);
  c=min(b);
  d=max(b);
  i=round(h/5);
  m=0;
  for k=1:4
    m=m+sum(image(16*c:16*d,k*i));
  end;
  m=(64*(d-c))/m;
  edgeDistance=round((m+e)/2);
  Program no. 6
  (Minutia marking)
  function [end_list,branch_list,ridgeOrderMap,edgeWidth] =
mark_minutia(in,
  inBound,inArea,block);
  [w,h] = size(in);
  [ridgeOrderMap,totalRidgeNum] = bwlabel(in);
  imageBound = inBound;
```

```
imageArea = inArea;

blkSize = block;

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

edgeWidth = interRidgeWidth(in,inArea,blkSize);

end_list    = [];

branch_list = [];

for n=1:totalRidgeNum

  [m,n] = find(ridgeOrderMap==n);

  b = [m,n];

  ridgeW = size(b,1);

    for x = 1:ridgeW

    i = b(x,1);

    j = b(x,2);

 %ifimageArea(ceil(i/blkSize),ceil(j/blkSize))==1&

imageBound(ceil(i/blkSize),ceil(j/blkSize)) ~= 1

if inArea(ceil(i/blkSize),ceil(j/blkSize)) == 1

    neiborNum = 0;

    neiborNum = sum(sum(in(i-1:i+1,j-1:j+1)));

    neiborNum = neiborNum -1;

  if neiborNum == 1

  end_list =[end_list; [i,j]];

 elseif neiborNum == 3
```

```
        %if two neighbors among the three are connected directly
        %there may be three braches are counted in the nearing three cells
        tmp=in(i-1:i+1,j-1:j+1);
        tmp(2,2)=0;
        [abr,bbr]=find(tmp==1);
        t=[abr,bbr];
        if isempty(branch_list)
          branch_list = [branch_list;[i,j]];
        else
       for p=1:3
          cbr=find(branch_list(:,1)==(abr(p)-2+i) & branch_list(:,2)==(bbr(p)-2+j)
);
          if ~isempty(cbr)
            p=4;
            break;
          end;
        end;

        if p==3
          branch_list = [branch_list;[i,j]];
        end;
        end;
```

```
  end;
        end;
        end;
       end;
```

Program no.7

(False Minutia removal)

```
function [pathMap, final_end,final_branch]
=remove_spurious_Minutia(in,end_list,branch_list,inArea,ridgeOrderMap,
edgeWidth
[w,h] = size(in);
final_end = [];
final_branch =[];
direct = [];
pathMap = [];
end_list(:,3) = 0;
branch_list(:,3) = 1;
minutiaeList = [end_list;branch_list];
finalList = minutiaeList;
[numberOfMinutia,dummy] = size(minutiaeList);
suspectMinList = [];
for i= 1:numberOfMinutia-1
  for j = i+1:numberOfMinutia
```

```
        d =( (minutiaeList(i,1) - minutiaeList(j,1))^2 + (minutiaeList(i,2)-
minutiaeList(j,2))^2)^0.5;


        if d < edgeWidth
          suspectMinList =[suspectMinList;[i,j]];
        end;
      end;
    end;
    [totalSuspectMin,dummy] = size(suspectMinList);
    for k = 1:totalSuspectMin
      typesum = minutiaeList(suspectMinList(k,1),3) +
minutiaeList(suspectMinList(k,2),3)
      if typesum == 1
        % branch - end pair
        if
ridgeOrderMap(minutiaeList(suspectMinList(k,1),1),minutiaeList(suspectMinLi
st(k,1),2) )
    ==
ridgeOrderMap(minutiaeList(suspectMinList(k,2),1),minutiaeList(suspectMinLi
st(k,2),2) )
          finalList(suspectMinList(k,1),1:2) = [-1,-1];
        finalList(suspectMinList(k,2),1:2) = [-1,-1];
```

```
        end;
      elseif typesum == 2
        % branch - branch pair
        if
ridgeOrderMap(minutiaeList(suspectMinList(k,1),1),minutiaeList(suspectMinLi
st(k,1),2) )
    ==
ridgeOrderMap(minutiaeList(suspectMinList(k,2),1),minutiaeList(suspectMinLi
st(k,2),2) )
          finalList(suspectMinList(k,1),1:2) = [-1,-1];
        finalList(suspectMinList(k,2),1:2) = [-1,-1];
        end;
      elseif typesum == 0
        % end - end pair
        a = minutiaeList(suspectMinList(k,1),1:3);
        b = minutiaeList(suspectMinList(k,2),1:3);
        if ridgeOrderMap(a(1),a(2)) ~=  ridgeOrderMap(b(1),b(2))
          [thetaA,pathA,dd,mm] = getLocalTheta(in,a,edgeWidth);
          [thetaB,pathB,dd,mm] = getLocalTheta(in,b,edgeWidth);
          %the connected line between the two point
          thetaC = atan2( (pathA(1,1)-pathB(1,1)), (pathA(1,2) - pathB(1,2)) );
          angleAB = abs(thetaA-thetaB);
```

```
            angleAC = abs(thetaA-thetaC);
            if ( (or(angleAB < pi/3, abs(angleAB -pi)<pi/3 )) & (or(angleAC < pi/3,
abs(angleAC - pi)
    < pi/3)) )
              finalList(suspectMinList(k,1),1:2) = [-1,-1];
              finalList(suspectMinList(k,2),1:2) = [-1,-1];
            end;
            %remove short ridge later
          elseif  ridgeOrderMap(a(1),a(2)) ==  ridgeOrderMap(b(1),b(2))
            finalList(suspectMinList(k,1),1:2) = [-1,-1];
            finalList(suspectMinList(k,2),1:2) = [-1,-1];
          end;
        end;
    end;
      for k =1:numberOfMinutia
        if finalList(k,1:2) ~= [-1,-1]
          if finalList(k,3) == 0
            [thetak,pathk,dd,mm] = getLocalTheta(in,finalList(k,:),edgeWidth);
            if size(pathk,1) >= edgeWidth
             final_end=[final_end;[finalList(k,1:2),thetak]];
    [id,dummy] = size(final_end);
    pathk(:,3) = id;
```

```
                pathMap = [pathMap;pathk];
              end;
            else
              final_branch=[final_branch;finalList(k,1:2)];
              [thetak,path1,path2,path3] =
getLocalTheta(in,finalList(k,:),edgeWidth);
              if size(path1,1)>=edgeWidth & size(path2,1)>=edgeWidth &
size(path3,1)>=edgeWidth
              final_end=[final_end;[path1(1,1:2),thetak(1)]];
              [id,dummy] = size(final_end);
              path1(:,3) = id;
              pathMap = [pathMap;path1];

              final_end=[final_end;[path2(1,1:2),thetak(2)]];
              path2(:,3) = id+1;
             pathMap = [pathMap;path2];
              final_end=[final_end;[path3(1,1:2),thetak(3)]];
    path3(:,3) = id+2;
              pathMap = [pathMap;path3];
             end;
         end;
        end;
```

```
  end;
```

Program no.8

(Alignment stage)

```
function [newXY] = MinuOriginTransRidge(real_end,k,ridgeMap
    theta = real_end(k,3);
    if theta <0
theta1=2*pi+theta;
end;
theta1=pi/2-theta;
    rotate_mat=[cos(theta1),-sin(theta1);sin(theta1),cos(theta1)];
    %locate all the ridge points connecting to the miniutia
    %and transpose it as the form:
    %x1 x2 x3...
    %y1 y2 y3...
    pathPointForK = find(ridgeMap(:,3)== k);
    toBeTransformedPointSet =
ridgeMap(min(pathPointForK):max(pathPointForK),1:2)';
     %translate the minutia position (x,y) to (0,0)
    %translate all other ridge points according to the basis
    tonyTrickLength = size(toBeTransformedPointSet,2);
    pathStart = real_end(k,1:2)';
```

```
        translatedPointSet = toBeTransformedPointSet -
pathStart(:,ones(1,tonyTrickLength)
        %rotate the point sets
        newXY = rotate_mat*translatedPointS
    function [newXY] = MinuOrigin_TransAll(real_end,k)
    theta = real_end(k,3);
    if theta <0
    theta1=2*pi+theta;
    end;
    theta1=pi/2-theta;
    rotate_mat=[cos(theta1),-sin(theta1),0;sin(theta1),cos(theta1),0;0,0,1];
        toBeTransformedPointSet = real_end';
        tonyTrickLength = size(toBeTransformedPointSet,2);
        pathStart = real_end(k,:)';
        translatedPointSet = toBeTransformedPointSet -
pathStart(:,ones(1,tonyTrickLength));
        newXY = rotate_mat*translatedPointSet;
       %ensure the direction is in the domain[-pi,pi]
        for i=1:tonyTrickLength
          if or(newXY(3,i)>pi,newXY(3,i)<-pi)
             newXY(3,i) = 2*pi - sign(newXY(3,i))*newXY(3,i);
         end;
```

```
    end;
```

Program no.9

(Minutiae matching)

```
function [newXY] = MinuOrigin_TransAll(real_end,k)
theta = real_end(k,3);
if theta <0
theta1=2*pi+theta;
end;
theta1=pi/2-theta;
rotate_mat=[cos(theta1),-sin(theta1),0;sin(theta1),cos(theta1),0;0,0,1];
    toBeTransformedPointSet = real_end';
    tonyTrickLength = size(toBeTransformedPointSet,2);
    pathStart = real_end(k,:)';
    translatedPointSet = toBeTransformedPointSet -
pathStart(:,ones(1,tonyTrickLength));
    newXY = rotate_mat*translatedPointSet;
      for i=1:tonyTrickLength
      if or(newXY(3,i)>pi,newXY(3,i)<-pi)
        newXY(3,i) = 2*pi - sign(newXY(3,i))*newXY(3,i);
      end;
    end;
```

REFERENCES

1. "Digital Image processing using MATLAB" -by Steven L. Eddins
2. "Handbook of fingerprint recognition" -Davide Maltoni, Dario Maio, Anil K. Jain, Salil Prabhakar
3. Journal of Electronic Imaging/Mehmet Sezgin and Bulent Sankur; Survey over image thresholding techniques and quantitative performance evaluation/Jan.2004
4. Computer science/International Journal of Image Processing (IJIP)/Dec.2010 (CSCjournals.org)
5. Interpol's public records/History of fingerprints (http://www.interpol.int/Public/Forensic/fingerprints/History/BriefHistoricOutline.pdf)
6. Course notes from the Swiss Federal Institute of Technology/speech processing and biometric group (http://scgwww.epfl.ch/)