A

MAJOR PROJECT

ON

FINGERPRINT RECOGNITION

Submitted in partial fulfilment of the requirement
for the award of the degree of

MASTER OF TECHNOLOGY

IN

(VLSI & EMBEDDED SYSTEM)

Submitted by

AJAY KUMAR
(Roll No. 02/vlsi/09)
2009-2011

Under the Guidance of

SMT. S. INDU
(Associate Professor)

Department of Electronics & Communication

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(FORMERLY DELHI COLLEGE OF ENGINEERING)
BAWANA ROAD, DELHI-110042

CERTIFICATE

This is to certify that the thesis entitled **"FINGERPRINT RECOGNITION"** being submitted by **Mr.Ajay kumar**, in the partial fulfilment of the requirement for the degree of Master of Technology in VLSI Design & Embedded System in the Department of Electronics & Communication, Delhi Technological University, Delhi is a bonafide record of the research work done by him under my supervision and guidance. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree to the best of my knowledge and belief.

Mrs. S. Indu
Associate Professor
Deptt. of Electronics & Comm.
Delhi Technological University

ACKNOWLEDGEMENT

I am highly grateful to my project guide, **Smt. S. Indu**, for her support and encouragement. I am greatly indebted to her for the many occasions in which she has gone out of her way to assist me in the timely completion of this dissertation. She has been an ongoing inspiration for me not only in terms of academic endeavours, but her charismatic personality has also made my studies a much more interesting and rewarding experience. I would also like to take the opportunity to present my sincere regards to our Head of Department **Prof**. **Rajiv Kapoor** for his support and encouragement. I am also thankful to my friends for their unconditional support and motivation during this work.

(Ajay Kumar)
Roll no. 02/vlsi/09
M.Tech.(Vlsi & Embedded system)
Deptt. of Electronics & Communication
Delhi Technological University

ABSTRACT

The popular Biometric used to authenticate a person is Fingerprint which is unique and permanent throughout a person's life. A minutia matching is widely used for fingerprint recognition and can be classified as ridge ending and ridge bifurcation. Automatic minutiae detection is an extremely critical process, especially in low-quality fingerprints where noise and contrast deficiency can originate pixel configurations similar to minutiae or hide real minutiae. The technique describe here, is consists of minutiae extraction and minutiae matching. the minutia extraction algorithm proposed here, is much faster and more reliable and implemented for extracting features from an input fingerprint image. For minutia matching, an alignment-based elastic matching algorithm has been developed. The proposed algorithm results in an efficient, robust and fast representation of fingerprints which accurately retains the fidelity in minutiae (ridge endings and bifurcations).

CONTENTS

CHAPTER 1

INTRODUCTION

Introduction

Fingerprint based identification has been known and used for a very long time wing to their uniqueness and immutability, fingerprints are today the most widely used biometric features. Most automatic systems for fingerprint comparison are based on minutiae matching. Minutiae characteristics are local discontinuities in the fingerprint pattern. This report, is based on a two-class minutiae classification: termination and bifurcation.

For each minutia we store the membership class, the coordinates and the angle that the tangent to the minutia forms with the horizontal direction . The problem of automatic minutiae extraction has been thoroughly studied but never completely solved. The main difficulty is that fingerprint quality is often too low, noise and contrast deficiency can produce false minutiae and hide valid minutiae.

Several approaches to automatic minutiae extraction have been proposed: although rather different from one other, most of these methods transform fingerprint images into binary images through an ad hoc algorithm. The images obtained are submitted to a thinning process which allows for the ridge line thickness to be reduced to one pixel.. Finally, a simple image scan allows for locating the pixels that correspond to minutiae.

Then image enhancement and the use of an adaptive threshold, aimed to preserve the same number of 1 and 0 pixels for each neighbourhood, form the basis of the binarization technique a technique for enhancement and binarization based on the convolution of the image with a filter oriented according to the directional image. The directional image may be conceived as a matrix whose elements represent the tangent direction to the ridge lines of the original image.

A binary ridge image is an image where all the ridge pixels are assigned a value one and nonridge pixels are assigned a value zero. The binary image can be obtained by

applying a ridge extraction algorithm on a gray-level fingerprint image. Since ridges and valleys in a fingerprint image alternate and run parallel to each other in a local neighbourhood, a number of simple heuristics can be used to differentiate the spurious ridge configurations from the true ridge configurations in a binary ridge image. However, after applying a ridge extraction algorithm on the original gray-level images,
information about the true ridge structures is often lost depending on the performance of the ridge extraction algorithm. Therefore, enhancement of binary ridge images has its inherent limitations. In a gray-level fingerprint image, ridges and valleys in a local neighbourhood form a sinusoidal-shaped plane wave which has a well-defined frequency and orientation. A number of techniques that take advantage of this information have been proposed to enhance gray-level fingerprint images. However, they usually assume that the local ridge orientations can be reliably estimated.

In practice, this assumption is not valid for fingerprint images of poor quality, which greatly restricts the applicability of these techniques. Hong et al. [19] proposed a decomposition method to estimate the orientation field from a set of filtered images obtained by applying a bank of Gabor filters on the input fingerprint images. Although this algorithm can obtain a reliable orientation estimate even for
corrupted images.

After the binerization thining process is done and then it become easy to find the minutie marking points, 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.

My proposed procedures in removing false minutia have 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 [26] that does not utilize the relations among the false minutia types. Since various data acquisition conditions such as impression pressure can easily change one type of minutia into the other, most researchers adopt the unification representation for both termination and bifurcation. So each minutia is completely characterized by the following parameters

at last: 1) x-coordinate, 2) y-coordinate, and 3) orientation.

The orientation calculation for a bifurcation needs to be specially considered. All three ridges deriving from the bifurcation point have their own direction, [24] represents the bifurcation orientation using a technique proposed]. [15] simply chooses the minimum angle among the three anticlockwise orientations starting from the x-axis. Both methods cast the other two directions away, so some information loses. Here I propose a novel representation to break a bifurcation into three terminations. The three new terminations are the three neighbour pixels of the bifurcation and each of the three ridges connected to the bifurcation before is now associated with a termination respectively

My approach to elastically 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 minutia 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 fingerprints.

1.1 Application Scenrios

1. Automatic Fingerprint Authentication System
2. Fingerprint recognition technique is used by Investigation departments

CHAPTER 2

RELATED WORKS

RELATED WORK

G. Sambasiva Rao et al., [1] proposed fingerprint identification technique using a gray level watershed method to find out the ridges present on a fingerprint image by directly scanned fingerprints or inked impression.

Robert Hastings [2] developed a method for enhancing the ridge pattern by using a process of oriented diffusion by adaptation of anisotropic diffusion to smooth the image in the direction parallel to the ridge flow. The image intensity varies smoothly as one traverse along the ridges or valleys by removing most of the small irregularities and breaks but with the identity of the individual ridges and valleys preserved.

Jinwei Gu*,* et al., [3] proposed a method for fingerprint verification which includes both minutiae and model based orientation field is used. It gives robust discriminatory information other than minutiae points. Fingerprint matching is done by combining the decisions of the matchers based on the orientation field and minutiae.

V. Vijaya Kumari and N. Suriyanarayanan [4] proposed a method for performance measure of local operators in fingerprint by detecting the edges of fingerprint images using five local operators namely Sobel, Roberts, Prewitt, Canny and LoG. The edge detected image is further segmented to extract individual segments from the image.

Raju Sonavane, and B.S. Sawant [5] presented a method by introducing a special domain fingerprint enhancement method which decomposes the fingerprint image into a set of filtered images then orientation field is estimated. A quality mask distinguishes the recoverable and unrecoverable corrupted regions in the input image are generated. Using the estimated orientation field, the input fingerprint image is adaptively enhanced in the recoverable regions.

Eric P. Kukula, et al., [6] purposed a method to investigate the effect of five different force levels on fingerprint matching performance, image quality scores, and minutiae count between optical and capacitance fingerprint sensors. Three images were

collected from the right index fingers of 75 participants for each sensing technology. Descriptive statistics, analysis of variance, and Kruskal-Wallis nonparametric tests were conducted to assess significant differences in minutiae counts and image quality scores based on the force level. The results reveal a significant difference in image quality score based on the force level and each sensor technology, yet there is no significant difference in minutiae count based on the force levels of the capacitance sensor. The image quality score, shown to be effected by force and sensor type, is one of many factors that influence the system matching performance.

R. Girgisa et al., [7] proposed a method to describe a fingerprint matching based on lines extraction and graph matching principles by adopting a hybrid scheme which consists of a genetic algorithm phase and a local search phase. Experimental results demonstrate the robustness of algorithm.

Duoqian Maio et al., [8] used principal graph algorithm by kegl to obtain principal curves for auto fingerprint identification system. From principal curves, minutiae extraction algorithm is used to extract the minutiae of the fingerprint. The experimental results shows curves obtained from graph algorithm are smoother than the thinning algorithm. The fingerprint features such as singular points, positions and direction of core and delta obtained from a binarised fingerprint image. The method is producing good classification results.

Lie Wei [9] proposed a method for rapid singularities searching algorithm which uses delta field Poincare index and a rapid classification algorithm to classify the fingerprint in to 5 classes. The detection algorithm searches the direction field which has the larger direction changes to get the singularities. Singularities detection is used to increase the accuracy.

Hartwig Fronthaler, et al., [10] Proposed fingerprint enhancement to improve the matching performance and computational efficiency by using an image scale pyramid and directional filtering in the spatial domain. The proposed novel fingerprint image post-processing algorithm makes an efforts to reliably differentiate spurious minutiae from true ones by making use of ridge number information, referring to original gray-

level image, designing and arranging various processing techniques properly, and also selecting various processing parameters carefully. The proposed post-processing algorithm is effective and efficient

Luping Ji, and Zhang Yi [11] proposed a method for estimating four direction orientation field by considering four steps, i) preprocessing fingerprint image, ii) determining the primary ridge of fingerprint block using neuron pulse coupled neural network, iii) estimating block direction by projective distance variance of a ridge, instead of a full block, iv) correcting the estimated orientation field

Manvjeet Kaur et al., [12] have introduced combined methods to build a minutia extractor and a minutia matcher. Segmentation with Morphological operations used to improve thinning, false minutiae removal, minutia marking.

Haiping Lu et al., [13] proposed an effective and efficient algorithm for minutiae extraction to improve the overall performance of an automatic fingerprint identification system because it is very important to preserve true minutiae while removing spurious minutiae in post-processing. The proposed novel fingerprint image post-processing algorithm makes an efforts to reliably differentiate spurious minutiae from true ones by making use of ridge number information, referring to original gray-level image, designing and arranging various processing techniques properly, and also selecting various processing parameters carefully. The proposed post-processing algorithm is effective and efficient.

Prabhakar S, Jain. A.K. et al., [14] has developed filter-based representation technique for fingerprint identification. The technique exploits both local and global characteristics in a fingerprint to make identification. Each fingerprint image is filtered in a number of directions and a 640-dimensinal feature vector is extracted in the central region of the fingerprint. The feature vector is compact and requires only 640 bytes. The matching stage computes the Euclidian distance between the template finger code and the input finger code. The method gives good matching with high accuracy.

CHAPTER 3

FINGERPRINT RECOGNITION

FINGERPRINT RECOGNITION

Fingerprints can be represented by a large number of features, including the overall ridge flow pattern, location, direction , and location of minutia points, ridge counts between pairs of minutiae, and location of pores. All these features contribute to fingerprint individuality.

In this study, I choose minutia representation of the fingerprints because it is utilized by forensic experts, it has been demonstrated to be relatively stable and it has been adopted by most of the commercially available automatic fingerprint.

Each individual has unique fingerprints. The uniqueness of a fingerprint is exclusively determined by the local ridge characteristics and their relationships These local ridge characteristics are not evenly distributed. Most of them depend heavily on the impression conditions and quality of fingerprints and are rarely observed in fingerprints. There are two most prominent local ridge characteristics, called minutiae and these are :

1) ridge ending
2) ridge bifurcation.

A ridge ending is defined as the point where a ridge ends abruptly or it is also known as termination and ridge bifurcation is defined as the point where a ridge forks or diverges into branch ridges.

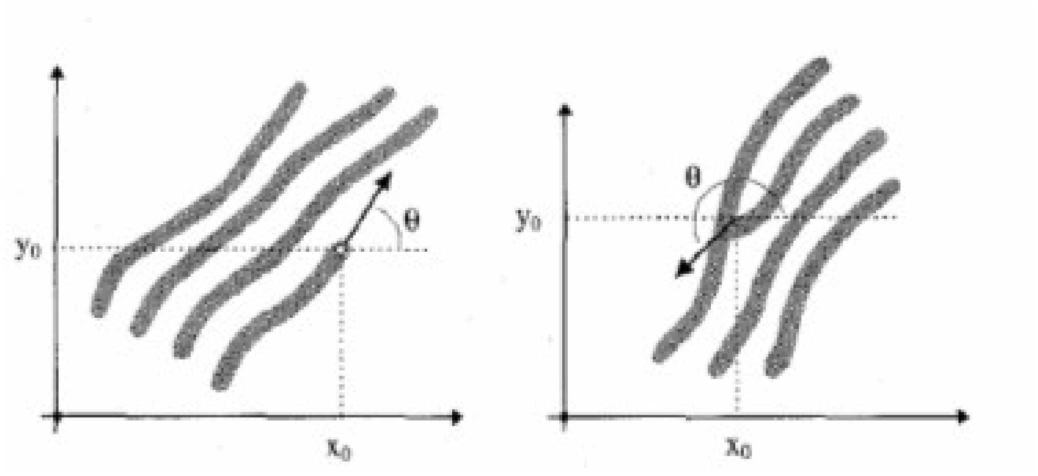


Fig 3.1 Minutiae representation, Fig 3.1.a shows a termination minutiae: (x_0,y_0) are the minutiae coordinates; Θ is the angle that the minutiae tangent forms with the horizontal direction. Fig 3.1.b shows a bifurcation minutiae: Θ is now defined by means of the termination minutiae existing in the complementary image in correspondence with the original bifurcation

The fingerprint recognition are used for two sub-domains: one is fingerprint verification and the other is fingerprint identification (Figure 3.2). In addition, different from the manual approach for fingerprint recognition by experts, the fingerprint recognition here is referred as AFRS (Automatic Fingerprint Recognition System), which is program-based.

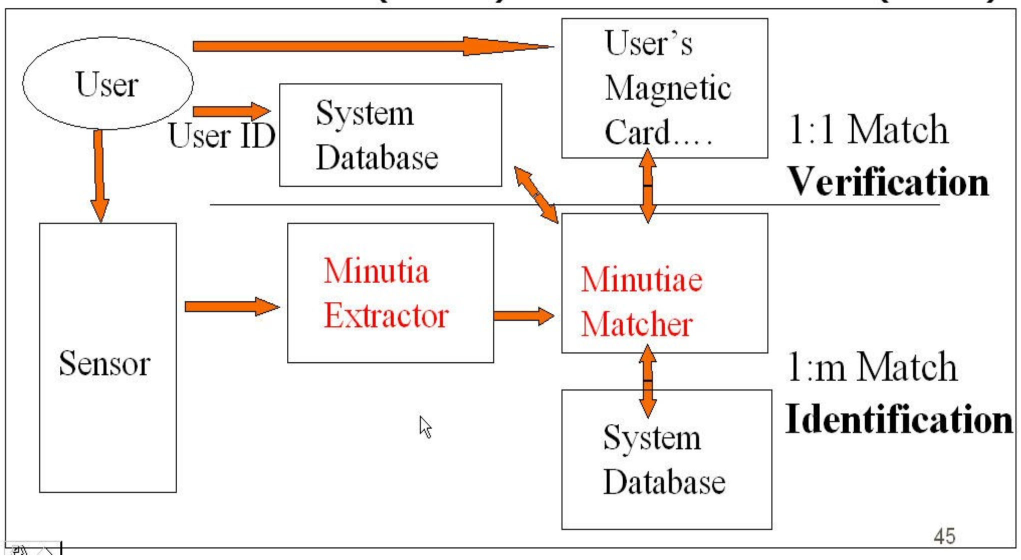


Figure 3.2. Verification vs. Identification

Fingerprint verification is used to verify the authenticity of one person by his fingerprint. The user provides his fingerprint together with his identity information like his 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).

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

However, all fingerprint recognition, 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.

Algorithm used:

For obtaining minutiae extractor of fingerprints, I use a three stage approach that is widely used by researchers. This three stage approach consist of preprocessing, minutia extraction and postprocessing stage.

Preprocessing stage :

Image Enhancement

Image Binerization

Image Segmentation

Minutia Extraction:

Thinning

Minutiae Marking

Postprocessing:

Remove false minutia

Histogram Equalization and Fourier Transform are used for image enhancement by me in the preprocessing stage of the fingerprints recognition. And then binarization has been done to the fingerprint image by using the locally adaptive threshold method . Then the task of image segmentation is done by a three-step approach: block direction estimation, segmentation by direction intensity and Region of Interest extraction by Morphological operations. Most methods used in the preprocessing stage are developed by other researchers but they form a new combination in my project. Also the morphological operations for extraction ROI are introduced to fingerprint image segmentation.

In minutia extraction stage, again three thinning algorithms are tested and the Morphological thinning operation is finally bid out with high efficiency and pretty good thinning quality. The minutia marking is a simple task as most literatures reported but one special case is found during my implementation is enforced to avoid such kind of oversight.

For the postprocessing stage, a more rigorous algorithm is developed to remove false minutiae. A novel representation for bifurcations is proposed to unify terminations and bifurcations.

Minutiae Matcher:

Align two fingerprint images

Minutiae match

Any two minutia are chosen as a reference minutia pair for matching and then match their associated ridges first. If the ridges match well, two fingerprint images are aligned and matching is conducted for all remaining minutia.

3.1 FINGERPRINT IMAGE PREPROCESSING

3.1.1 Fingerprint Image Enhancement

Fingerprint Image enhancement is the process to make the image clearer for easy further operations. Since the fingerprint images acquired from sensors or other medias are not assured with perfect quality, those enhancement methods, for increasing the contrast between ridges and furrows 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.

Two Methods are used in my fingerprint recognition system: the first one is Histogram Equalization, the second is Fast Fourier Transform.

Histogram Equalization:

Histogram equalization has been done as a mapping of gray levels p into gray levels q such that the distribution of gray level q is uniform. 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 has the bimodal type [Figure 3.1.1.1]

This mapping stretches contrast (expands the range of gray levels) for gray levels near the histogram maxima. Since contrast is expanded for most of the image pixels, the transformation improves the detectability of many image features.

The probability density function of a pixel intensity level is r_k given by:

$$P_r(r_k) = n_k / n \qquad (1)$$

where:, k == 0,1, ... ,255 ,n $_k$ is the number of pixels at intensity level r $_k$ and n is the total number of pixels. The histogram is derived by plotting P $_r$ (r $_k$) against r $_k$. A new intensity s $_k$ of level k is defined as:

$$s_k = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} p_r(r_j) \qquad (2)$$

We apply the histogram equalization locally by using a local windows of 16 x16 pixels. This results in expanding the contrast locally, and changing the intensity of ·each pixel according to its local neighborhood. the histogram after the histogram equalization occupies all the range from 0 to 255 and the visualization effect is enhanced [Figure 3.1.1.2].

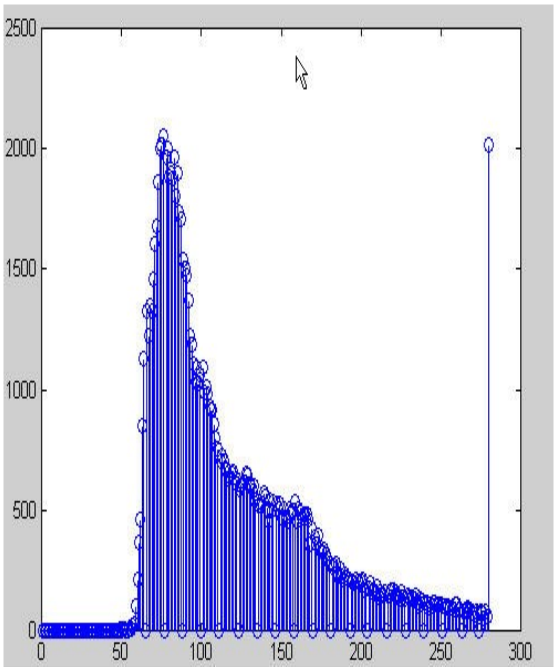


Figure 3.1.1 Original histogram of a fingerprint image

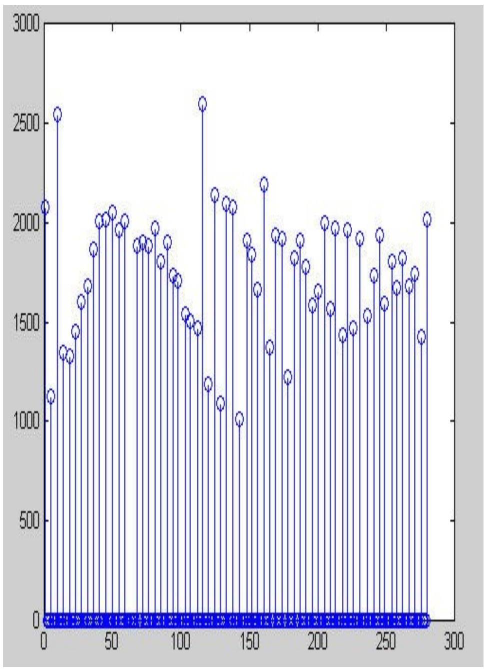


Figure 3.1.2 Histogram after the Histogram Equalization

The right side of the following figure [Figure 3.1.1.3] is the output after the histogram equalization.

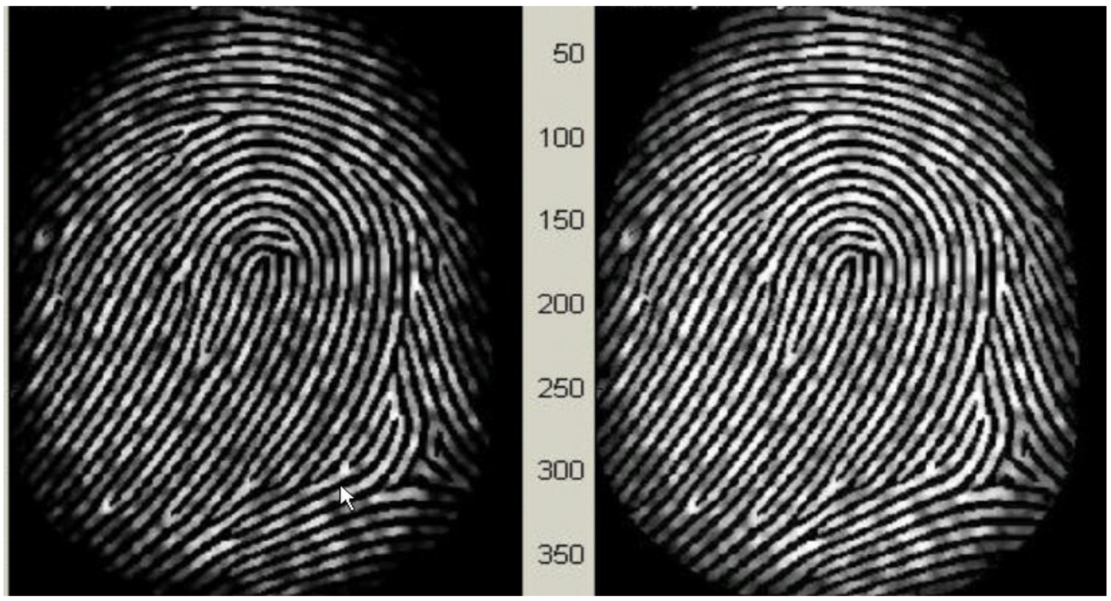


Figure 3.1.3 Histogram Enhancement.

Original Image (Left). Enhanced image (Right)

Fingerprint Enhancement by Fourier Transform

In this study, the image into small processing blocks (32 by 32 pixels) and perform the Fourier transform according to:

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

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)|.

Get the enhanced block according to

$$g(x,y) = F^{-1}\left\{F(u,v) \times \left|F(u,v)\right|^{K}\right\} \tag{4}$$

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

$$f(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u,v) \times \exp\left\{j2\pi \times \left(\frac{ux}{M} + \frac{vy}{N}\right)\right\} \tag{5}$$

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

The k in formula (4) 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 3.1.2.1 presents the image after FFT enhancement.

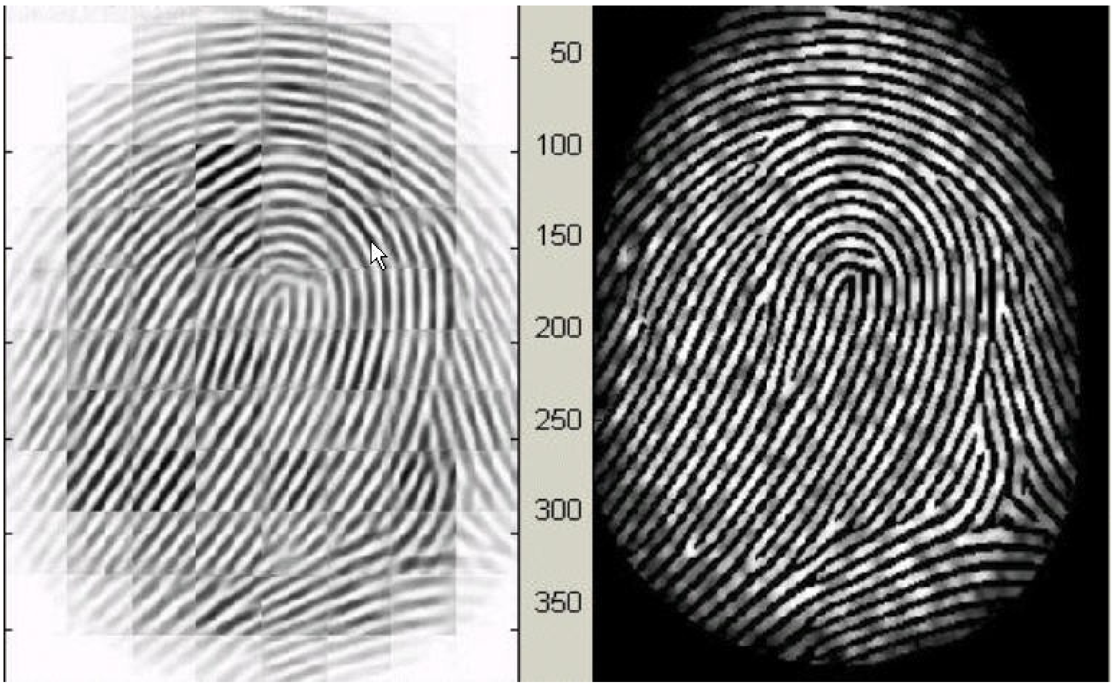


Figure 3.1.4 Fingerprint enhancement by FFT
Enhanced image (left), Original image (right)

The image enhanced after FFT has the improvements to connect some falsely broken points on ridges and to remove some spurious connections between ridges. The shown image at the left side of figure 3.1.2.1 is also processed with histogram equalization after the FFT transform.

3.1.2 Fingerprint Image Binarization

Fingerprint Image Binarization is used 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 binarization, ridges in the fingerprint are shown with black colour and furrows are with white colour.

Here 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 3.1.2.1].

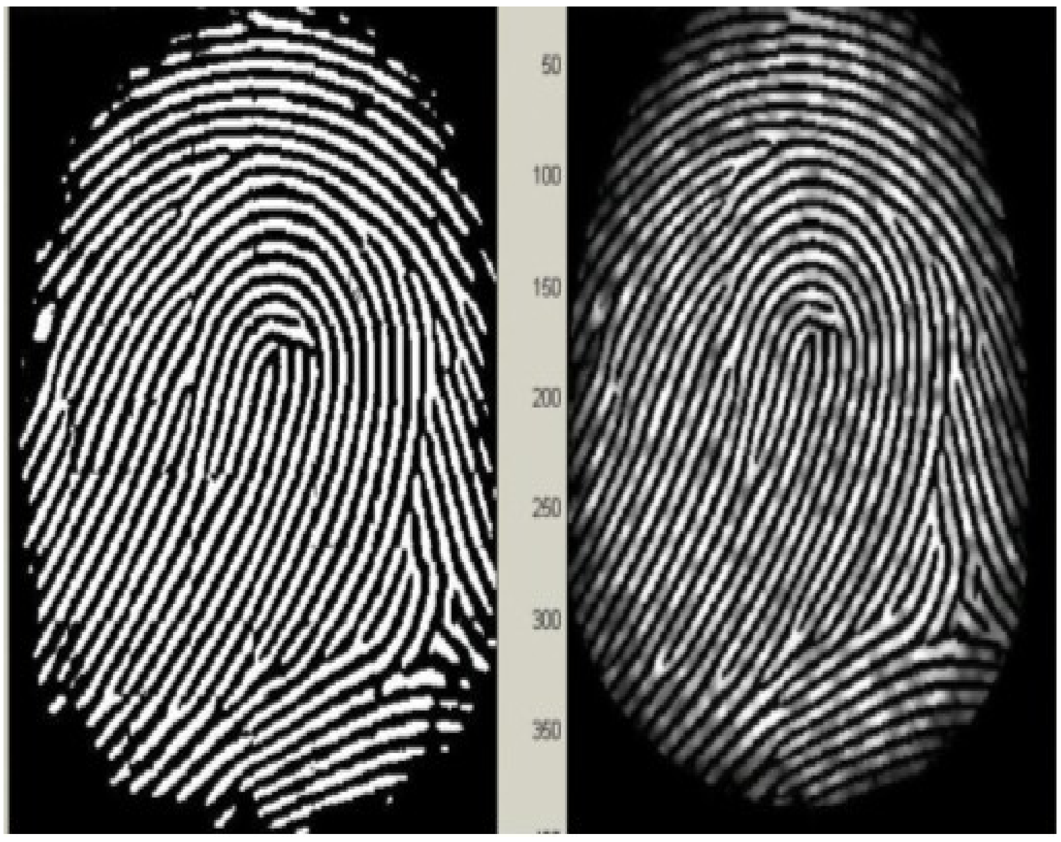


Figure 3.1.2.1 the Fingerprint image after adaptive binarization

Binarized image(left), Enhanced gray image(right)

3.1.3 Fingerprint Image Segmentation

Image contains the background informations also along and furrows, which are of no use so this background informations are discarded.. Then the bound of the remaining effective area is sketched out since the minutia in the bound region are confusing with those spurious minutia that are generated when the ridges are out of the sensor. In general, only a Region of Interest (ROI) is useful to be recognized for each fingerprint image.

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 intrigued from some Morphological methods.

Block direction estimation

1.1 The block direction for each block is estimated for fingerprint image with WxW in size(W is 16 pixels by default). The algorithm is worked as:

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

II. For each block, use Following formula to get the Least Square approximation of the block direction.

   $tg2ß = 2 \Sigma \Sigma (g_x * g_y) / \Sigma \Sigma (g_x^2 - g_y^2)$ 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.

$$tg2\Theta = 2\sin\Theta\cos\Theta / (\cos^2\Theta - \sin^2\Theta) \quad (6)$$

1.2 After finished with the estimation of each block direction, those blocks without significant information on ridges and furrows are discarded based on the following formulas:

$$E = \{2 \Sigma \Sigma (g_x * g_y) + \Sigma \Sigma (g_x^2 - g_y^2)\} / W*W* \Sigma \Sigma (g_x^2 + g_y^2) \quad (7)$$

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.

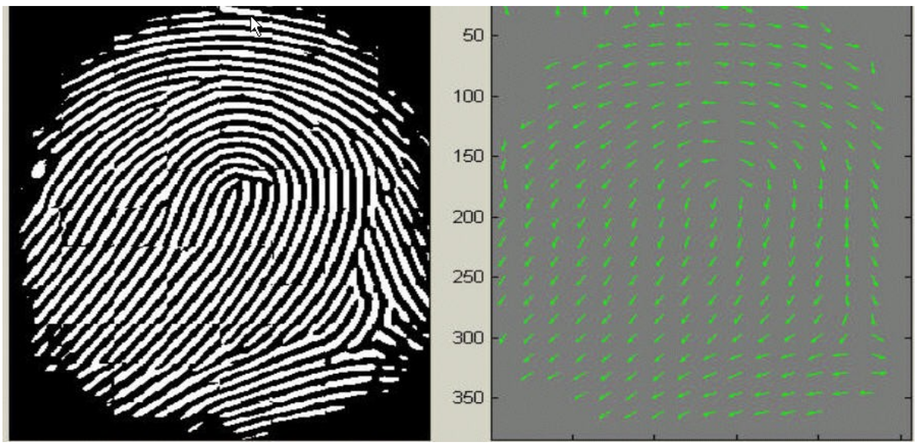


Figure 3.1.3.1 Direction map.

Binarized fingerprint (left), Direction map (right)

ROI extraction by Morphological operations

There are two Morphological operations called 'OPEN' and 'CLOSE' which are used here. The 'CLOSE' operation can shrink images and eliminate small cavities [Figure 3.1.3.3]. The 'OPEN' operation can expand images and remove peaks introduced by background noise [Figure 3.1.3.4].

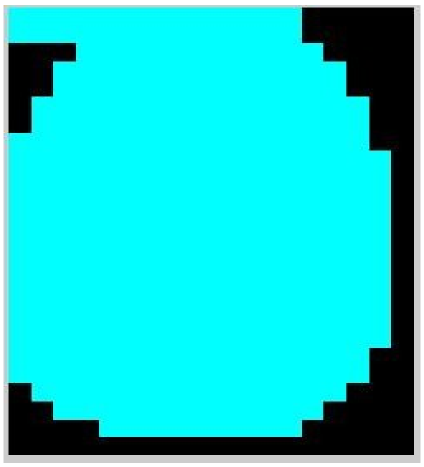


Figure 3.1.3.2 Original Image Area

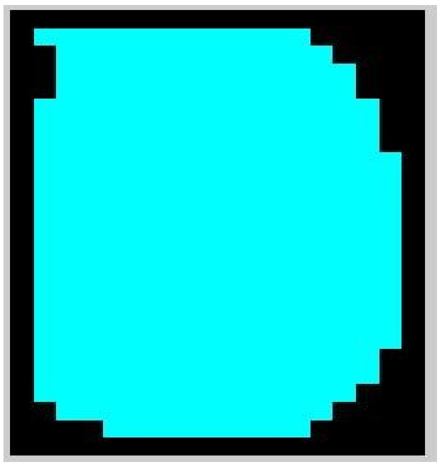
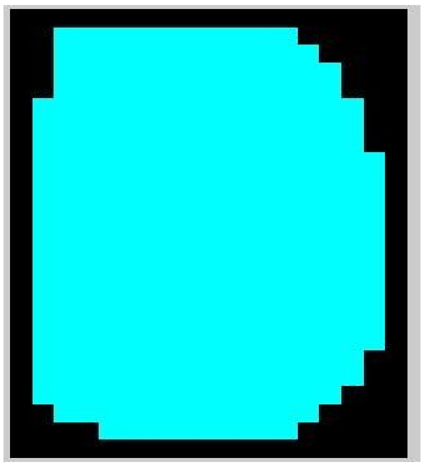


Figure 3.1.3.3 After CLOSE operation



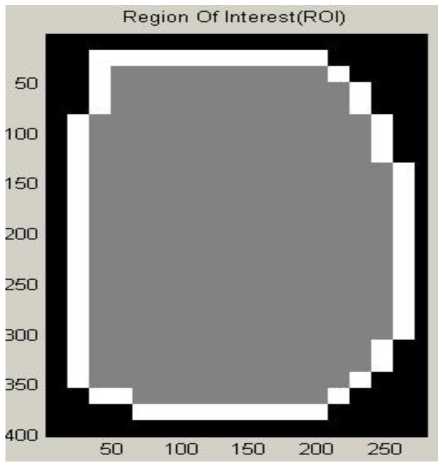


Figure 3.1.3.4 After OPEN operation   Figure 3.1.3.5 ROI + Bound

Figure 3.1.3.5 show the interest fingerprint image area and its bound. The bound is the subtraction of the closed area from the opened area. Then the algorithm throws away 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

3.2 MINUTIA EXTRACTION

3.2.1 Fingerprint Ridge Thinning

Ridge Thinning is to eliminate the redundant pixels of ridges till the ridges are just one pixel wide. [26] uses an iterative, parallel thinning algorithm. Thinning algorithm will be developed by means of mathmatical morphological, like a kind ***of*** non-linear image processing, for extracting a set of interest lines**],** obtaining the thinned ridge map desired. The basic definitions and properties of standard and soft morphological operations used for binary images are presented. The morphological thinning operator is the subtraction between the input image and the sup-generating operator with structuring elements A and B. The result will be the original image but with pixels which center contains the pattern specified by A and B marked as zero. This operation removes pixels which satisfy the pattern given by the structuring elements A and B. Skeleton by Thinning presents the construction of homotopic and non-homotopic skeletons through the recursive use of thinning operators. The skeleton by thinning is the infinite application of the Thinning operator using a sequence of structuring elements A and B generated **by** successive rotations of a specific pattern defined. By changing the pattern and the step used for its rotation, it is possible to generate a large number of thinning operators. Depending of the pattern and step used, the operator can be homotopic (preserve the topology of the original image) or not. 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. In my testing, such an iterative, parallel thinning algorithm has bad efficiency although it can get an ideal thinned ridge map after enough scans. [16] uses a one-in-all method to extract thinned ridges from gray-level fingerprint images directly. Their method traces along the ridges having maximum gray intensity value. However, binarization is implicitly enforced since only pixels with maximum gray intensity value are remained. Also in my testing, the advancement of each trace step still has large computation complexity although it does not require the movement of pixel by pixel as in other thinning algorithms.

3.2.2 Minutia Marking

Marking minutia points is relatively easy after the fingerprint ridge thinning. 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.

In general, for each 3x3 window, if the central pixel is 1 and has exactly 3 one-value neighbours, then the central pixel is a ridge branch [Figure 3.2.2.1]. If the central pixel is 1 and has only 1 one-value neighbour, then the central pixel is a ridge ending [Figure3.2.2.2].

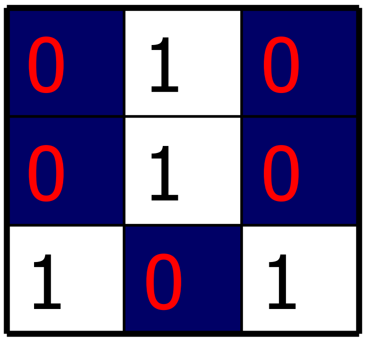


Figure 3.2.2.1 Bifurcation

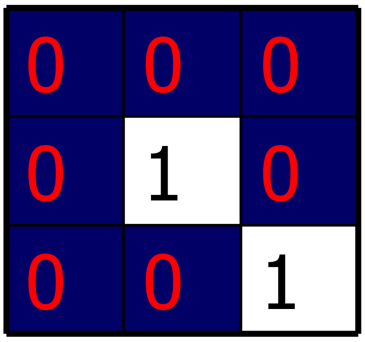


Figure 3.2.2.2 Termination

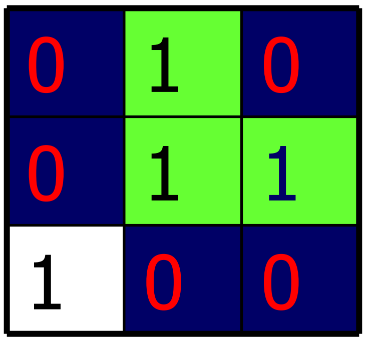


Figure 3.2.2.3 Triple counting branch

Figure 3.2.2.3 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 neighbour 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 neighbours 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 neighbouring 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.

All thinned ridges in the fingerprint image are labeled with a unique ID together with the minutia marking for further operation. The labeling operation is realized by using the Morphological operation.

3.3 MINUTIA POSTPROCESSING

3.3.1 False Minutia 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. The accuracy of matching will significantly affected by the false minutia if they are simply regarded as genuine minutia. So, to keep the fingerprint verification system effective, it is necessary to use some mechanisms for removing false minutia.

Seven types of false minutia are specified in following diagrams:

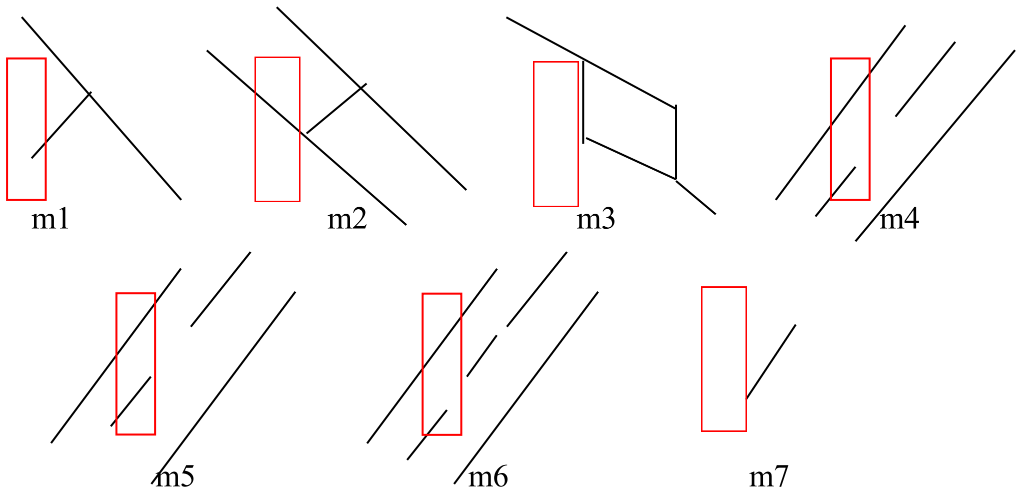


Figure 3.3.1. False Minutia Structures. m1 is a spike piercing into a valley. In the m2 case a spike falsely connects two ridges. m3 has two near bifurcations located in the same ridge. The two ridge broken points in the m4 case have nearly the same orientation and a short distance. m5 is alike the m4 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.

[18] only handles the case m1, m4,m5 and m6. [23] and [16] have not false minutia removal by simply assuming the image quality is fairly good. [26] has not a Systematic healing method to remove those spurious minutia although it lists all types of false minutia shown in Figure 3.3.1 except the m3 case.

Procedures for 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(m1 case) . Remove both of them. Where D is the average inter-ridge width representing the average distance between two parallel neighbouring 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 minutia derived from a broken ridge and are removed. (case m4, m5, m6).

4. If two terminations are located in a short ridge with length less than D, remove the two terminations (m7).

There are two advantages of my proposed procedure in removing false minutia. 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 [26] that does not utilize 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.

3.3.2 MINUTIA 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 partially derived from the [15] 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 that has 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 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.

3.3.3 Alignment Stage

1. 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:

$$S = \Sigma^{m}_{i=0} x_i X_i / [\Sigma^{m}_{i=0} x_i^2 X_i^2]^\wedge 0.5, \qquad (8)$$

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 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) is the parameters of the reference minutia, and TM is

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

The following diagram illustrate the effect of translation and rotation:

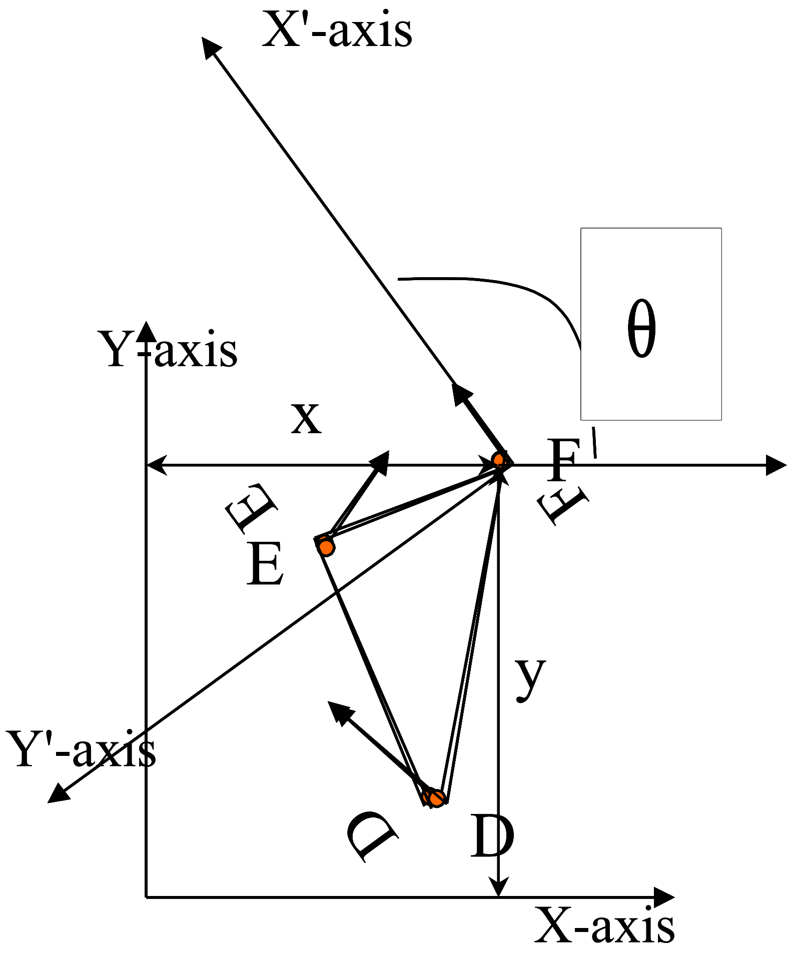


FIG.3.3.3.1 The new coordinate system is originated at minutia F and the new x-axis is coincident with the direction of minutia F. No scaling effect is taken into account by assuming two fingerprints from the same finger have nearly the same size.

Here the method is used to align two fingerprints that are almost same with the one used by [15] but is different at step 2. [15] uses the rotation angle calculated from all the sparsely sampled ridge points. My method use the rotation angle calculated earlier by densely tracing a short ridge start from the minutia with length D. Since I have already got the minutia direction at the minutia extraction stage, obviously my method reduces the redundant calculation but still holds the accuracy. My approach is to transform each according to its own reference minutia and then do match in a unified x-y coordinate. Therefore, less computation workload is achieved through my method.

3.3.4 Match Stage

For the aligned minutia patterns there is a need of matching algorithm that should 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 quantizations of minutia.

My approach to elastically 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 minutia 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 100*ratio and ranges from 0 to 100. If the score is larger than a pre-specified threshold, the two fingerprints are from the same finger. However, the elastic match algorithm has large computation complexity and is vulnerable to spurious minutia.

CHAPTER 4

EXPERIMENTAL RESULTS AND CONCLUSION

EXPERIMENTATION RESULTS

4.1 Evaluation indexes for fingerprint recognition

The performance of a fingerprint recognition is determined by the two indexes: one is FRR (false rejection rate) and the other is FAR (false acceptance rate). For an image database, each sample is matched against the remaining samples of the same finger to compute the False Rejection Rate. If the matching *g* against *h* is performed, the symmetric one (i.e., *h* against *g*) is not executed to avoid correlation. All the scores for such matches are composed into a series of Correct Score. Also the first sample of each finger in the database is matched against the first sample of the remaining fingers to compute the False Acceptance Rate. If the matching *g* against *h* is performed, the symmetric one (i.e., *h* against *g*) is not executed to avoid correlation. All the scores from such matches are composed into a series of Incorrect Score.

4.2 Experimentation Results

A set of given fingerprints is used to test the experiment performance. My program tests all the images without any fine-tuning for the database. The experiments show my program can differentiate imposturous minutia pairs from genuine minutia pairs in a certain confidence level. Furthermore, good experiment designs can surely improve the accuracy. Further studies on good designs of training and testing are expected to improve the result.

Here is the diagram for Correct Score and Incorrect Score distribution:

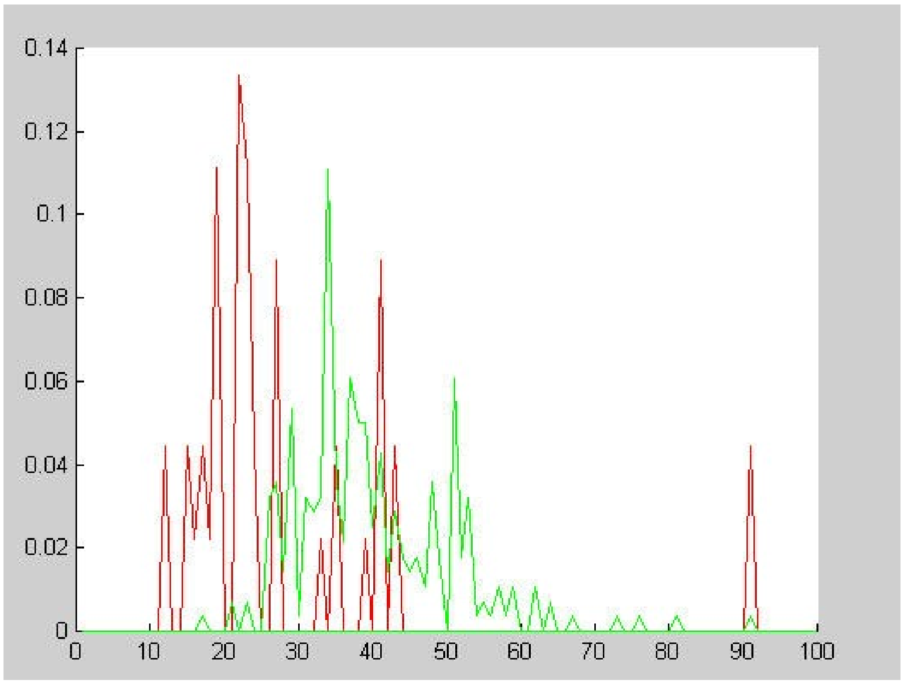
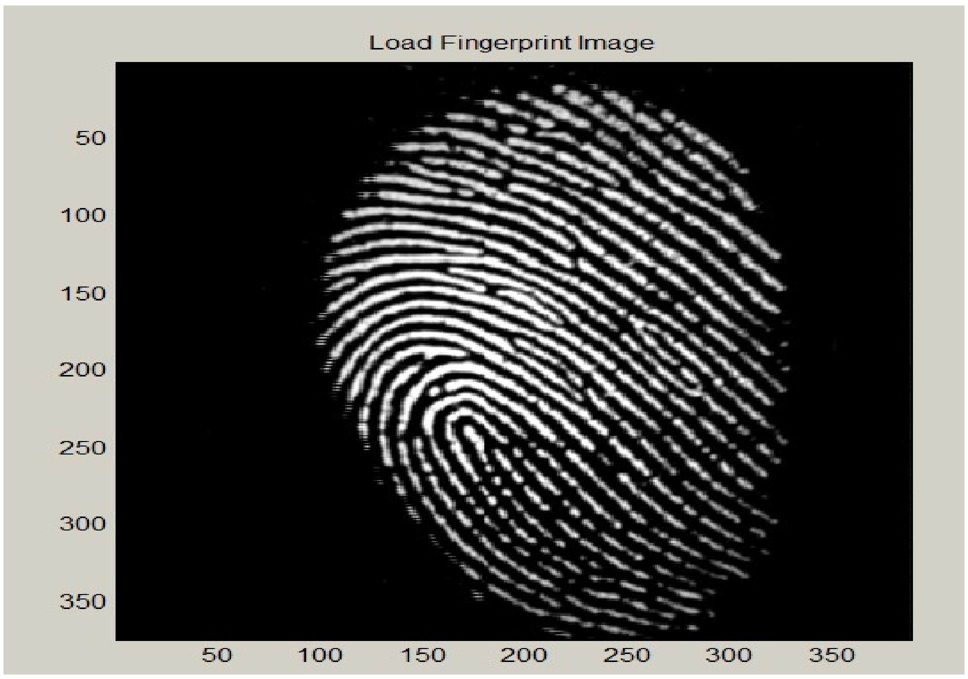


Figure 4.2.1 Distribution of Correct Scores and Incorrect Scores
Red line: Incorrect Score
Green line: Correct Scores

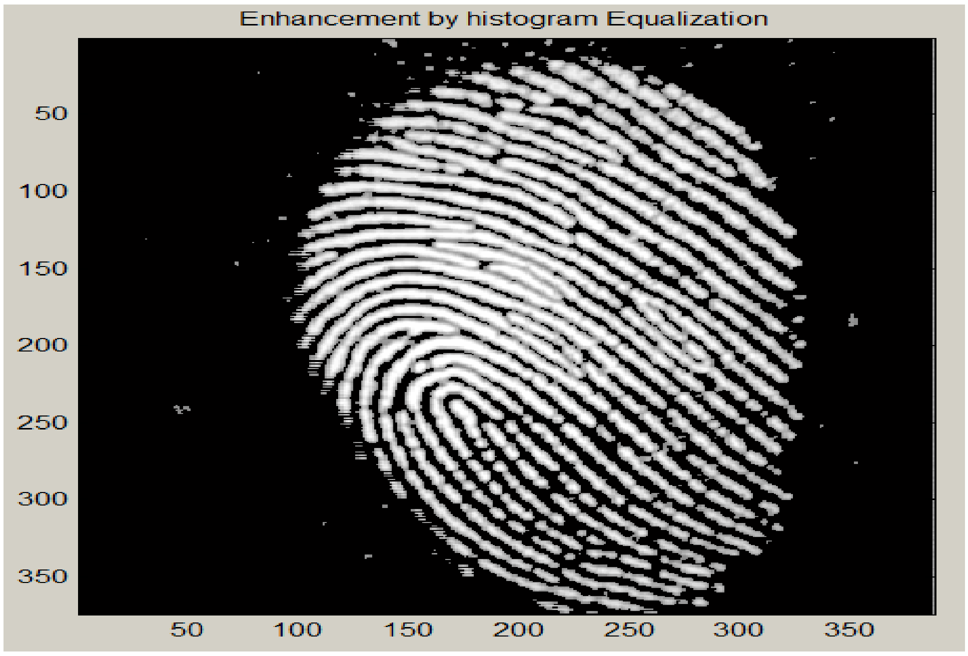
Two partially overlapped distributions are exist that can be shown in above figure . The Red curve whose peaks are mainly located at the left part means the average incorrect match score is 25. The green curve whose peaks are mainly located on the right side of red curve means the average correct match score is 35. This indicates the algorithm is capable of differentiate fingerprints at a good correct rate by setting an appropriate threshold value.

The proposed algorithm has been tested on computer simulated using MATLAB 7.0. The approach can be explained in the following steps:

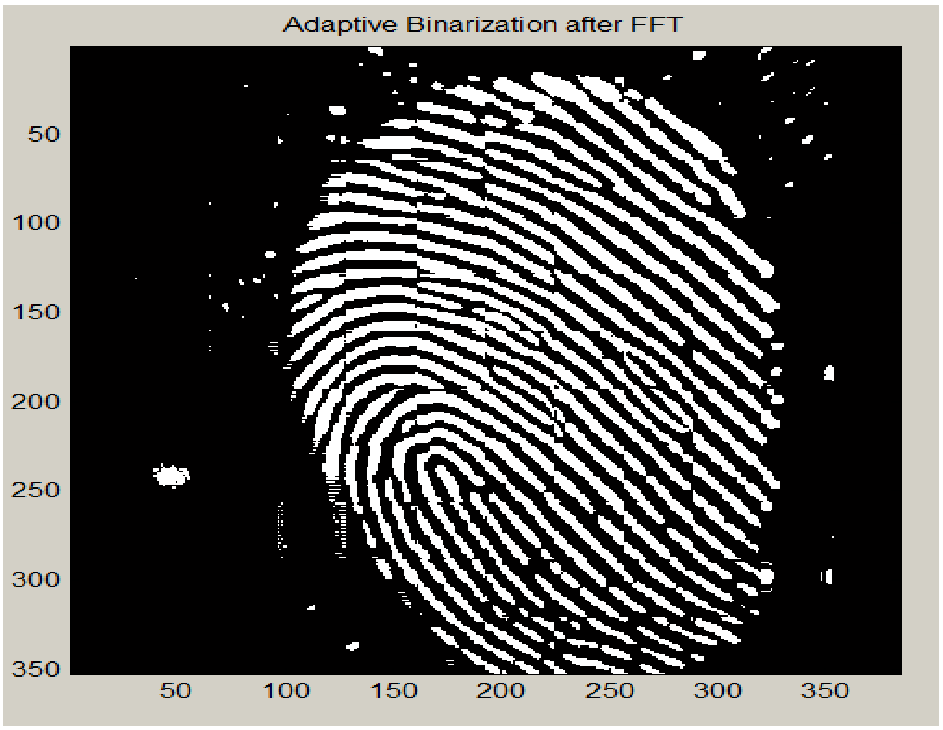
Step 1 : Loading the fingerprint image.



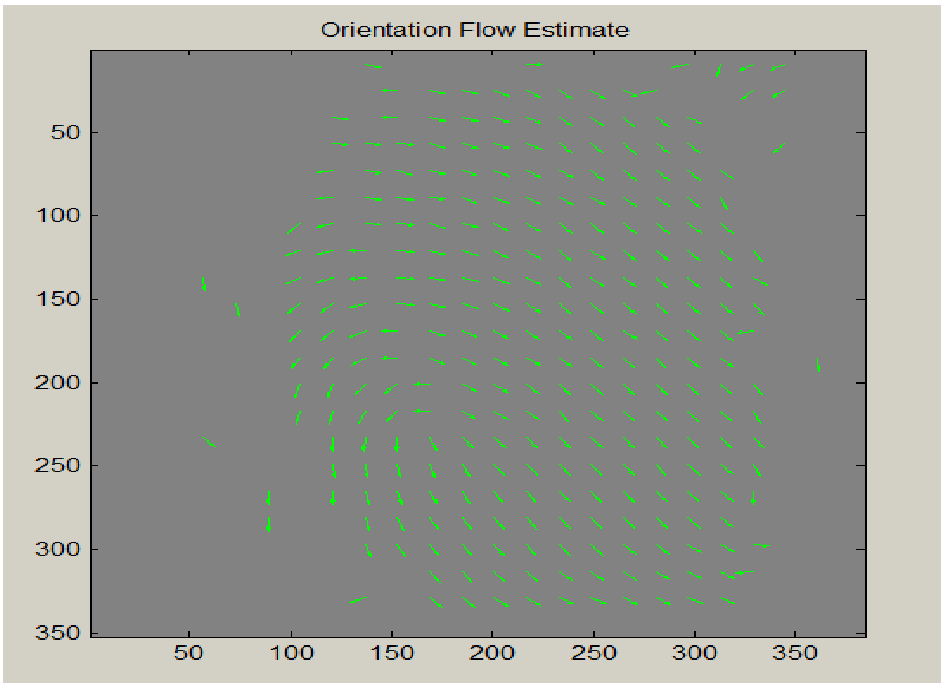
Step 2: Enhancement by histogram Equalization



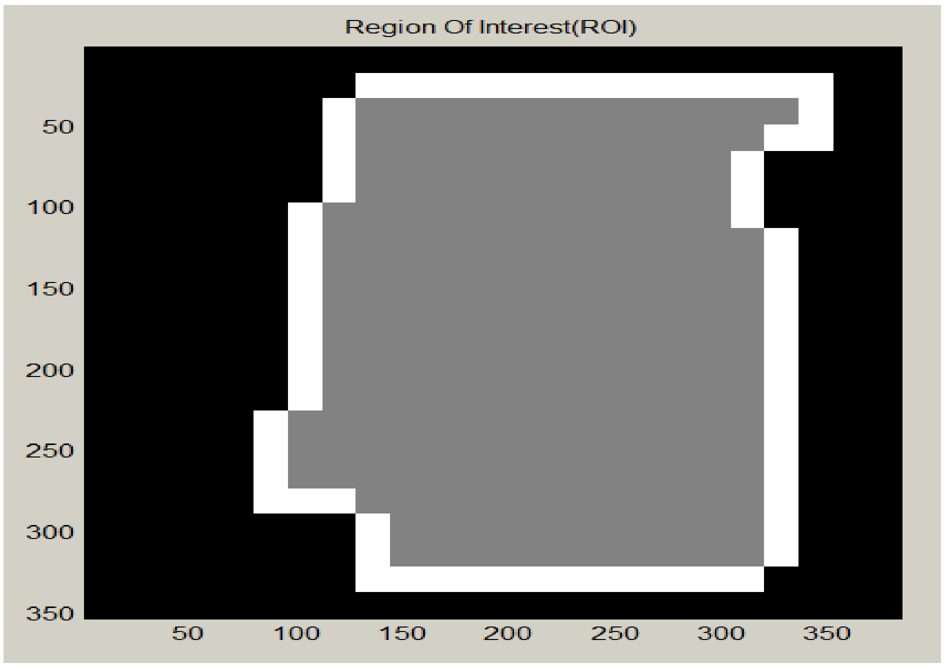
Step 3: Adaptive binarization after FFT



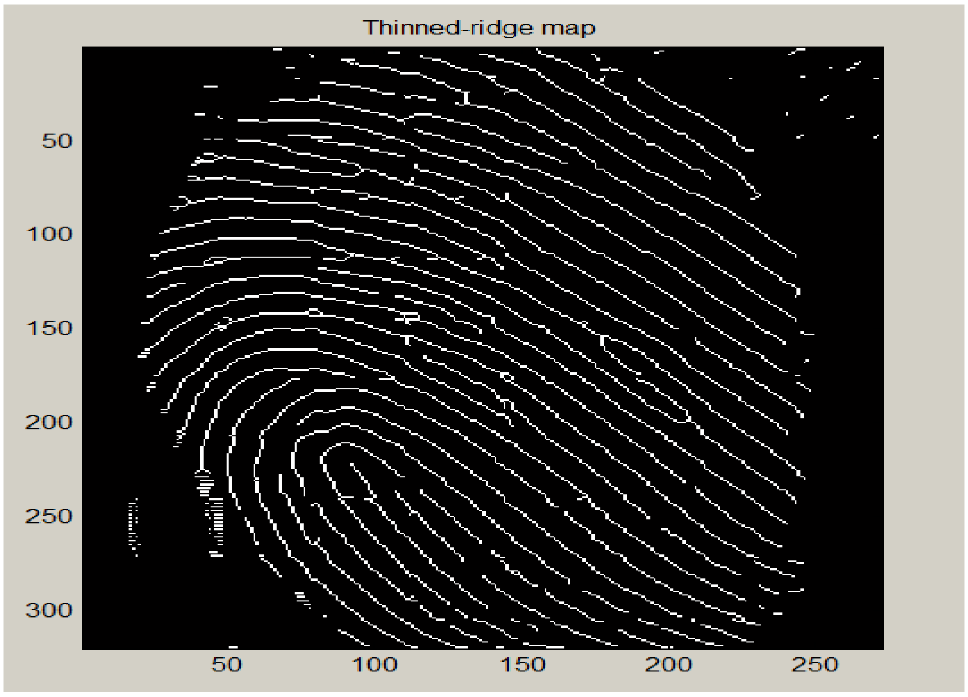
Step 4: Step to find orientation.



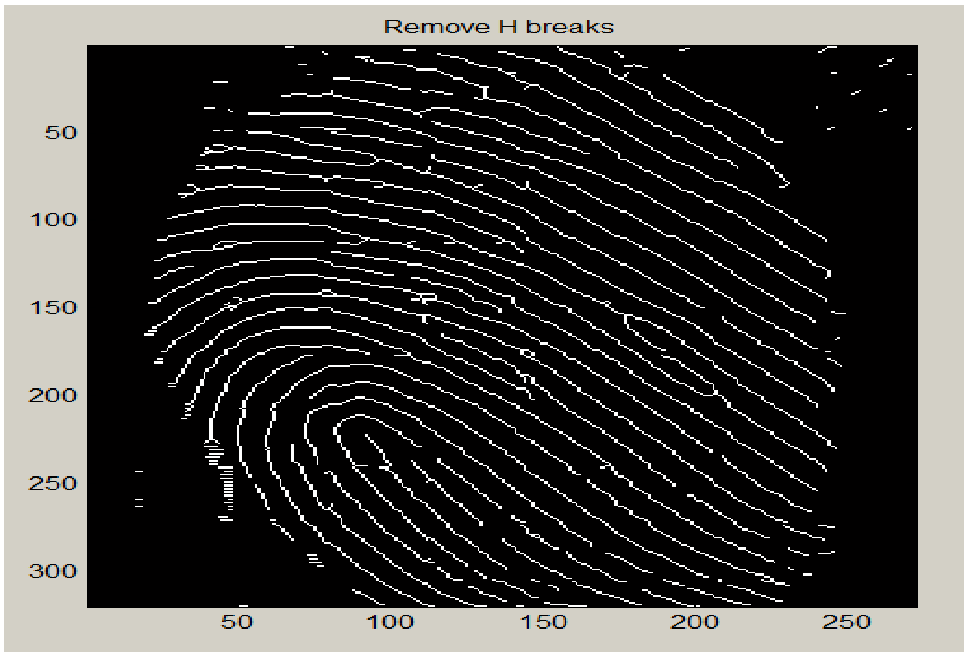
Step 5: find Region of Interest(ROI)



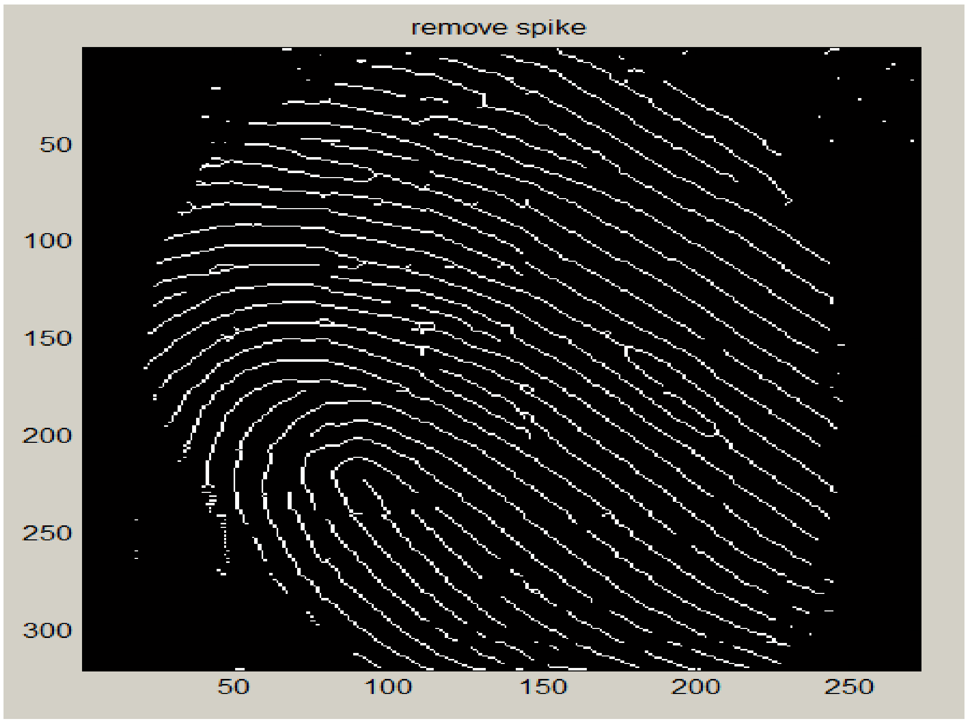
Step 6: for thinning the fingerprint image



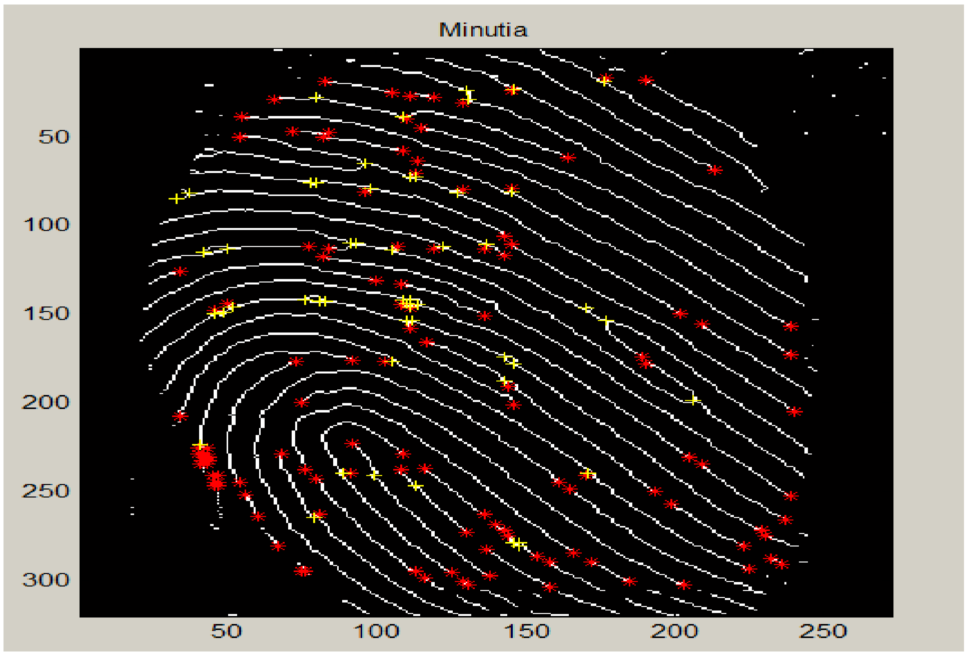
Step 7: for removing H breaks



Step 8: for removing spikes



Step 9: for extracting minutiae.



4.3 CONCLUSTION

My project has combined many methods to build a minutia extractor and a minutia matcher. The combination of multiple methods comes from a wide investigation into research paper. Also some novel changes like segmentation using Morphological operations, minutia marking with special considering the triple branch counting, minutia unification by decomposing a branch into three terminations, and matching in the unified x-y coordinate system after a two-step transformation are used in my project, which are not reported in other literatures I referred to.

CHAPTER 5

REFERENCES
&
APPENDIX

References:

1. G.Sambasiva Rao, C. NagaRaju, L. S. S. Reddy and E. V. Prasad, "*A Novel Fingerprints Identification System Based on the Edge Detection",* International Journal of Computer Science and Network Security, vol. 8, pp. 394-397, (2008).

2. Robert Hastings, "*Ridge Enhancement in Fingerprint Images Using Oriented Diffusion",* IEEE Computer Society on Digital Image Computing Techniques and Applications, pp. 245-252, (2007).

3. Jinwei Gu, Jie Zhou, and Chunyu Yang, "*Fingerprint Recognition by Combining Global Structure and Local Cues",* IEEE Transactions on Image Processing, vol. 15, no. 7, pp. 1952 – 1964, (2006).

4. V.Vijaya Kumari and N. Suriyanarayanan, "*Performance Measure of Local Operators in Fingerprint Detection"*, Academic Open Internet Journal, vol. 23, pp. 1-7, (2008).

5. Raju Sonavane and B. S. Sawant, "*Noisy Fingerprint Image Enhancement Technique for Image Analysis: A Structure Similarity Measure Approach",* Journal of Computer Science and Network Security, vol. 7 no. 9, pp. 225-230, (2007).

6. Eric P. Kukula, Christine R. Blomeke, Shimon K. Modi, and Tephen J. Elliott, "*Effect of Human Interaction on Fingerprint Matching Performance, Image Quality, and Minutiae Count",* International Conference on Information Technology and Applications, pp. 771-776, (2008).

7. M. R. Girgisa, A. A. Sewisyb and R. F. Mansourc, "*Employing Generic Algorithms for Precise Fingerprint Matching Based on Line Extraction",* Graphics, Vision and Image Procession Journal, vol. 7, pp. 51-59, (2007).

8. Duoqian Miao, Qingshi Tang, and Wenjie Fu, "*Fingerprint Minutiae Extraction Based on Principal Cures",* the Journal of the Pattern Recognition Letters, vol. 28, pp. 2184-2189, (2007).

9. Liu Wei, "Fingerprint *Classification using Singularities Detection*", International Journal of Mathematics and Computers in Simulation, issue 2, vol. 2, pp. 158-162, (2008).

10. Hartwing Fronthaler, Klaus kollreider, and Josef Bigun, *"Local Features for Enhancement and Minutiae Extraction in Fingerprints*", IEEE Transactions on Image Processing, vol. 17, no, 3, pp. 354- 363, (2008).

11. Luping Ji, Zhang Yi, "*Fingerprint Orientation field Estimation using Ridge Protection",* The Journal of the Pattern Recognition, vol. 41, pp. 1491-1503, (2008)

12. Manvjeet Kaur, Mukhwinder Singh, Akshay Girdhar, and Parvinder S. Sandhu, "*Fingerprint Verification System using Minutiae Extraction Technique",* Proceedings of World Academy of Science, Engineering and Technology vol. 36, pp. 497-502, (2008).

13. Haiping Lu, Xudong Jiang and Wei-Yun Yau, "*Effective and Efficient Fingerprint Image Post processing",* International Conference on Control, Automation, Robotics and Vision, vol. 2, pp. 985- 989, (2002)

14.Prabhakar, S, Jain, A.K, Jianguo Wang, Pankanti S, Bolle, "*Minutia Verification and Classification for Fingerprint Matching",* International Conference on Pattern Recognition vol. 1, pp. 25-29, (2002).

15. Lin Hong. "Automatic Personal Identification Using Fingerprints", Ph.D. Thesis, 1998.

16. D.Maio and D. Maltoni. Direct gray-scale minutiae detection in fingerprints. IEEE Trans. Pattern Anal. And Machine Intell., 19(1):27-40, 1997.
17. Jain. A.K., Hong, L., and Bolle, R.(1997), "On-Line Fingerprint Verification," IEEE Trans. On Pattern Anal and Machine Intell, 19(4), pp. 302-314.

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

19. L. Hong, A.K. Jain, S. Pankanti, and R. Bolle, “Fingerprint Enhancement,”*Proc. First IEEE WACV*, pp. 202-207, Sarasota, Fla.,1996.

20. Lee, C.J., and Wang, S.D.: Fingerprint feature extration using Gabor filters, Electron. Lett., 1999, 35, (4), pp.288-290.

21. M. Tico, P.Kuosmanen and J.Saarinen. Wavelet domain features for fingerprint recognition, Electroni. Lett., 2001, 37, (1), pp.21-22.

22. L. Hong, Y. Wan and A.K. Jain, "Fingerprint Image Enhancement: Algorithms and Performance Evaluation", IEEE Transactions on PAMI ,Vol. 20, No. 8, pp.777-789, August 1998.

23. Image Systems Engineering Program, Stanford University. Student project By Thomas Yeo, Wee Peng Tay, Ying Yu Tai.

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

25. M. J. Donahue and S. I. Rokhlin, "On the Use of Level Curves in Image Analysis," Image Understanding, VOL. 57, pp 652 - 655, 1992.

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

Appendix A

Matlab Scripts

Following Scripts are used in this work.

1. Script reading the images.

```
function b = readimage(w);
% read the image w
a=imread(w);
b = double(a);
```

2. Script for laoding the image.

```
function image1=loadimage
% dialog for opening fingerprint files
[imagefile1 , pathname]= uigetfile('*.bmp;*.BMP;*.tif;*.TIF;*.jpg','Open An
Fingerprint image');
if imagefile1 ~= 0
    cd(pathname);
    image1=imread(char(imagefile1));
    image1=255-double(image1);
end;
```

3. Script for FFT of image for 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;

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

%d=max(out(:)); %Find max pixel value in C.
%c=min(out(:));
%figure
%imshow(out,[c d]);
 final=histeq(uint8(final));
```

4. Script for segmentation.

```
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;

%white is ridge -> large

if i+W-1 <= w & j+W-1 <= h
   mean_thres = mean2(a(i:i+W-1,j:j+W-1));
   %threshold value is choosed
     mean_thres = 0.8*mean_thres;
     %before binarization
     %ridges are black, small intensity value -> 1 (white ridge)

     %the background and valleys are white, large intensity value -> 0(black)
     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;
```

5. Script for finding Region of Interest(ROI).

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

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

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

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

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

```
%cut out the ROI region image
 %show background,bound,innerArea with different gray intensity:0,100,200

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

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

in=in.*tmp;

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

%inner area
roiArea = im2double(roiArea) - im2double(roiBound);
 if nargin == 3
    colormap(gray);
   imagesc(tmplate);
end;
```

6. Script to show minutiae.

```
function show_minutia(image,end_list,branch_list);
 %show the image of all points in the list
```

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



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

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

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

if ~isempty(end_list)

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

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

7. Script for finding origin of minutiae.

```
function [newXY] = MinuOrigin_TransAll(real_end,k)
%  MinuOrigin_all(real_end,k)
%   set the k-th minutia as origin and align its direction to zero(along x)
%  and then accomodate all other minutia points in the fingerprint to the new origin

%  Also see MinuOrigin
%  The difference between MinuOrigin and MinuOrigin_all is that the orientation of
each minutia is also adjusted with the origin minutia


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;
```

8. Script for mark minutiae.

```
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);

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

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

    elseif neiborNum == 3
      %if two neighbors among the three are connected directly
      %there may be three braches are counted in the nearing three cells
      tmp=in(i-1:i+1,j-1:j+1);

      tmp(2,2)=0;
      [abr,bbr]=find(tmp==1);
      t=[abr,bbr];
      if isempty(branch_list)
        branch_list = [branch_list;[i,j]];
      else

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

9. Script to find inner ridgewidth.

```
function edgeDistance = interRidgeWidth(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);
```

10. Script for removing false minutiae.

```
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);
%totalSuspectMin

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(suspectMinList(k,1
),2) ) ==
ridgeOrderMap(minutiaeList(suspectMinList(k,2),1),minutiaeList(suspectMinList(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(suspectMinList(k,1
),2) ) ==
ridgeOrderMap(minutiaeList(suspectMinList(k,2),1),minutiaeList(suspectMinList(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 points

        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;

%final_end
%pathMap
%edgeWidth
```

11. Script for graphical user interface (gui)

```
%start_gui_single_mode.m
clear
FigWin = figure('Position',[50 -50 650 500],...
   'Name','Fingerprint Recognition ',...
   'NumberTitle','on',...
   'Color',[ 0.8 0.8 0.8 ]);

AxesHandle1 = axes('Position',[0.2 0.2 0.4 0.8],...
   'Box','off');
AxesHandle2 = axes('Position',[0.2 0.2 0.2 0.2],...
   'Box','off');

BackColor = get(gcf,'Color');
[ 0.8 0.8 0.7 ]

[ 0.8 0.7 0.6 ]

 FrameBox = uicontrol(FigWin,...
  'Units','normalized', ...
 'Style','frame',...
 'BackgroundColor',[ 0.7 0.7 0.7 ],...
  'ForegroundColor',[ 0.7 0.7 0.7 ],...
  'Position',[0 0 0.15 1]);

%create static text.
Text2 = uicontrol(FigWin,...
   'Style','text',...
   'Units','normalized', ...

   'Position',[0 0.95 1 0.05],...
   'FontSize',15,...
 'BackgroundColor',[ 0.7 0.7 0.6 ],...
   'HorizontalAlignment','right', ...
   'String','Fingerprint Recognition');



%Text2 = uicontrol(FigWin,...
 %  'Style','text',...
 %  'Units','normalized', ...
  % 'Position',[1 1 1 0.05],...
 %  'FontSize',15,...
  % 'BackgroundColor',[ 0.7 0.7 0.6 ],...
  % 'HorizontalAlignment','right', ...
  % 'String','Fingerprint Verification');

w=16;
textLoad='Load Fingerprint Image';
h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[0,320,80,20],...
   'String','Load',...
   'Callback',...
   ['image1=loadimage;'...
    'subplot(AxesHandle1);'...
    'imagesc(image1);'...
    'title(textLoad);'...
     'colormap(gray);']);

text_filterArea='Orientation Flow Estimate';
h=uicontrol(FigWin,...
   'Style','pushbutton',...
    'Position',[0,240,80,20],...
    'String','Direction',...
    'Callback',...
  ['subplot(AxesHandle2);
[o1Bound,o1Area]=direction(image1,16);title(text_filterArea);']);

text_ROI='Region Of Interest(ROI)';
h=uicontrol(FigWin,...
  'Style','pushbutton',...
   'Position',[0,220,80,20],...
   'String','ROI Area',...
   'Callback',...
  ['subplot(AxesHandle2);
[o2,o1Bound,o1Area]=drawROI(image1,o1Bound,o1Area);title(text_ROI);']);

text_eq='Enhancement by histogram Equalization';
h=uicontrol(FigWin,...


  'Style','pushbutton',...
  'Position',[0,300,80,20],...
  'String','his-Equalization',...
  'Callback',...
  ['subplot(AxesHandle2);image1=histeq(uint8(image1));imagesc(image1);title(text_e
q);']);

text21='Adaptive Binarization after FFT';
h=uicontrol(FigWin,...
  'Style','pushbutton',...
  'Position',[0,260,80,20],...
  'String','Binarization',...
  'Callback',...
  [%'W=inputdlg(text);W=str2num(char(W));'...
    'subplot(AxesHandle1);'...
    'image1=adaptiveThres(double(image1),32);title(text21);']);
```

```
% text='Please input the FFT factor(0~1)';
  text_fft='Enhancement by FFT';
  h=uicontrol(FigWin,...
  'Style','pushbutton',...
  'Position',[0,280,80,20],...
  'String','fft',...
  'Callback',...
   ['W=inputdlg(text);W=str2double(char(W));'...
    'subplot(AxesHandle1);image1=fftenhance(image1,W);imagesc(image1);title(text_
fft);']);

text31='Thinned-ridge map';
h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[0,200,80,20],...
   'String','Thining',...
   'Callback',...
   ['subplot(AxesHandle2);o1=im2double(bwmorph(o2,''thin'',Inf));imagesc(o1,
[0,1]);title(text31);']);

text41='Remove H breaks';
h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[0,180,80,20],...
   'String','remove H breaks',...
    'Callback',...
   ['subplot(AxesHandle2);o1=im2double(bwmorph(o1,''clean''));o1=im2double(bwm
orph(o1,''hbreak''));imagesc(o1,[0,1]);title(text41);']);
```

```
textn1='remove spike';
h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[0,160,80,20],...
   'String','Removing spike',...
```

```
  'Callback',...
  ['subplot(AxesHandle2);o1=im2double(bwmorph(o1,''spur''));imagesc(o1,
[0,1]);title(textn1);']);

%% locate minutia and show all those minutia
text51='Minutia';
h=uicontrol(FigWin,...
  'Style','pushbutton',...
   'Position',[0,140,80,20],...
   'String','Extract',...
   'Callback',...
  ['[end_list1,branch_list1,ridgeMap1,edgeWidth]=mark_minutia(o1,o1Bound,o1Are
a,w);'...
   'subplot(AxesHandle2);show_minutia(o1,end_list1,branch_list1);title(text51);']);

%Process for removing spurious minutia
text61='Remove spurious minutia';
h=uicontrol(FigWin,...
  'Style','pushbutton',...
   'Position',[0,120,80,20],...
   'String','Real Minutiae',...
  'Callback',...
  ['[pathMap1,real_end1,real_branch1]=remove_spurious_Minutia(o1,end_list1,branc
h_list1,o1Area,ridgeMap1,edgeWidth);'...
  'subplot(AxesHandle1);show_minutia(o1,real_end1,real_branch1);title(text61);']);
```

```
%save template file, including the minutia position,direction,and ridge information
textSaveName='file name';
h=uicontrol(FigWin,...
  'Style','pushbutton',...
   'Position',[0,100,80,20],...
   'String','save',...
  'Callback',...
  ['W=inputdlg(textSaveName);W=char(W);'...
```

```
    'save(W,''real_end1'',''pathMap1'',''-ASCII'');']);

%invoke template file loader and do matching
h=uicontrol('Style','pushbutton',...
   'String','Match',...
    'Position',[0,80,80,20],...
    'Callback',...
   ['finger1=fingerTemplateRead;'...
    'finger2=fingerTemplateRead;'...
    'percent_match=match_end(finger1,finger2,10);']);
```