FINGERPRINT RECOGNITION

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Electrical Engineering

By

NEETA MURMU

And

ABHA OTTI

Under the Guidance of

Prof. P. K. Sahu

Department of Electrical Engineering

National Institute of Technology

Rourkela-769008

National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "FINGERPRINT RECOGNITION" submitted by Neeta Murmu and Abha Otti in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Electrical Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date**:**

Rourkela

Prof. P. K. Sahu

Dept. of Electrical Engineering,

National Institute of Technology

Rourkela - 769008, Orissa

ACKNOWLEDGEMENT

We would like to articulate our deep gratitude to our project guide Prof. P.K. Sahu for his guidance, advice and constant support in the project work. We would like to thank him for being our advisor here at National Institute of Technology, Rourkela. We would like to thank all faculty members and staff of the Department of Electrical Engineering, N.I.T. Rourkela for their generous help in various ways for this project.

Last but not the least, our sincere thanks to all of our friends who have patiently extended all sorts of help in this project.

Neeta Murmu
Roll No-10502015
Dept. of EE, NIT, Rourkela

Abha Otti
Roll No-10502020
Dept. of EE, NIT, Rourkela

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 special consideration of the triple branch counting and 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 1

Introduction

1.INTRODUCTION

1.1 What is a fingerprint?

Skin on human fingertips contains ridges and valleys which together forms distinctive patterns. These patterns are fully developed under pregnancy and are permanent throughout whole lifetime. Prints of those patterns are called fingerprints. Injuries like cuts, burns and bruises can temporarily damage quality of fingerprints but when fully healed, patterns will be restored.

Through various studies it has been observed that no two persons have the same fingerprints, hence they are unique for every individual .

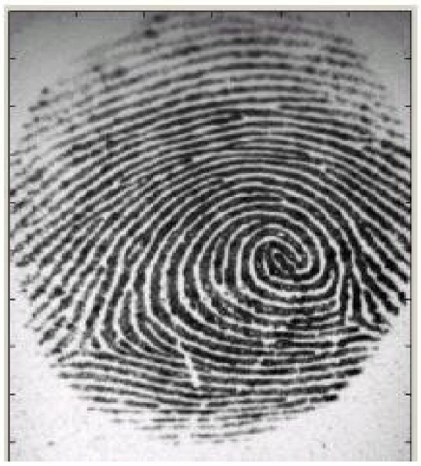


Figure 1. A fingerprint image obtained by optical sensor

Due to the above mentioned properties, fingerprints are very popular as biometrics measurements. Especially in law enforcement where they have been used over a hundred years to help solve crime. Unfortunately fingerprint matching is a complex pattern recognition problem. Manual fingerprint matching is not only time consuming but education and training of experts takes a long time. Therefore since 1960s there have been done a lot of effort on development of automatic fingerprint recognition systems.

Automatization of the fingerprint recognition process turned out to be success in forensic applications. Achievements made in forensic area expanded the usage of the automatic fingerprint recognition into the civilian applications. Fingerprints have remarkable permanency and individuality over the time. The observations showed that the fingerprints offer more secure and reliable person identification than keys, passwords or id-cards can provide. Examples such as mobile phones and computers equipped with fingerprint sensing devices for fingerprint based password protection are being produced to replace ordinary password protection methods. Those are only a fraction of civilian applications where fingerprints can be used.

1.2 Fingerprint recognition

The method that is selected for fingerprint matching was first discovered by Sir Francis Galton. In 1888 he observed that fingerprints are rich in details also called minutiae in form of discontinuities in ridges. He also noticed that position of those minutiae doesn’t change over the time. Therefore minutiae matching are a good way to establish if two fingerprints are from the same person or not.

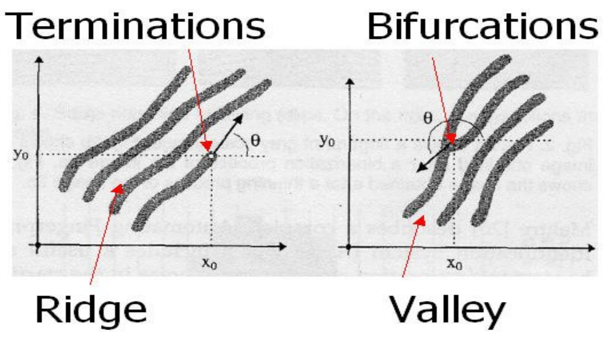


Figure 2 Minutia. (Valley is also referred as Furrow,
Termination is also called Ending,
and Bifurcation is also called Branch)

The two most important minutiae are termination and bifurcation, termination, which is the immediate ending of a ridge; the other is called bifurcation, which is the point on the ridge from which two branches derive.

The fingerprint recognition problem can be grouped into two sub-domains: one is fingerprint verification and the other is fingerprint identification.

Objective

The objective is to implement fingerprint recognition algorithm. The Region of Interest (ROI) for each fingerprint image is extracted after enhancing its quality. The concept of Crossing Number is used to extract the minutia, followed by false minutiae elimination. An alignment based matching algorithm is then used for minutia matching.

CHAPTER 2

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].

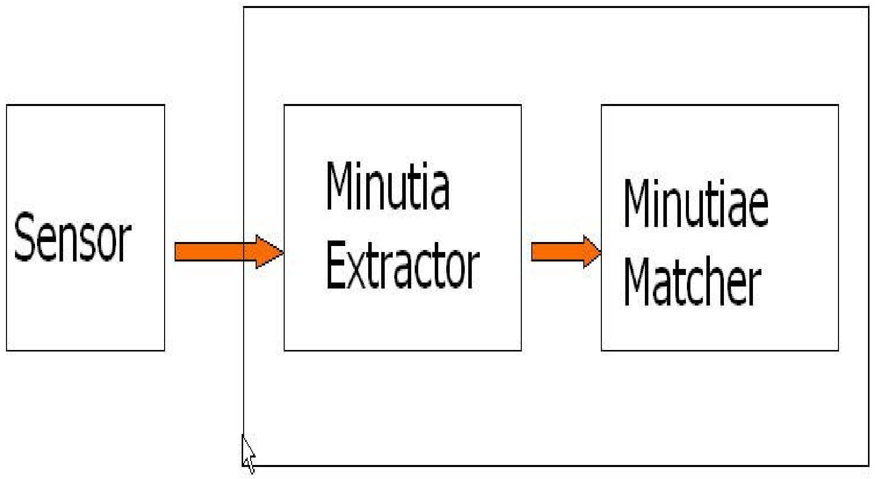


Figure 2.1.1 Simplified Fingerprint Recognition System

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 used in this project is from the available fingerprints provided by FVC2002 (Fingerprint Verification Competition 2002). So no acquisition stage has been implemented.

The minutia extractor and minutia matcher modules have been explained in detail in the next part for algorithm design and other subsequent sections.

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 postprocessing stage [Figure 2.2.1].

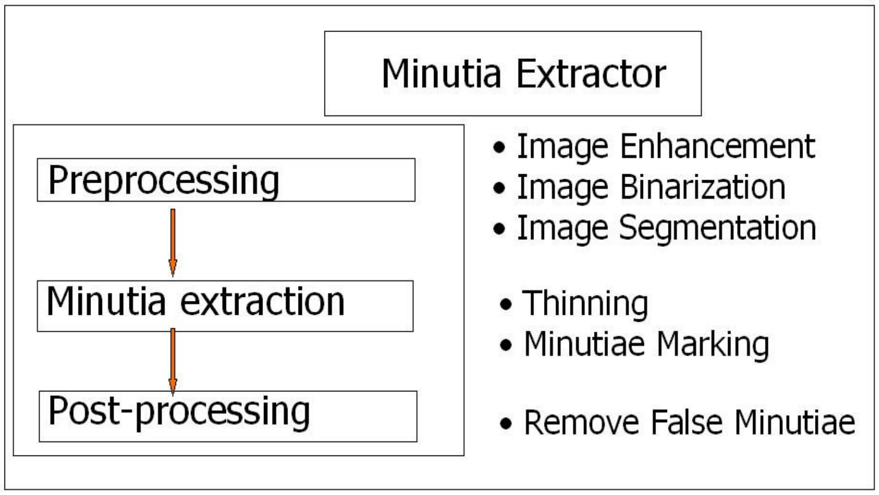


Figure 2.2.1 Minutia Extractor

For the fingerprint image preprocessing stage, Histogram Equalization and Fourier Transform have been 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 postprocessing stage, a more rigorous algorithm is developed to remove false minutia. Also a novel representation for bifurcations is proposed to unify terminations and bifurcations.

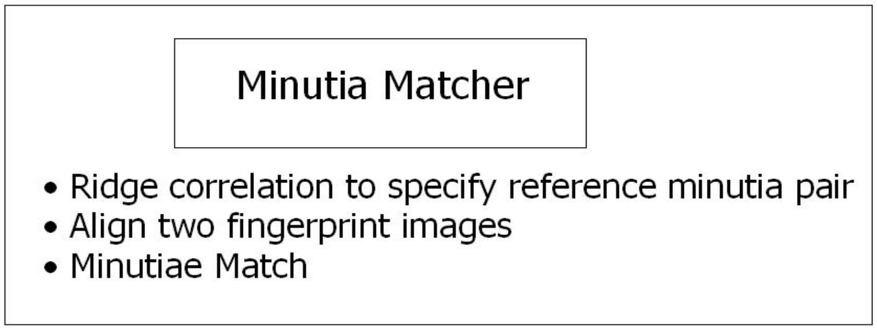

Minutia Matcher
• Ridge correlation to specify reference minutia pair
• Align two fingerprint images
• Minutiae Match

Figure2.2.2 Minutia Matcher

The minutia matcher chooses any two minutia 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 remaining minutia .

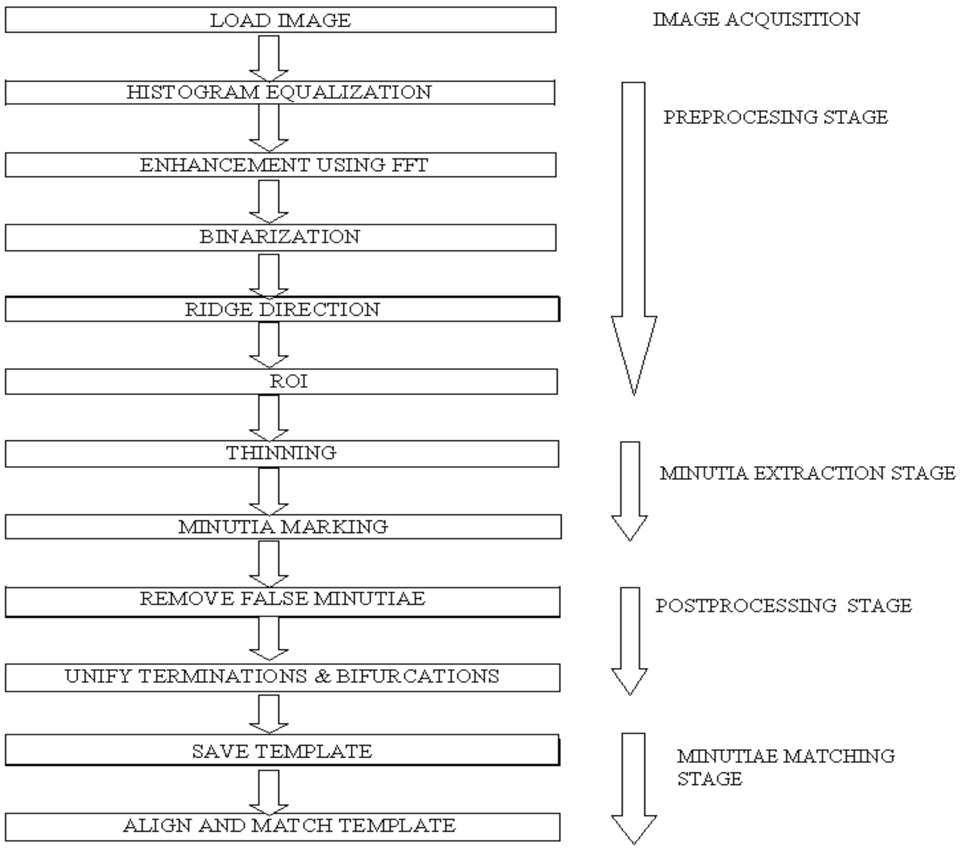


Figure 2.2.3 Steps involved in fingerprint recognition algorithm

CHAPTER 3

Fingerprint

Image Preprocessing

3.1 Fingerprint Image Enhancement

Fingerprint Image enhancement is 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 adopted for image enhancement stage: the first one is Histogram Equalization; the next one is Fourier Transform.

3.1.1 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 has the bimodal type [Figure 3.1.1.1], 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].

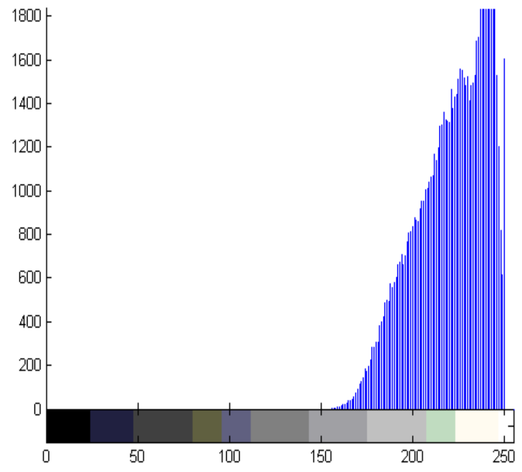


Figure 3.1.1.1 The Original Histogram of a fingerprint image

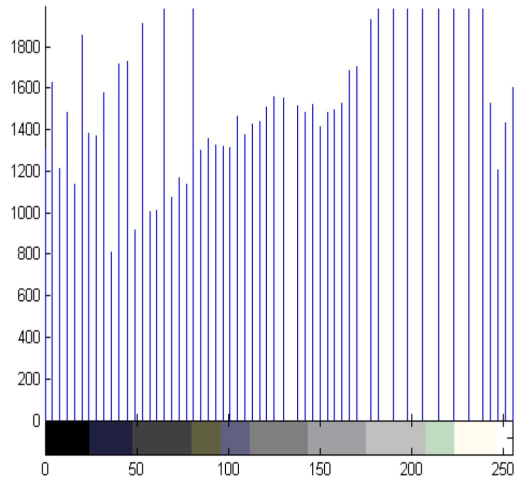


Figure 3.1.1.2 Histogram after histogram equalization

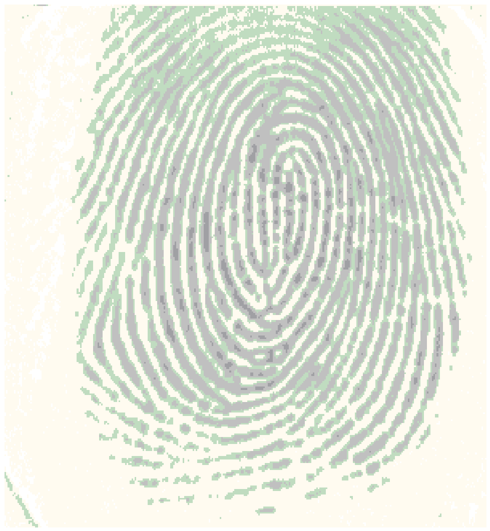


Figure 3.1.1.3. Original Image

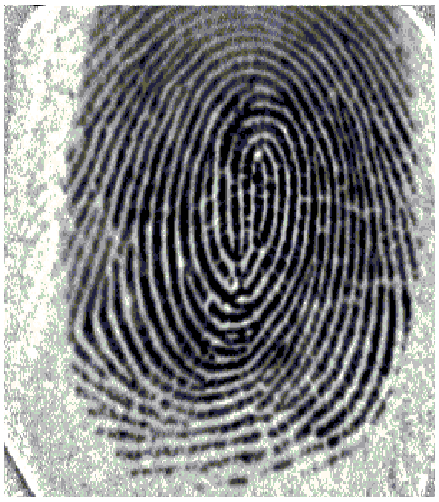


Figure 3.1.1.4 Enhanced Image after Histogram Equalization

3.1.2 Fingerprint Enhancement by Fourier Transform

The image was divided into small processing blocks (32 by 32 pixels) and the Fourier transform was performed according to:

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

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

In order to enhance a specific block by its dominant frequencies, the FFT of the block was multiplied by its magnitude a set of times.

Where the magnitude of the original FFT = abs(F(u,v)) = |F(u,v)|.

The enhanced block is obtained 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 was chosen as k=0.45 to calculate. While having a higher "k" improves the appearance of the ridges, filling up small holes in ridges, having too high a "k" can result in false joining of ridges. Thus a termination might become a bifurcation. Figure 3.1.2.2 represents the image after FFT enhancement.

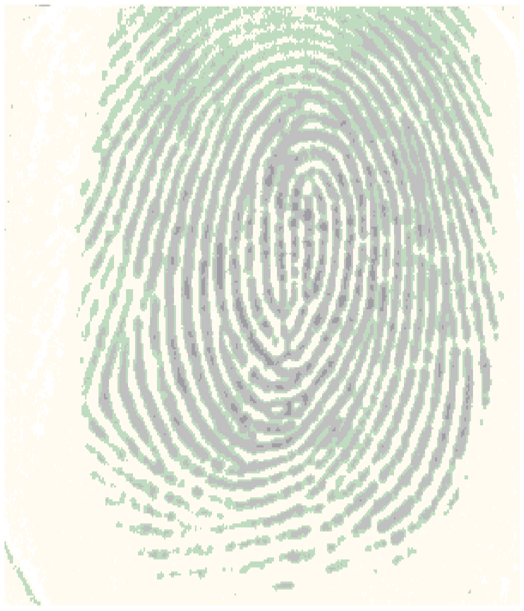


Figure 3.1.2.1 Original image

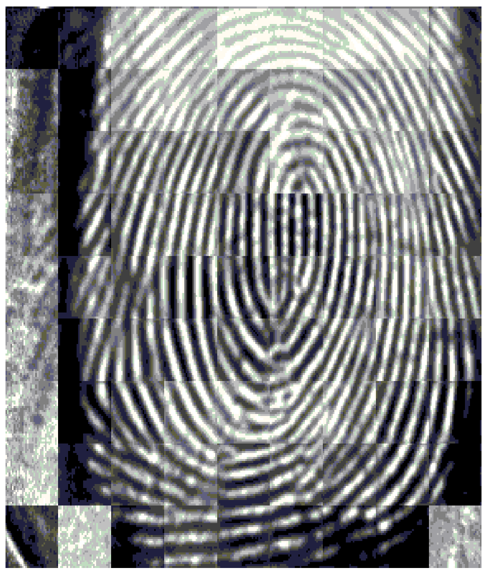


Figure 3.1.2.2. Fingerprint Enhanced By FFT

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

3.2 Fingerprint Image Binarization

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

A locally adaptive binarization method is performed to binarize the fingerprint image. Such a named method comes from the mechanism of transforming a pixel value to 1 if the value is larger than the mean intensity value of the current block (16x16) to which the pixel belongs [Figure 3.2.1and Figure 3.2.2].

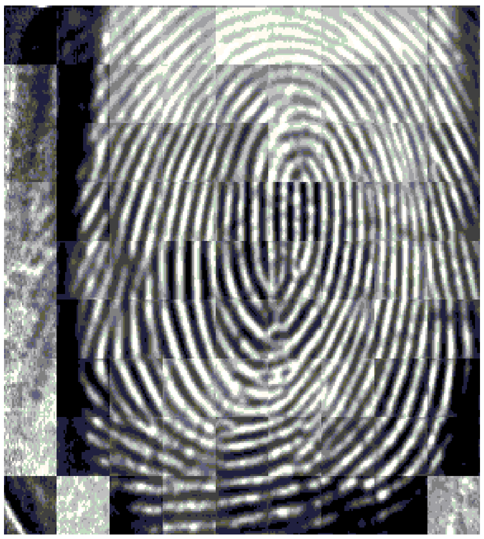


Figure 3.2.1 Enhanced Image

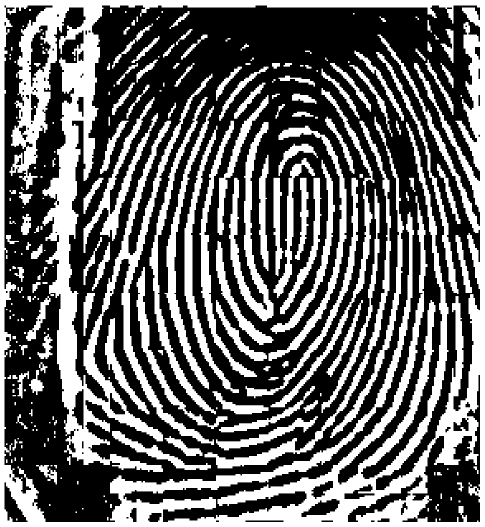


Figure 3.2.2 Image after Adaptive Binarization

3.3 Fingerprint Image Segmentation

In general, only a Region of Interest (ROI) is useful to be recognized for each fingerprint image. The image area without effective ridges and furrows is first discarded since it only holds background information. 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.

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.

3..3.1 Block direction estimation

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

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

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

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

$tg2\theta = 2\sin\theta \cos\theta / (\cos^2\theta - \sin^2\theta)$

After the estimation of each block direction, those blocks without significant information on ridges and furrows 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 (assuming there is only one fingerprint in each image.)

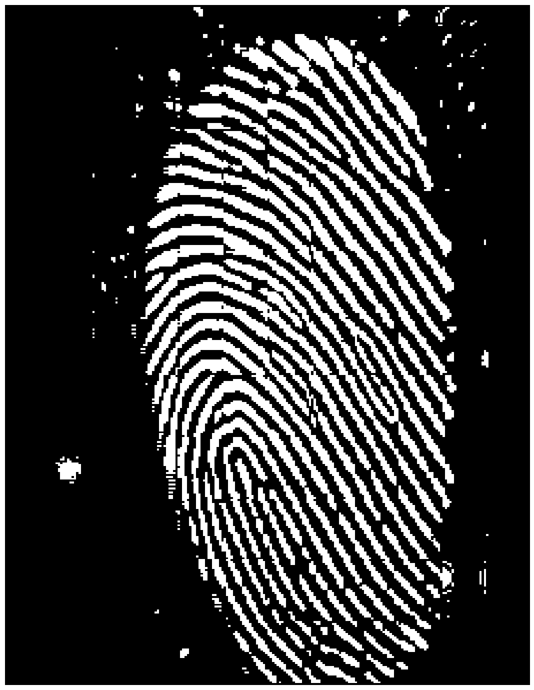


Figure 3.3.1.1.Binarization Image

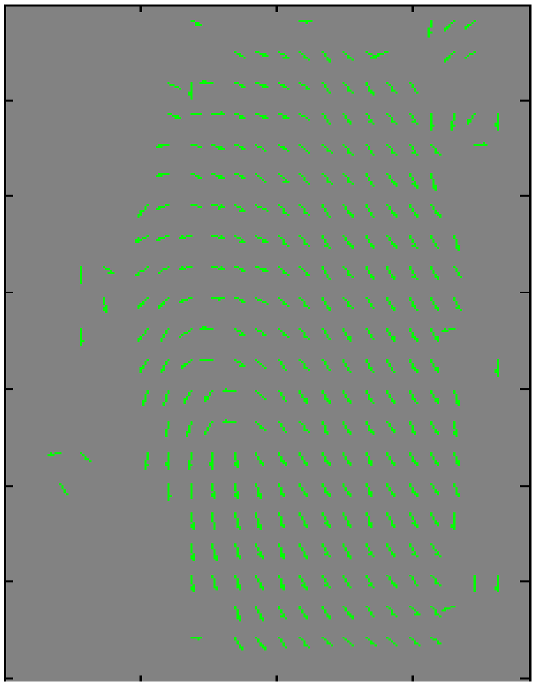


Figure 3.3.1.2 Direction Map

3.3.2 ROI extraction by Morphological operations

Two Morphological operations called ‘OPEN’ and ‘CLOSE’ are adopted. The ‘OPEN’ operation can expand images and remove peaks introduced by background noise [Figure 3.3.2.3]. The ‘CLOSE’ operation can shrink images and eliminate small cavities [Figure 3.3.2.2].

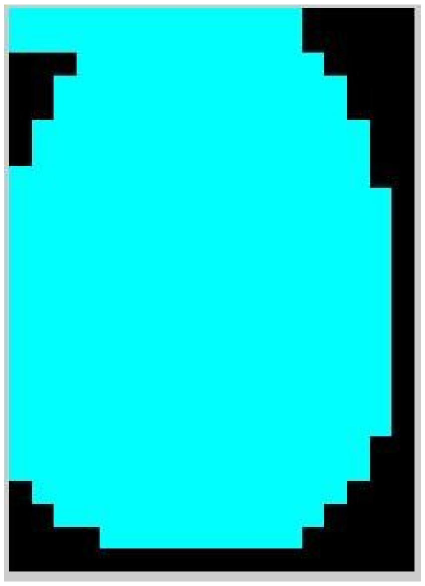


Figure 3.3.2.1 Original Image Area

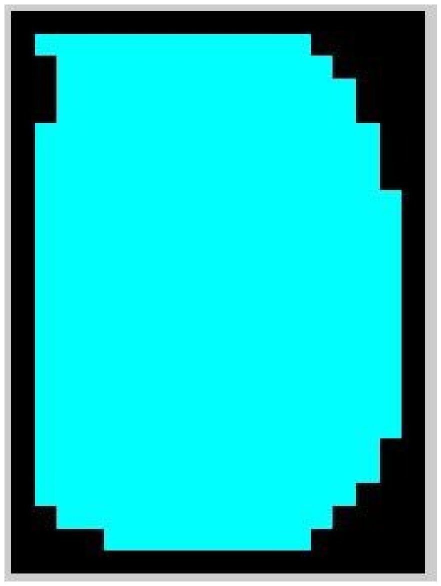


Figure 3.3.2.2 After CLOSE operation

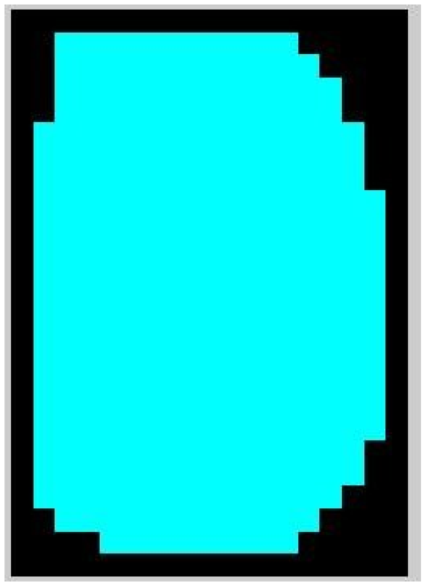


Figure 3.3.2.3 After OPEN operation

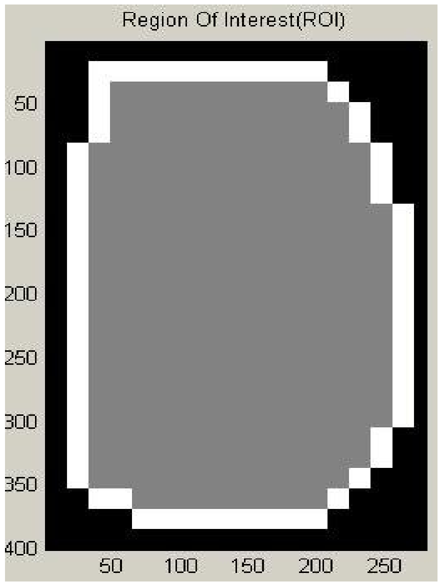


Figure 3.3.2.4 ROI + Bound

Figure 3.3.2.4 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.

CHAPTER 4

Minutia Extraction

4.1 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 three Morphological operations to remove some H breaks, isolated points and spikes.

4.2 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., if Cn(P) = = 1 it's a ridge end and if Cn(P) = = 3 it's a ridge bifurcation point, for a pixel P.

0	1	0
0	1	0
1	0	1

Figure 4.2.1. Bifurcation

0	0	0
0	1	0
0	0	1

Figure 4.2.2. Termination

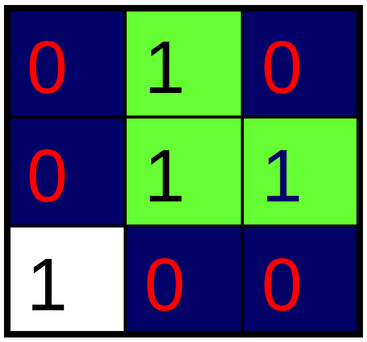


Figure 4.2.3 Triple counting branch

Figure 4.2.3 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 to scan a row of the thinned ridge image and sum up all pixels in the row whose value is one. Then divide the row length with the above summation to get an inter-ridge width. For more accuracy, such kind of row scan is performed upon several other rows and column scans are also conducted, finally all the inter-ridge widths are averaged to get the D.

Together with the minutia marking, all thinned ridges in the fingerprint image are labeled with a unique ID for further operation. The labeling operation is realized by using the Morphological operation: BWLABEL.

CHAPTER 5

Minutia Postprocessing

5.1 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. Actually all the earlier stages themselves occasionally introduce some artifacts which later lead to spurious minutia. These false minutia will significantly affect the accuracy of matching if they are simply regarded as genuine minutia. So some mechanisms of removing false minutia are essential to keep the fingerprint verification system effective.

Seven types of false minutia are specified in following diagrams:

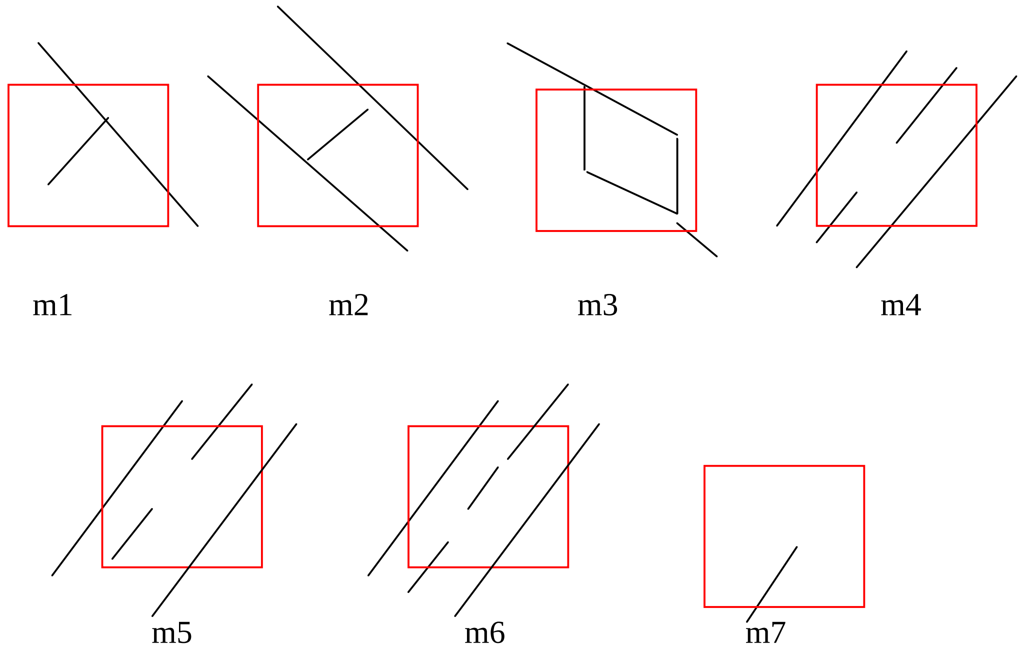


Figure 5.1.1. False Minutia Structures.

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 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), both of them are removed. 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, the two bifurcations are removed. (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 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).

5.2 Unify terminations and bifurcations

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. The bifurcation is broken into three terminations. The three new terminations are the three neighbor pixels of the bifurcation and each of the three ridges connected to the bifurcation before is now associated with a termination respectively [Figure 5.2.1].

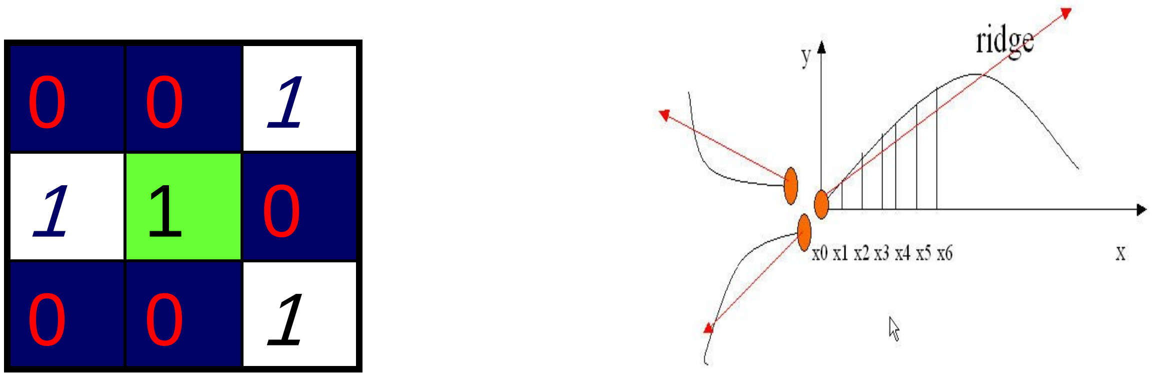


Figure 5.2.1 A bifurcation to three terminations

Three neighbors become terminations (Left)

Each termination has their own orientation (Right)

And the orientation of each termination (tx,ty) is estimated by following method :

A ridge segment is tracked whose starting point is the termination and length is D. All x-coordinates of points in the ridge segment are summed up. The above summation is then divided with D to get sx. And sy can be obtained using the same way.

The direction is obtained from:

atan((sy-ty)/(sx-tx)).

CHAPTER 6

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 is used. 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, any one minutia from each image is chosen, and the similarity of the two ridges associated with the two referenced minutia points is calculated. If the similarity is larger than a threshold, each set of minutia is transformed to a new coordination system whose origin is at the referenced point and whose x-axis is coincident with the direction of the referenced point.
2. Match stage: After obtaining two set of transformed minutia points, the elastic match algorithm is used to count the matched minutia pairs by assuming two minutia having nearly the same position and direction are identical.

6.1 Alignment Stage

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

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

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

where ($x_{i\sim} x_n$) and ($X_{i\sim} X_N$) are the set of minutia for each fingerprint image respectively. And m is minimal one of the n and N value. If the similarity score is larger than 0.8, then the next step is executed else the next pair of rideges are continued to match.

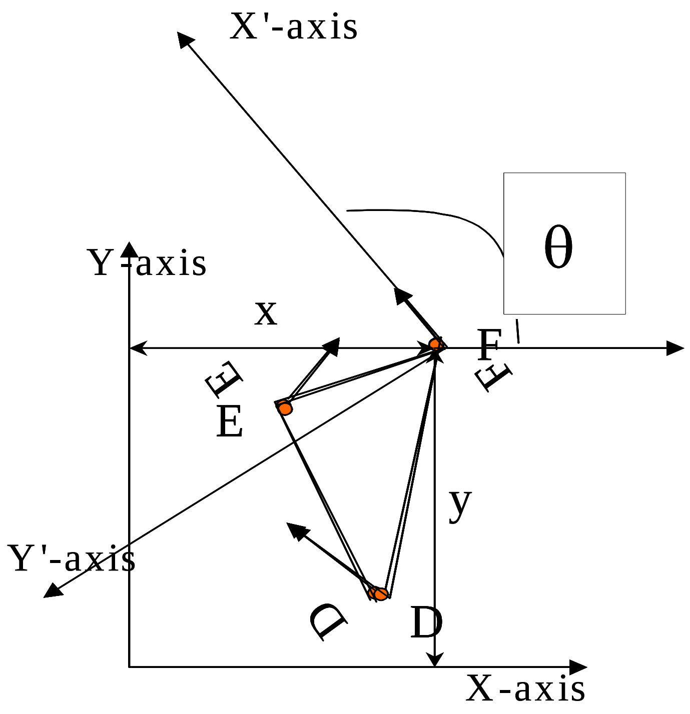
. For each fingerprint, all other minutia are translated and rotated with respect to the reference minutia according to the following formula:

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

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:



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.

6.2 Match Stage

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

The elastic matching of 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 7
Experimentation

7.1 SOURCE CODE

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) == 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;
```

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(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 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;
```

CHAPTER 8
Conclusion

8.1 Evalution indexes for fingerprint recognition

Two indexes are well accepted to determine the performance of a fingerprint recognition system: one is FRR (false rejection rate) and the other is FAR (false acceptance rate).

FAR-describes the number of times, someone is inaccurately positively matched.FRR- derives the number of times,someone who should be identified positively is instead rejected.

- FAR
  (%) FAR=(FA/N)*100
  FA= number of incidents of false acceptance
  N=total number of samples
- FRR
  (%) FRR=(FR/N)*100
  FR=number of incidents of false rejections

8.2 Experimentation results

A fingerprint database from the FVC2000 (Fingerprint Verification Competition 2000) is used to test the experiment performance. The algorithm is capable of differentiating fingerprints at a good correct rate by setting an appropriate threshold value.

Threshold Value	False Acceptance Rate	False Reject Rate
7	0.07%	7.1%
8	0.02%	9.4%
9	0.01%	12.5%
10	0	14.3%

The incorrect acceptance and false rejection are due to some fingerprint images with bad quality and the vulnerable minutia match algorithm.

8.3 Conclusion

The reliability of any automatic fingerprint system strongly relies on the precision obtained in the minutia extraction process. A number of factors are detrimental to the correct location of minutia. Among them, poor image quality is the most serious one. In this project, we have combined many methods to build a minutia extractor and a minutia matcher. The following concepts have been used- segmentation using Morphological operations,minutia marking by specially 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 2-step transformation in order to increase the precision of the minutia localization process and elimination of spurious minutia with higher accuracy. 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 which could improve the future stages and the final outcome.

REFERENCES:

1. A. Jain, R. Bolle, and S. Pankanti, “Biometrics Personal Identification in Networked Society”, Kluwer Academic Publishers New York, Boston, Dordrecht, London, Moscow, pp. 1-64, 2002.
2. A. K. Jain, F. Patrick, A. Arun, “Handbook of Biometrics”, Springer Science+Business Media, LLC, 1st edition, pp. 1-42, 2008.
3. D. Maio, and D. Maltoni, “Direct gray-scale minutiae detection in fingerprints”, IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 19(1), pp. 27-40, 1997.
4. D. Maltoni, D. Maio, and A. Jain, S. Prabhakar, “4.3: Minutiae-based Methods’ (extract) from Handbook of Fingerprint Recognition”, Springer, New York, pp. 141-144, 2003.
5. E. Hastings, “A Survey of Thinning Methodologies”, Pattern analysis and Machine Intelligence, IEEE Transactions, vol. 4, Issue 9, pp. 869-885, 1992.
6. L. Hong, "Automatic Personal Identification Using Fingerprints", Ph.D. Thesis, 1998.
7. L. Lam, S. W. Lee, and C. Y. Suen, “Thinning Methodologies-A Comprehensive Survey”, IEEE Transactions on Pattern analysis and machine intelligence, vol. 14, no. 9, 1992.
8. L.C. Jain, U. Halici, I. Hayashi, S.B. Lee, and S. Tsutsui, “Intelligent biometric techniques in fingerprint and face recognition”, The CRC Press, 1999.
9. K. Nallaperumall, A. L. Fred, and S. Padmapriya, “A Novel Technique for Fingerprint Feature Extraction Using Fixed Size Templates”, IEEE 2005 Conference, pp. 371-374, 2005.
10. P. Komarinski, P. T. Higgins, and K. M. Higgins, K. Fox Lisa, “Automated Fingerprint Identification Systems (AFIS)”, Elsevier Academic Press, pp. 1-118, 2005.