Linnæus University
Sweden

Master Thesis

Fingerprint image enhancement and minutiae extraction algorithm

Author: Letian Cao and Yazhou Wang
Supervisor: Sven Nordebo
Examiner: Sven Nordebo

Date: 2016-05-31D
Course Code: 5ED36E
Subject: Electrical Engineering
Level: Advanced

Department Of technology

Abstract

This work aims to study the procedures of fingerprint identification system and to present some efficient algorithms for pre-processing and minutiae extraction. Most pre-processing steps consist of normalization, segmentation and orientation estimation, which focus on decreasing the variance of fingerprints, separating foreground areas and background areas and tracking the direction of ridge lines, respectively. Minutiae extraction is typically divided into two approaches: binarization based method and direct gray scale extraction. However, we put emphasis on binarization based method in this research since it is more commonly used method in research papers. The results of simulation based on a set of fingerprints downloaded from FVC 2006 database showed that algorithms we used are accurate and reliable.

Keywords: Fingerprints; Minutiae; Algorithm; Simulation; Normalization; Segmentation; Orientation; Extraction; Binarization; Thinning

Contents

1 Introduction

With the development of access control system, many applications of biometric identification technology are developed, such as facial recognition, iris recognition, handwriting recognition, fingerprint recognition and speech recognition,etc. Because of the lifetime invariance, uniqueness and convenience of the fingerprint, fingerprint recognition is one of the most well-known biometric identification technologies. And it is by far the most used biometric solution for authentication on computerized systems.

In history, fingerprint image was acquired by using so-called "ink technology" for law enforcement applications. The subject finger should be smeared with black ink and pressed on a paper card, and then the card is scanned by a general scanner to produce the digital image. This acquisition process is called off-line sensing. However, it is not convenient in civil applications or other fields. Nowadays, most individuals and enterprises are preferred to use the automated fingerprint identification system (AFIS) which can sense the finger surface with fingerprint reader directly. No ink is required for capturing fingerprint image, and all things needed to do is to present a finger to an electronic scanner.

The scanner will read the ridge pattern of the fingerprint and send the image to a computer. After pre-processing and matching, a personal identity can be verified.

1.1 Problem discussion

Fingerprint is unique and easy to recognize two fingerprint images that are totally the same or different, while in reality even the same finger still can not leave two totally same scanning image every time. In addition, there are a lot of bad conditions in fingerprint scanning, such as marks on the finger, incomplete fingerprint image, a rotation in fingerprint image and so on. All these problems may cause the fingerprint system unable to correctly identify the users' fingerprints. Hence, there must be some algorithms to improve the fingerprint identification system to reduce error rate.

1.2 Aim and purpose

Basically, the fingerprint identification system always contains pre-processing, minutiae extraction and matching. Each of them has plenty choices on the algorithms that can be applied.

The fingerprint identification methods that will be discussed in this paper are composed of pre-processing and minutiae extraction. The fingerprint information collected by computer contains much noise, and the purpose of pre-processing is to enhance the quality of an image and remove unnecessary noises. Generally, pre-processing involves normalization, image orientation, segmentation, filtering, ridge frequency image, region mask, binarization, thinning and so on.

In this paper we will use normalization and orientation estimation as the fingerprint enhancement and use binarization and thinning as part of the minutiae extraction, considering the method we used for the minutiae extraction. The Figure 1 shows the flow chart of the proposed fingerprint identification system that will be used in this paper.

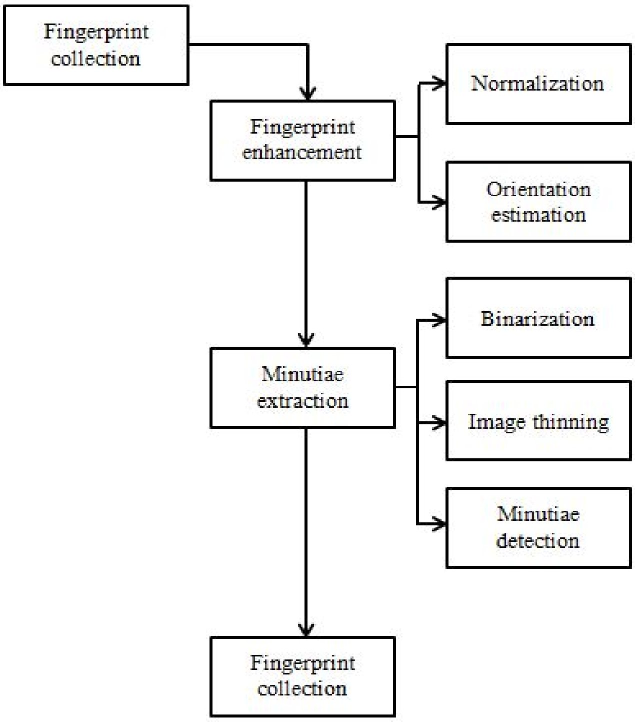


Figure 1: Flow chart of the fingerprint identification system

The fingerprint collection exists different gray scale and background, which can influence the quality of fingerprint images. Hence normalization and segmentation are needed in uniting image variance and background elimination. Orientation estimation is one of the essential parts in fingerprint enhancement to present an unique orientation field. This orientation field can show the visual clues such as local ridge orientation, ridge continuity, ridge tendency, etc. Minutiae extraction is an extremely important part of the fingerprint recognition system. Most of noises can be eliminated by pre-processing procedure, while the pre-processed image still contains many false minutiae. Extracting all the true minutiae and removing all the false minutiae is the main task in this step.

1.3 Limitations

Because of the limited time and resources, the research of the fingerprint system in this paper has some limitations:

1. The fingerprint system in this paper does not contain the matching part. This paper will mainly focus on fingerprint enhancement and minutiae extraction.
2. There are a lot of algorithms for the fingerprint system, but this paper will only choose some of the most common ones among the research papers in this field.
3. The existing algorithm can not be perfect, if the quality of the fingerprint image is very poor, the system can not be very effective.

We wish that all of these limitations can be improved by completing the whole fingerprint system and making more improvement in the future research on this topic.

2 Fingerprint enhancement

The fingerprint image enhancement in this part includes normalization, segmentation and orientation estimation. The goal of each enhancement algorithm is to enhance the image quality of the ridge structures in the recoverable regions and weaken or eliminate the unrecoverable regions. The following sections will introduce and explain three efficient fingerprint enhancement algorithms in details.

2.1 Normalization

The quality of the input fingerprint image largely determines the performance of the pre-processing and minutiae extraction techniques. In an ideal fingerprint image, lines of ridge and valley flow in a constant direction alternately and locally. In this case, the ridges and minutiae points can be easily detected and positioned in the image. However, due to skin condition like wet and cuts, sensor noises and incorrect finger pressure, a large proportion of fingerprint images with poor quality are exported. These reasons that will lead to the problems in minutiae extraction are shown as below:

1. A large number of false minutiae are extracted
2. A significant number of useful minutiae are missed
3. Large erroneous locations of minutiae are detected

Hence, in order to achieve good performance from minutiae extraction algorithms in low quality fingerprint images, the normalization algorithm that can improve the clarity of image structure is needed.

2.1.1 Normalization algorithm

This method of normalization was used by Hong, Wan, and Jain (1998)[1]. It determines the new intensity value for each pixel in an image. Normalization is a pixel-wise operation, which does not change the clarity of ridges and valleys. The main purpose of this method is to reduce the variation of grayscale values along the ridge and valley, which is shown on the subsequent steps:

$$M\left(I\right)=\frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}I\left(i,j\right), \tag{1}$$

$$\mathrm{VAR}\left(I\right)=\frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(I\left(i,j\right)-M\left(I\right)\right)^2, \tag{2}$$

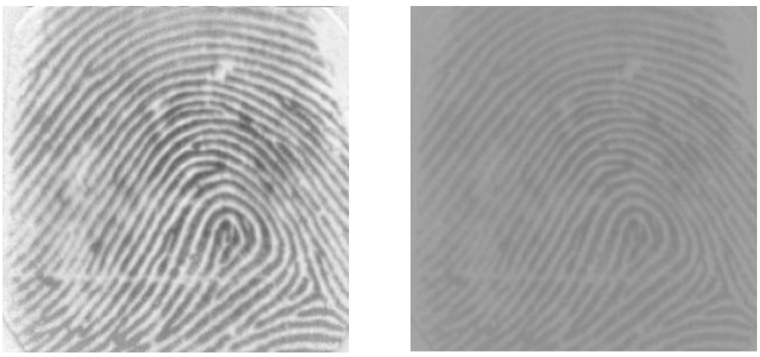
where I(i, j) is the gray-scale of the pixel (i, j), M and VAR are the original fingerprint image mean and variance values. N is the length of the column and row in the fingerprint image. The resolution of the images that used in this paper is 300×300. The normalized gray-scale of the image is defined as:

$$\Gamma\left(i, j\right) = \begin{cases} M_0 + \sqrt{\frac{\mathrm{VAR}_0 (I(i,j)-M)^2}{\mathrm{VAR}}}, & \text{if } I\left(i, j\right) > M, \\ M_0 - \sqrt{\frac{\mathrm{VAR}_0 (I(i,j)-M)^2}{\mathrm{VAR}}}, & \text{otherwise}, \end{cases} \tag{3}$$

where M_0 and VAR_0 are the desired mean and variance, respectively. $\Gamma\left(i, j\right)$ is the normalized gray-scale at pixel (i, j).

2.1.2 Normalization simulation

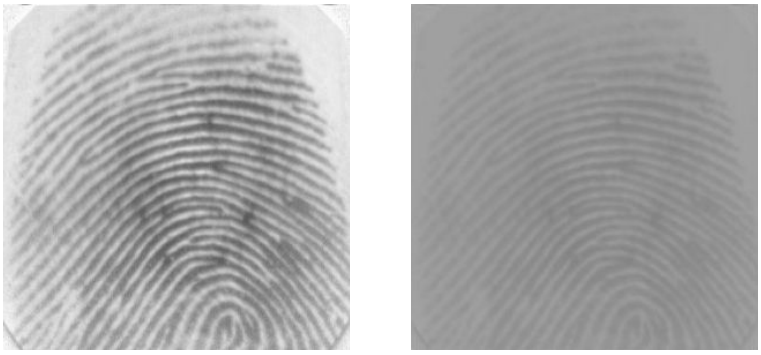
Two different fingerprints are chosen for performance verification. Fingerprint A has more minutiae points and fingerprint B is more smooth relatively. M_0 and VAR_0 are set as 150 and 50. The results of normalization are shown in following figures.



(a) Original fingerprint image (b) Normalized fingerprint image

Figure 2: Normalization of fingerprint A

It can be seen from the Figure 2 and Figure 3 that above method is both suitable for relatively smooth fingerprints and complex fingerprints. After the normalization, the variance of the original image has a significant decrease. This process can help to enhance the ridges on the edge of the fingerprint which have a low gray-scale value in the original fingerprint image and weaken the white line(the noise in Figure 2.a) in the fingerprint at the same time.



(a) Original fingerprint image (b) Normalized fingerprint image

Figure 3: Normalization of fingerprint B

2.2 Segmentation

A captured fingerprint image can generally be divided into two components, foreground (fingerprint area) and background. In order to avoid the extraction of minutiae in background that has noisy areas, the step called segmentation in pre-processing is needed. The segmentation algorithm is to distinguish between background and foreground. The method of segmentation is based on the characteristics of fingerprint images (such as variance and mean) to design algorithms. Mean value gives the contribution of individual pixel intensity for the entire image and variance is normally used to find how each pixel varies from the neighbor pixels (or center pixel) and classify the fingerprint into different regions.

The traditional Segmentation method is to divide image into $w \times w$ blocks and calculate the mean value and variance value for each block. Then compare the variance with threshold which is the boundary intensity of background and foreground directly. However, this comparative approach does not include the consideration of noise influences on mean value and variance value, which may lead to erroneous segmentation. Hence, an adaptive variance approach which considers the noise influence can be implemented as following.

2.2.1 Segmentation algorithm

On the basis of above concept, the fingerprint segmentation algorithm can be generally divided into the following steps:

1. Calculate the mean and variance values for each block:

$$\text{mean}\,(m,n) = \sum_{i=1}^{w}\sum_{j=1}^{w} g\,(i,j)\,, m = 1,2,\cdots,M; n = 1,2,\cdots,N, \tag{4}$$

$$\begin{gathered}\text{var}\,(m,n) = \frac{\sum_{i=1}^{w}\sum_{j=1}^{w}\left[g\,(i,j) - \text{mean}\,(m,n)\right]^2}{w \times w}, \\ m = 1,2,\cdots,M; n = 1,2,\cdots,N.\end{gathered} \tag{5}$$

2. Calculate the average of all block mean and variance value:

$$\begin{aligned} G_{Mean} &= \frac{1}{M \times N}\sum_{i=1}^{N}\sum_{j=1}^{M}\text{mean}\,(m,n)\,, \\ V_{Mean} &= \frac{1}{M \times N}\sum_{i=1}^{N}\sum_{j=1}^{M}\text{var}\,(m,n)\,. \end{aligned} \tag{6}$$

3. Calculate the relative mean and relative variance values of the foreground area:

$$G_{Frg} = \frac{\text{TG}_{Frg}}{\text{NG}_{Frg}}, \qquad V_{Frg} = \frac{\text{TV}_{Frg}}{\text{NV}_{Frg}}, \tag{7}$$

   where TG_{Frg} is the sum of the mean and NG_{Frg} is the number of blocks whose $\text{mean}\,(m,n) > G_{Mean}$. Similarly, TV_{Frg} is the sum of the variance and NV_{Frg} is the number of blocks whose $\text{var}\,(m,n) > V_{Mean}$.

4. Calculate the relative mean and relative variance values of the background area:

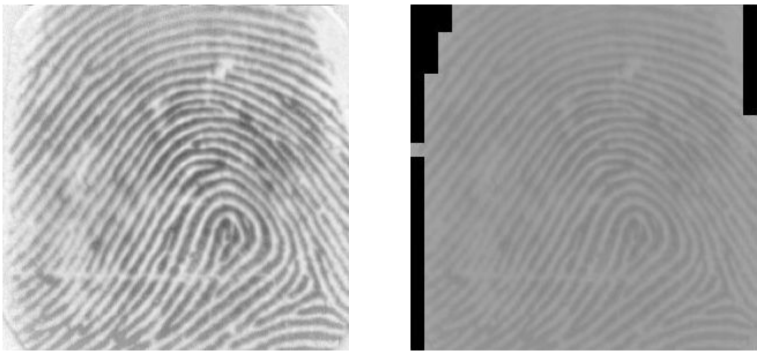
$$G_{Bkg} = \frac{\text{TG}_{Bkg}}{\text{NG}_{Bkg}}, \qquad V_{Bkg} = \frac{\text{TV}_{Bkg}}{\text{NV}_{Bkg}}, \tag{8}$$

   where TG_{Bkg} is the sum of the mean and NG_{Bkg} is the number of blocks whose $\text{mean}\,(m,n) < G_{Mean}$. Similarly, TV_{Bkg} is the sum of the variance and NV_{Bkg} is the number of blocks whose $\text{var}\,(m,n) < V_{Mean}$.

5. G_{Bkg} and V_{Bkg} can be defined as threshold of background and foreground respectively. When a block on condition that $\text{mean}\,(m,n) < G_{Bkg}$ and $\text{var}\,(m,n) < V_{Bkg}$, this block belong to noisy area or background.

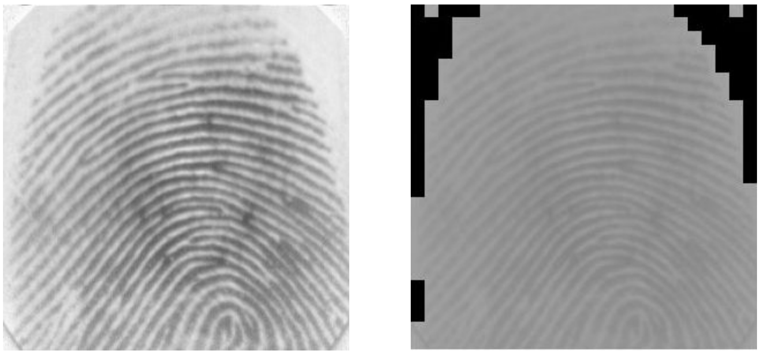
2.2.2 Simulation of segmentation

Two experimental results are shown in following figures. Fingerprint B has larger background area than fingerprint A. And all images are divided into 12×12 blocks.



(a) Original fingerprint image (b) The result of segmentation

Figure 4: Segentation of fingerprint A



(a) Original fingerprint image (b) The result of segmentation

Figure 5: Segentation of fingerprint B

It can be seen from the Figure 4 and Figure 5 that background area has been successfully separated. After simulation, clear ridge lines and structure can be obtained, which is greatly beneficial to subsequent steps, such as thinning and minutiae extraction. This method considers the influence of noise on variance and mean values, which is robust to fingerprints with complex background.

2.3 Orientation estimation

The purpose of orientation estimation is to find the orientation filed of the fingerprint images, while orientation field is an essential property of the fingerprint images. Orientation field defines the angles of the ridges and furrows rotation in a local neighborhood, and it is indispensable for the minutiae extraction which is the further step in the fingerprint identification.

2.3.1 Orientation estimation algorithm

There is a classical method which is called gradient-based method and this method has been adopted by many researchers [1]. The algorithm can be realized by these main steps:

1. Dividing the fingerprint image into blocks of size $w \times w$
2. Computing the gradient vectors of all the pixels in the fingerprint image, by using the formula (9):
$$[G_x(x,y), G_y(x,y)]^T = \left[\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}\right]^T = \begin{bmatrix} \frac{I(x+1,y)-I(x-1,y)}{2} \\ \frac{I(x,y+1)-I(x,y-1)}{2} \end{bmatrix}, \tag{9}$$
where $I(x,y)$is the gray-scale value of the pixel which is at (x,y), and $[G_x(x,y), G_y(x,y)]^T$ is the gradient vectors of the pixel which is at (x,y).
3. Squaring all the point gradient vectors $[G_x(x,y), G_y(x,y)]^T$. Here we assume that every point gradient vectors are complex number, which mean that the gradient vectors can be written as $G_x(x,y) + \mathrm{i}G_y(x,y)$. So the squared gradient vectors can be obtained by using the formula (3):
$$G_{sx}(x,y) = G_x^2(x,y) - G_y^2(x,y), \tag{10}$$
$$G_{sy}(x,y) = 2G_x(x,y)\,G_y(x,y), \tag{11}$$
where $G_{sx}(x,y)$is the real part of the squared gradient vectors, while the $G_{sy}(x,y)$ is the image part of the squared gradient vectors. Then we calculate the mean value of the squared gradient vectors in every blocks:
$$[G_{msx}(x,y), G_{msy}(x,y)]^T = \left[\sum_{x=1}^{W}\sum_{y=1}^{W} G_{sx}(x,y), \sum_{x=1}^{W}\sum_{y=1}^{W} G_{sy}(x,y)\right]^T. \tag{12}$$
4. Computing the θ which is the angle of the ridge-valley orientation, while θ should not be less than 0 and less than π. Since the gradient vectors has been squared, the angles that can be calculated by *arctan* is twice of the gradient vectors' mean value in every block. So we should half the angles,and then add $\frac{\pi}{2}$ on them to rotate the gradient vectors and turn the gradient vectors into the ridge-valley orientations.

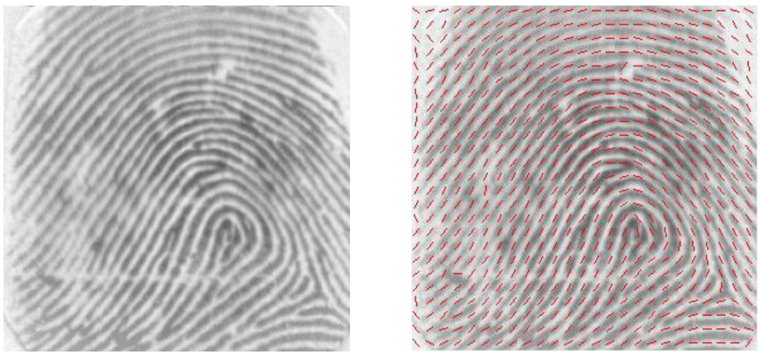
What's more, we should guarantee that θ is still in its value range. Considering all these situation, θ can be obtain by formula (13)

$$\theta = \frac{1}{2}\pi + \frac{1}{2}\begin{cases} \tan^{-1}\left(\frac{G_{msy}}{G_{msx}}\right), & \text{if } G_{msx} \geq 0; \\ \tan^{-1}\left(\frac{G_{msy}}{G_{msx}}\right) + \pi, & \text{if } G_{msx} < 0 \cap G_{msy} \geq 0; \\ \tan^{-1}\left(\frac{G_{msy}}{G_{msx}}\right) - \pi, & \text{if } G_{msx} < 0 \cap G_{msy} < 0. \end{cases} \tag{13}$$

Finally, we get the orientation filed of the fingerprint images with the a matrix full of the angles at each blocks.

2.3.2 Simulation of orientation filed

After compiling the algorithm on the Matlab, a simulation can be done by the algorithm introduced above. After the computation of the angles, draw the orientation vectors on the fingerprint image to check the result.



(a) Original fingerprint image (b) Orientation field image with $w = 12$

Figure 6: The fingerprint after orientation estimation

Form Figure 6, we notice that the orientation filed result well present the ridges and furrows rotation, while it can also be easily disturbed by the noise and corrupted ridge and furrow structures. The reason is that the algorithm is all based on calculating the gray-values changing of the pixels. The fingerprint image with artificial noises is shown on Figure 7.

In order to eliminate these errors in the orientation estimation, we can use a Gaussian low-pass filter to smooth the orientation filed. Before using the Gaussian low-pass filter, a problem has to be solved. The angle θ is a matrix with the elements which are range from 0 to π, and from π to 0 it is not continuous. However, in the image there are two unexpected situations, which are ridges' orientations are close to 0 and

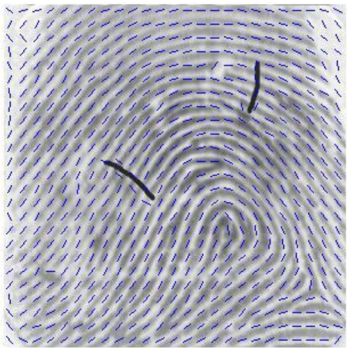


Figure 7: Orientation field of the fingerprint image with artificial noise.

the ridges' orientations are close to π, and the ridges in these two situations in the local neighborhood are continuous. So the θ has to be converted into a continuous vector field, and here one of the method is introduced as follows.

1. Using θ to create two new sets with sine and cosine functions:

$$\Phi_x(i,j) = \cos(2\theta(i,j)), \tag{14}$$

and

$$\Phi_y(i,j) = \sin(2\theta(i,j)), \tag{15}$$

where $\Phi_x(i,j)$ and $\Phi_y(i,j)$ are the two new sets. These two sets describe the x and y components of the continuous vector field.

2. Applying the Gaussian low-pass filter on these two sets:

$$\Phi_x^{'}(i,j) = H(w,\sigma)\,\Phi_x(i,j), \tag{16}$$

and

$$\Phi_y^{'}(i,j) = H(w,\sigma)\,\Phi_y(i,j), \tag{17}$$

where $\Phi_x^{'}(i,j)$ and $\Phi_y^{'}(i,j)$ are the sets after filtering. $H(w,\sigma)$ is the Gaussian low-pass filter.

3. Finding the new θ set:

$$\theta^{'}(i,j) = \frac{1}{2}\tan^{-1}\left(\frac{\Phi_y^{'}(i,j)}{\Phi_x^{'}(i,j)}\right). \tag{18}$$

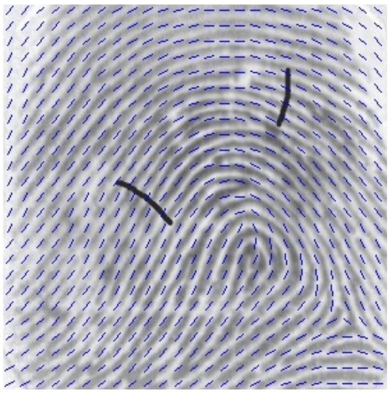


Figure 8: The new orientation field of the fingerprint image with artificial noise.

With this algorithm, the new angle set $\theta^{'}(i,j)$ will present a fairly smooth orientation filed and the result is shown in Figure 8.

After using the Gaussian low-pass filter, the orientation estimation will eliminate the noises in the fingerprint image and get a smooth orientation field.

3 Minutiae extraction

Most fingerprint recognition systems are based on minutiae matching, but a reliable minutiae matching algorithm requires accurate minutiae extraction. Therefore, minutiae extraction is an extremely important step in the whole system, especially in low-quality images where noise may hide the real minutiae. The common minutiae extraction methods are: binarization based method and direct gray scale extraction.

3.1 Direct gray scale extraction

The most used approach of minutiae extraction is based on binarization and thinning. However, some researchers still choose to extract minutiae directly from the gray-scale fingerprints, because they noticed that there are some disadvantages related to the binarization based method, such as minutiae points missing, time consuming and unsatisfactory performance on low quality images, etc.

Maio and Maltoni (1997) [14] proposed one kind of direct gray scale extraction approach which can be called ridge line following method. The basic principle is to set a start point X with given direction θ on a ridge line and move some certain pixels to a new point Y from the start point alone the direction θ by using ridge line following algorithm. Then draw a line through the point Y and the direction is perpendicular to θ. All points on the ridge along this line can be defined as a set Z. A new start point X_1 which has the maximum intensity among the set Z can be chosen. Step by step, until the start point terminate or intersect other ridge lines. Minutiae detection can be achieved by finding all ending points and bifurcation points.

Direct gray scale extraction reduces the procedure of computation, but this technique is more complex than binarization based method. And if the distance between two ridge lines is too large, a new start point may not be found through this algorithm.

3.2 Binarization based method

The binarization based method is to convert a pixel image into a binary image firstly, and then obtain the skeleton of the fingerprint to extract the minutiae through the thinning process. And the direct gray scale extraction is to extract the minutiae directly from the fingerprint image without binarization and thinning. In this work, we choose the binarization based method, because this method is used by more researchers, which is more mature and more referential.

3.2.1 Binarization

The image with only two intensity values is called the binary image. This image usually shows only black or white and black is represented by 0 and white is represented by 1. Hence, it can separate the ridge line from the background of fingerprints.

The basic principle of binarization is to compare the pixel intensity with the threshold, and setting the pixel whose intensity is less than threshold to 0 and the other to 1. Therefore, the threshold is particularly important for binarization. Thresholds are divided into global thresholds and local thresholds, where global thresholds means that defining a single threshold for the whole image and local thresholds means changing the threshold locally by adapting the average local intensity, respectively.

The binarization process is fast and good results for high quality images can be exported through global thresholds. However, for low quality or degraded images, global binarization method may produce noises along image borders. To overcome this disadvantage, the local threshold technique is proposed and it can estimate different thresholds for each pixel according to the grayscale level of neighborhood pixels.

3.2.2 Binarization algorithm

The integral sum image [16] means that the intensity of a random pixel equals the sum of intensities of all pixels above and left. The diagrammatic sketch is shown in Figure 9.

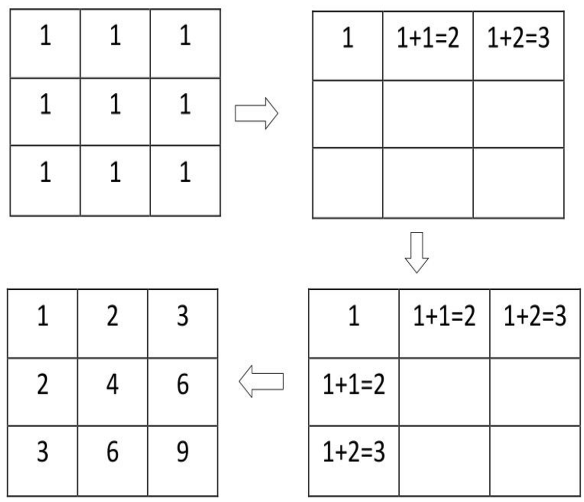


Figure 9: Calculation steps of integral sum image

For this figure, the intensity of integral sum image can be calculated as:

$$g\left(1,y\right)=I\left(1,y\right)+g\left(1,y-1\right),y=2,\cdots,n, \\ [\text{Intensity of } 1^{st} \text{ row } at\ (1,y)] \tag{19}$$

$$g\left(x,1\right)=I\left(x,1\right)+g\left(x-1,1\right),x=2,\cdots,m, \\ [\text{Intensity of } 1^{st} \text{ column } at\ (x,1)] \tag{20}$$

$$g\left(x,y\right)=I\left(x,y\right)+g\left(x,y-1\right)+g\left(x-1,y\right) \\ -g\left(x-1,y-1\right),\ y=2,\cdots,n,\ x=2,\cdots,m. \\ [\text{Intensity of all pixels at } (x,y)] \tag{21}$$

And the sum intensity in the center of local window (with size $w\times w$) $s(x,y)$ can be calculated as:

$$s\left(x,y\right)=\left[g\left(x+d-1,y+d-1\right)+g\left(x-d,y-d\right)\right] \\ +\left[g\left(x-d,y+d-1\right)+g\left(x+d-1,y-d\right)\right], \tag{22}$$

where $d=round(w/2)$ and w is an odd number.

The local mean $m(x,y)$ at (x,y) within the window $w\times w$ can be calculated as:

$$m\left(x,y\right)=\frac{s\left(x,y\right)}{w^2}. \tag{23}$$

The threshold $T(x,y)$ can be used in following equation:

$$b\left(x,y\right)=\begin{cases}0, & \text{if } I\left(x,y\right)\leq T\left(x,y\right)\ , \\ 1, & \text{otherwise}\ ,\end{cases} \tag{24}$$

where $b(x,y)$ is the intensity of binarized images.

And $T(x,y)$ can be calculated as:

$$T\left(x,y\right)=m\left(x,y\right)\left[1+k\left(\frac{\partial\left(x,y\right)}{1-\partial\left(x,y\right)}-1\right)\right], \tag{25}$$

where $\partial\left(x,y\right)=I\left(x,y\right)-m\left(x,y\right)$ and k is a constant that can control the level of adaptation of threshold.

There is an additional step that aims to get rid of lakes and islands in binary fingerprint image. Lakes and islands in binary fingerprint image means a small black or white region surrounded by the other opposite color. They can be removed by the function in the matlab: Icc=bwareaopen(Icc,C) , where C is the maximum size of the small region and any small region that is smaller than C pixels will be removed.

3.2.3 simulation of binarization

By using Matlab, the segmented image of fingerprint A and B is binarized, the result is shown in the following figures.

(a) Original fingerprint image

(b) Binarized image of fingerprint A

Figure 10: The binarization image of fingerprint A

(a) Original fingerprint image

(b) Binarized image of fingerprint B

Figure 11: The binarization image of fingerprint B

As shown in Figure 10 and Figure 11 gray-scale images are converted into binary images with only black and white after binarization. The small lakes and islands in the fingerprint A and B has disappeared. The profile of the image is clear, and its implementation is also beneficial to compressing data and thinning process.

3.3 Thinning

Thinning is the process that the ridge line thickness is reduced to one pixel width by deleting pixels at edge of ridge lines. The basic principle of thinning is to build deletion templates, and then compare the binary images with templates to determine whether pixels at the certain point should be deleted or not.

The thinning of fingerprint image is based on the binary image and the quality of the binary image have significant influence on the thinned image. Even a good binary algorithm may still leave small breaks, bridges between ridges and other artifacts, so choosing the appropriate pattern is the most critical step in this process. Using appropriate pattern can reduce the number of spikes that can result in extraction of false minutiae. After the thinning step, the skeleton image of fingerprints can be obtained. Ideally, the skeleton image should be positioned at the middle of the original image and maintain the connection of ridge lines.

3.3.1 Thinning algorithm

The thinning algorithm in this paper is based on hit-miss transformation that aims to compute the structure of series. The process is to hit image pixels with pattern. If the image is considered as a set, the operation of hit-miss transform can be described as below:

$$X \otimes S = X - X \odot S, \tag{26}$$

where X is the image set and S is the pattern set. $X \otimes S$ means the operations between X and S , which result in the output image and $X \odot S$ means the hit operation which result in a set of the pixels that should be deleted.

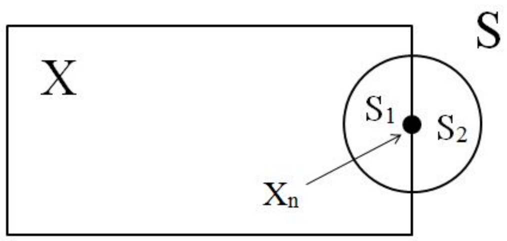


Figure 12: Hit/miss transformation example

Figure 12 is an example of hit/miss transformation, where S_1 and S_2 is the element in the set S. X_n is the element both in the set X and set S. If the S_1 belongs to set

X and the S_2 does not belong to set X, the result of $X \odot S$ should be the element X_n. Otherwise, the result of $X \odot S$ is nothing. After hitting all the pattern on the each element in the set X, the whole process will be completed.

When applying this hit/miss transform on the fingerprint image, the result of the operation will be the skeleton image of the fingerprint. Figure 13 shows the eight deletion pattern used in the operation.

0	0	0
*	1	*
1	1	1

(a)

0	*	1
0	1	1
0	*	1

(b)

1	1	1
*	1	*
0	0	0

(c)

1	*	0
1	1	0
1	*	0

(d)

*	0	0
1	1	0
*	1	*

(e)

0	0	*
0	1	1
*	1	*

(f)

*	1	*
0	1	1
0	0	*

(g)

*	1	*
1	1	0
*	0	0

(h)

Figure 13: Eight deletion patterns

P2	P1	P8
P3	P	P7
P4	P5	P6

Figure 14: Target pattern

Figure 14 is the target pattern which contain the center point P and the points around it(from P_1 to P_2). Then algorithm consists of the following steps:

1. A point is selected as the target point randomly and a target pattern is established with the target point and 8 neighborhood points.

2. Target patterns are compared with deletion templates, if they match the deletion template, then delete the target pixels, otherwise, reserve it.

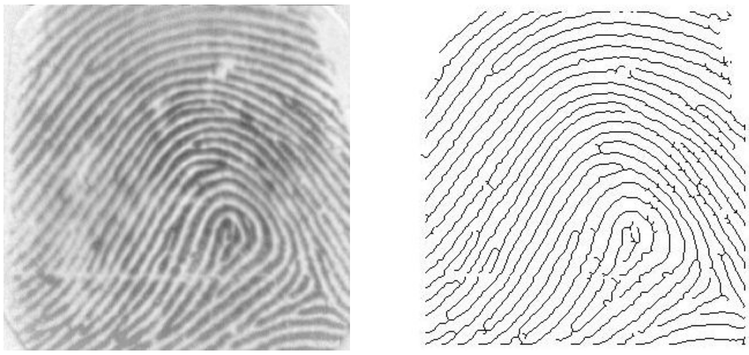
3. Repeat the above procedure until the pixel value of the fingerprint does not change.

3.3.2 simulation of thinning

Two different fingerprints are chosen for performance verification. Fingerprint A has more minutiae points and is complex, and fingerprint B is more smooth relatively. Thinning the binary images by using Matlab, such as the deletion pattern (c) can be achieved by following Matlab code. And the result images are shown as following:

```
% * * *    y11 y12 y13
% * @ *    y21 y22 y23
% * * *    y31 y32 y33

y11==1&&y12==1&&y13==1&&y31==0&&y32==0&&y33==0
```



(a) Original fingerprint image

(b) Thinning image of fingerprint A

Figure 15: The thinning image of fingerprint A

The binary image is transformed into fingerprint with single pixel width by thinning process successfully. However, there are still burrs, bifurcations and breakpoints on the image after thinning, which means simulation has some steps should be improved certainly. These disadvantages can be solved by reducing the deletion patterns or even the number of judgment times, which is not only simplifying the patterns, but also improving the speed of operation. This point can be researched in future.

3.4 Minutiae extraction algorithm

After getting the thinned fingerprint image, the next step is to use the neural network to extracts the correct minutiae points from the fingerprint image. This neural network has an input layer, a hidden layer and a output layer.

(a) Original fingerprint image

(b) Thinning image of fingerprint B

Figure 16: The thinning image of fingerprint B

- The input layer: The input layer consists of 9 neurons which is 3×3 pixel blocks from the fingerprint image.
- The hidden layer: The hidden layer consists 3×3 patterns of bifurcations and terminations.
- The output layer is a map which is the same size as the fingerprint image. In the map, 0 for non-minutiae points, 1 for termination point and 2 for bifurcation point.

This neural network is trained in off-line mode, because the training proceed only has one loop. There are known patterns for the termination points and the bifurcation points, which are shown in Figure 17 and Figure 18.

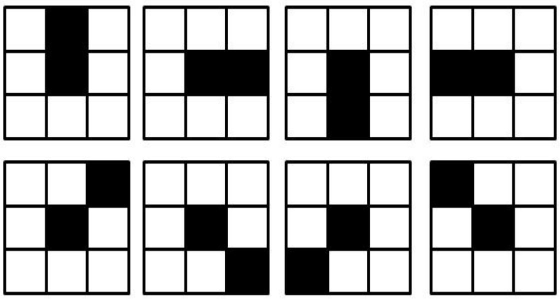


Figure 17: All the possible termination patterns that are used for neural network

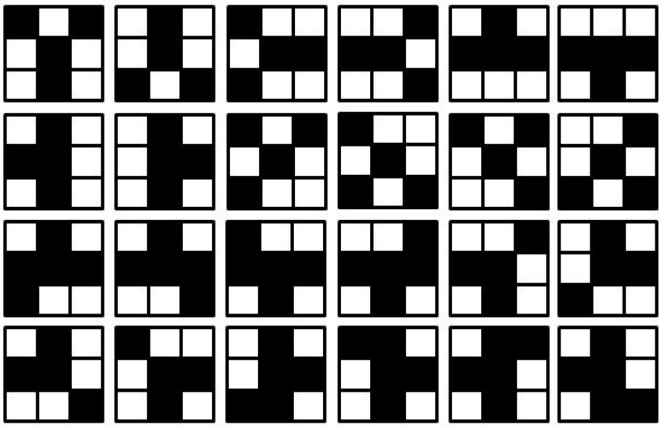


Figure 18: All the possible bifurcation patterns that are used for neural network

The neural network based minutiae extraction algorithm gives a map of the minutiae position. After marking the positions on the thinned fingerprint image, the following result can be achieved.

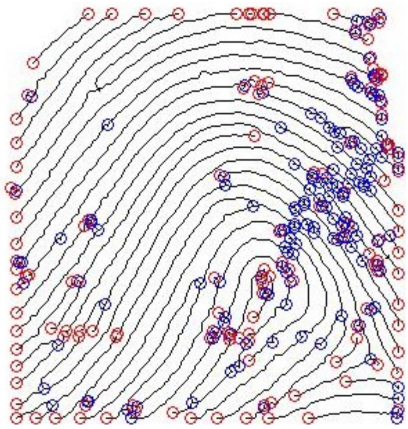


Figure 19: All the possible minutiae point in the fingerprint image

The Figure 19 shows all the possible minutiae points, where red circles present the termination point and the blue circles show the bifurcation points. While only part of them are true minutiae points, The other minutiae point are false minutiae point. They maybe caused by the edge of the fingerprint, some breaks on the ridge, some hooks on the ridge and so on.

According to the segmentation of the fingerprint, all the minutiae points between the foreground and background are false minutiae. In addition, there are some common structures in the fingerprint image which are false minutiae. The false minutiae showed

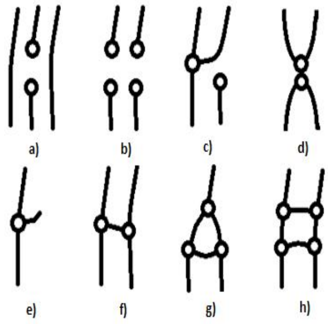


Figure 20: $a)$ to $b)$ are some false minutiae structures

in Figure 20 should be ignored. There is a common characteristic of these false minutiae, which is that these false minutiae are very close to each other. So we set up a few rules to ignore the false minutiae[2]:

- Rule 1 :if the distance between a termination and a bifurcation is smaller than D_1, then these two minutiae could be false minutiae. We should remove these two minutiae. Experimentally, D_1 is set to 10.
- Rule 2 :if the distance between two terminations is smaller than D_2, then these two minutiae could be false minutiae. We should remove these two minutiae. Experimentally, D_2 is set to 6.
- Rule 3 :if the distance between two bifurcations is smaller than D_3, then these two minutiae could be false minutiae. We should remove these two minutiae. Experimentally, D_3 is set to 6.

Figure 21 is the result of applying these rules on Figure 19. We can see that after applying the rules on the minutiae result, the false minutiae has a significant decrease.

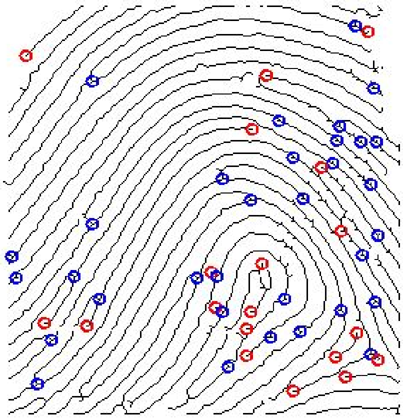
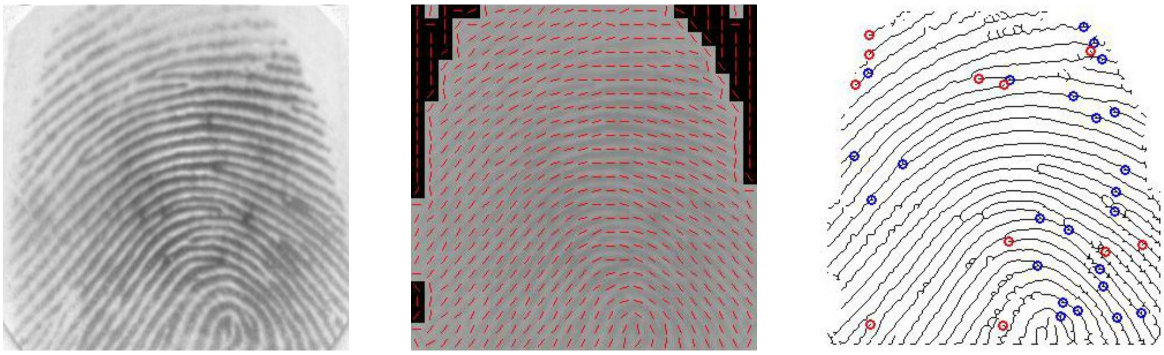


Figure 21: Ignoring false minutiae points

4 Analysis of simulation

The fingerprint image databases in this work is published by the organization of Fingerprint Verification Competition (FVC 2000, FVC 2002, FVC 2004, and FVC 2006) [8-11]. We use the database form the Fingerprint Verification Competition and choose 4 fingerprints with different characteristics as the input image samples for experimental results analysis in this section. Here these fingerprints are called as fingerprint A, fingerprint B, fingerprint C and fingerprint D and all of them are representative.

Firstly, Figure 22 and Figure 23 show the experimental results of fingerprint A and fingerprint B. The b) images in Figure 22 and Figure 23 well present the orientation

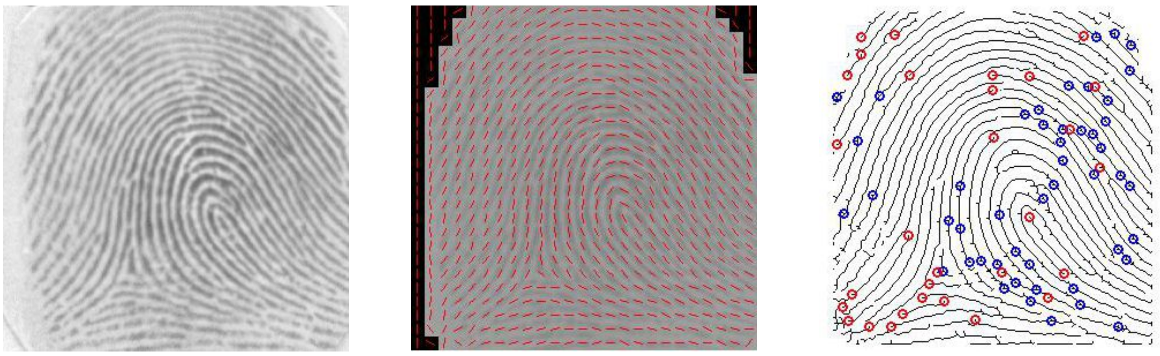


(a) Original image of fingerprint A

(b) Enhancement image of fingerprint A

(c) Thinned and minutiae extracted image of fingerprint A

Figure 22: The experimental result of fingerprint A



(a) Original image of fingerprint B

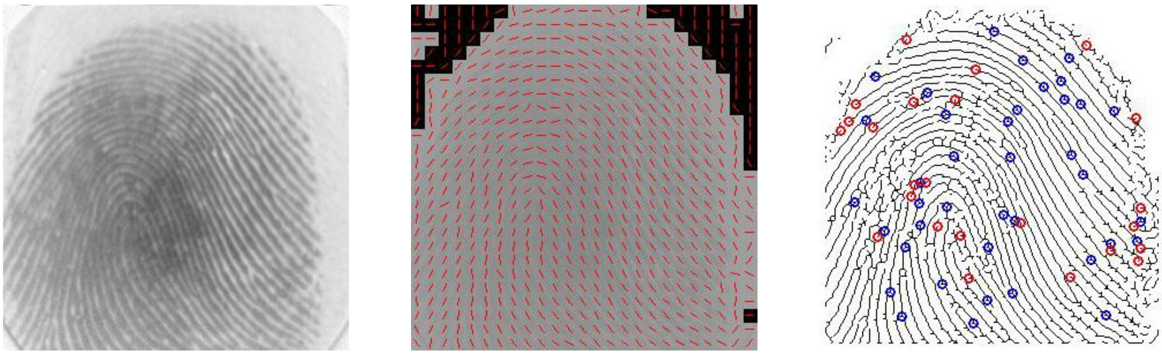
(b) Enhancement image of fingerprint B

(c) Thinned and minutiae extracted image of fingerprint B

Figure 23: The experimental result of fingerprint B

field of both fingerprint A and fingerprint B and separate each enhancement image into background and fingerprint itself. The c) image in Figure 22 finds almost all the true minutiae and succeed to eliminate almost all the false minutiae. However, although the c) image in Figure 23 finds nearly all the minutiae point, it still has some false minutiae points left. This problem is caused by the quantity of the spikes points. If more thinning patterns are added, the result could be improved and false minutiae could be decreased.

Secondly, Figure 24 shows the experimental results of fingerprint C:



(a) Original image of fingerprint C

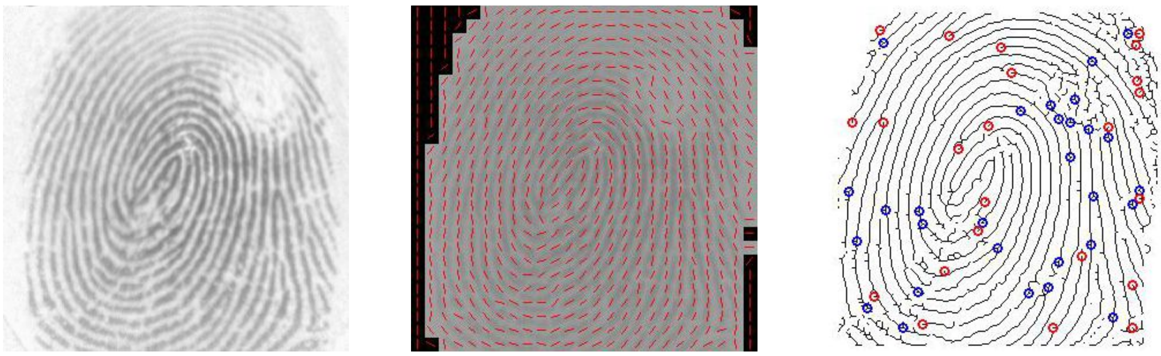
(b) Enhancement image of fingerprint C

(c) Thinned and minutiae extracted image of fingerprint C

Figure 24: The experimental result of fingerprint C

After seeing the a) image in Figure 24, it's obvious to see the main characteristic of fingerprint C, which is fingerprint C has a higher ridge frequency than normal fingerprint. It can be seen in b) image that this characteristic may not influence the enhancement while it will have a big impact in thinning process. Some mess thinned ridges may occur in small regions in fingerprint and it will add a lot of difficulties in minutiae extraction.

Thirdly, Figure 25 shows the experimental results of fingerprint D: On the top-left section of fingerprint D image, there is a small region in which ridges can barely be totally recognized because of distortion. This region is called unrecoverable corrupted region in many scientific research paper[5] and it is not the characteristic of fingerprint D itself. The unrecoverable corrupted region may be caused by various factors such as fingerprint scanning mistake, poor skin and ridge condition (occupational mark) and improper way of placing fingers. Even the neighboring regions can be influenced by it, which result in insufficient information for ridges. Usually this unrecoverable region should be marked out and separate it from interested region(well-defined region and recoverable region).



(a) Original image of fingerprint D

(b) Enhancement image of fingerprint D

(c) Thinned and minutiae extracted image of fingerprint D

Figure 25: The experimental result of fingerprint D

5 Summary and conclusion

All the algorithms that are discussed in this paper belong to fast fingerprint enhancement and fast minutiae extraction. The fingerprint enhancement improves the quality of the ridge image and eliminate the noise in the fingerprint image, which helps to make up the disadvantages in the minutiae extraction. Minutiae extraction algorithms are fast, since they use less patterns compared with some accurate image analysis algorithms. If the image is not processed, the fuzzy parts and noise may cause a lot of false minutiae and true minutiae missing. Hence, the combination of fast fingerprint enhancement and minutiae extraction in this paper is a good choice for fast fingerprint recognition system.

It's a pity to let the matching algorithm missing in this paper because of the time limitation. The plan was to discuss a fast matching algorithm for the whole system. Once the whole fingerprint recognition system is completed, it would be very efficient to deal with large fingerprint database. Only a little time would be needed to recognize the owner of the fingerprint in the large fingerprint database. Then the only thing needed to be improved would be how to decrease the error-rate in minutiae extraction and matching algorithm.

At last, we want to thank our supervisor Sven Nordebo for helping us a lot in the fingerprint image processing study and research. And we also want to thank the Biometric Recognition Group - ATVS for providing FVC 2006 fingerprint database to conduct this research.

References

[1] Hong,L., Wan,Y., and Jain, A. K.(1998). Fingerprint image enhancement: algorithm and performance evaluation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.20(8), pp.777-789.

[2] Pokhriyal, A. and Lehri, S.(2010). MERIT: Minutiae Extraction using Rotation Invariant Algorithm, *International Journal of Engineering Science and Technology*, Vol.2(7), pp.3225-3235.

[3] Pu, H., Chen, J. and Zhang, Y.(2007). Fingerprint Thinning Algorithm Based on Mathematical Morphology, *2007 8th International Conference on Electronic Measurement and Instruments*, pp.2-618-2-621.

[4] Wang, F., Wang, X. and Xu, L.(2009). An Improved Fingerprint Segmentation Algorithm Based on Mean and Variance, *International Workshop on Intelligent Systems and Applications*, pp.1-4.

[5] Alex, A.(2004),Handbook of Fingerprint Recognition, *Emerald Group Publishing Limited*.

[6] Mei, Y., Cao, G., Sun, H., Hou, R.(2012). A systematic gradient-based method for the computation of fingerprint's orientation field, *Computers and Electrical Engineering*, Vol.38(5), pp.1035-1046.

[7] Roli, B., Priti, S. and Punam, B.(2011). Minutiae Extraction from Fingerprint Images - a Review, *International Journal of Computer Science Issues (IJCSI)*, Vol.8(5), pp.74-85.

[8] Cappelli, R., Ferrara, M., Franco, A. and Maltoni, D.(2007). Fingerprint verification competition 2006, *Biometric Technology Today*, vol.15, pp.7-9.

[9] Fierrez, J., Ortega-Garcia, J., Torre-Toledano, D. and Gonzalez-Rodriguez, J.(2007). BioSec baseline corpus: A multimodal biometric database, *Pattern Recognition*, Vol. 40, n. 4, pp. 1389-1392.

[10] Maio, D., Malton, D., Capelli, R., Wayman, J. and Jain, A.(2002). FVC 2002: second fingerprint verification competition, *16th International Conference on Pattern Recognition*, vol. 3, pp. 811-814.

[11] Maio, D., Maltoni, D., Capelli, R., Wayman, J. and Jain, A.(2004). FVC 2004: Third Fingerprint Verification Competition, *Lecture Notes in Computer Science*, vol. 3072, pp.1-7.

[12] de Magalhaes, S. T. and Hessami, H. J. A. G.(2010). Analysis of Fingerprint Image to Verify a Person, *Global Security, Safety, and Sustainability: 6th International Conference*, ICGS3, pp.104-119.

[13] Shlomo, G., Mayer, A. and Daniel, K.(2002). Fingerprint image enhancement using filtering techniques, *Real-Time Imaging*, Vol.8(3), pp.227-236.

[14] Maio, D. and Maltoni, D.(1997). Direct gray-scale minutiae detection in fingerprints, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume: 19, Issue: 1.

[15] Chen, G., Chen, N. and Zeng Y.(2012). An improved OPTA fingerprint thinning algorithm based on neighborhood searching, *International Conference on Computer Science and Information Processing (CSIP)*, pp.637-640.

[16] Singh, T.R., Roy, S., Singh, O.I., Sinam, T., Singh, K.M.(2011). A New Local Adaptive Thresholding Technique in Binarization, *IJCSI International Journal of Computer Science Issues*, Vol.8, Issue 6, No 2.

Appendix

This is the code for the main system:

```
clc
clf
clear all
close all
originI=imread('101_7.tif');
[m,n,s] = size(originI);

I=originI;


if s == 3
    I = rgb2gray(originI);
end
I=double(I);
I1=I;
figure, imshow(uint8(I))
tic

%normalization
M=0;
var=0;
for x=1:m
    for y=1:n
        M=M+I(x,y);
    end
end
M1=M/(m*n);
for x=1:m
    for y=1:n
        var=var+(I(x,y)-M1)*(I(x,y)-M1);
    end
end
var1=var/(300*300);
for x=1:m
    for y=1:n
        if I(x,y)≥M1
            I(x,y)=150+sqrt(50*(I(x,y)-M1)*(I(x,y)-M1)/var1);
        else
```

```
            I(x,y)=150-sqrt(50*(I(x,y)-M1)*(I(x,y)-M1)/var1);
        end
    end
end

figure, imshow(uint8(I))




w=12;
Gx=zeros(m,n);
Gy=zeros(m,n);
for x=2:m-1
    for y=2:n-1
    Gx(x,y)=(I(x+1,y)-I(x-1,y));
    Gy(x,y)=(I(x,y+1)-I(x,y-1));
    end
end

%segmentation
M = 12;
H = m/M;
L= n/M;
aveg1=zeros(H,L);
var1=zeros(H,L);
for x=1:H;
    for y=1:L;
        aveg=0;
        var=0;
        for i=1:M;
            for j=1:M;
                aveg=I(i+(x-1)*M,j+(y-1)*M)+aveg;
            end
        end
        aveg1(x,y)=aveg/(M*M);
        for i=1:M;
            for j=1:M;
                var=(I(i+(x-1)*M,j+(y-1)*M)-aveg1(x,y))
                *(I(i+(x-1)*M,j+(y-1)*M)-aveg1(x,y))+var;
            end
        end
        var1(x,y)=var/(M*M);
    end
```

```
end
Gmean=0;
Vmean=0;
for x=1:H
    for y=1:L
        Gmean=Gmean+aveg1(x,y);
        Vmean=Vmean+var1(x,y);
    end
end
Gmean=Gmean/(H*L);
Vmean=Vmean/(H*L);
NGf=0;
TGf=0;
NVf=0;
TVf=0;
for x=1:H
    for y=1:L
        if Gmean<aveg1(x,y)
            NGf=NGf+1;
            TGf=TGf+aveg1(x,y);
        end
        if Vmean<var1(x,y)
            NVf=NVf+1;
            TVf=TVf+var1(x,y);
        end
    end
end
Gf=TGf/NGf;
Vf=TVf/NVf;
NGb=0;
TGb=0;
TVb=0;
NVb=0;
for x=1:H
    for y=1:L
        if Gf<aveg1(x,y)
            NGb=NGb+1;
            TGb=TGb+aveg1(x,y);
        end
        if var1(x,y)<Vf
            NVb=NVb+1;
            TVb=TVb+var1(x,y);
        end
    end
```

```
end
Gb=TGb/NGb;
Vb=TVb/NVb;
ground=zeros(H,L);
T1=Gb;
T2=Vb;

for x=1:H
    for y=1:L
        if aveg1(x,y)>T1 && var1(x,y)<T2
            ground(x,y)=1;
        end
    end
end

for x=2:H-1
    for y=2:L-1
        if ground(x,y)==1
            if ground(x-1,y) + ground(x-1,y+1) +ground(x,y+1) + ...
                ground(x+1,y+1) + ground(x+1,y) + ground(x+1,y-1) + ...
                ground(x,y-1) + ground(x-1,y-1) ≤4
                 ground(x,y)=0;
            end
        end
    end
end

Icc = ones(m,n);

for x=1:H
    for y=1:L
        if  ground(x,y)==1
            for i=1:M
                for j=1:M
                    I(i+(x-1)*M,j+(y-1)*M)=0;
                    Icc(i+(x-1)*M,j+(y-1)*M)=0;
                end
            end
        end
    end
end

figure, imshow(uint8(I))

```

```
%orientation eatimation
Gsx=zeros(m,n);
Gsy=zeros(m,n);

for x=2:m-1
    for y=2:n-1
    Gsx(x,y)=Gx(x,y)^2-Gy(x,y)^2;
    Gsy(x,y)=2*Gx(x,y)*Gy(x,y);
    end
end

Gbx2=zeros(25,25);
Gby2=zeros(25,25);
for h=1:25
    for g=1:25
        for x=1+(h-1)*w:h*w
           for y=1+(g-1)*w:g*w
           Gbx2(h,g)=Gbx2(h,g)+Gsx(x,y);
           Gby2(h,g)=Gby2(h,g)+Gsy(x,y);
           end
        end
    end
end

for h=1:25
    for g=1:25
        if Gbx2(h,g)>0
           theta2(h,g)=pi/2+atan(Gby2(h,g)/Gbx2(h,g))/2;
        end
        if Gbx2(h,g)<0 && Gby2(h,g)≥0
           theta2(h,g)=pi/2+(atan(Gby2(h,g)/Gbx2(h,g))+pi)/2;
        end
        if Gbx2(h,g)<0 && Gby2(h,g)<0
           theta2(h,g)=pi/2+(atan(Gby2(h,g)/Gbx2(h,g))-pi)/2;
        end
    end
end

for h=1:25
    for g=1:25
        theta3(h,g)=atan(Gby2(h,g)/Gbx2(h,g))/2;
    end
end
```

```
%orientation field filtering
for h=1:25
    for g=1:25
     Phix(h,g)=cos(2*theta2(h,g));
     Phiy(h,g)=sin(2*theta2(h,g));
    end
end
f = fspecial('gaussian', 10, 2);
Phix=filter2(f, Phix);
Phiy=filter2(f, Phiy);
for h=1:25
    for g=1:25
    O1(h,g)=1/2*atan2(Phiy(h,g),Phix(h,g));
    end
end

figure(1)
hold on

for m=0:24
    for n=0:24

        plot(12*(n+0.5+1i*(m+0.5)+1i*10^(-20)+0.3*[-1 ...
            1]*exp(1i*(3/2*pi-O1(m+1,n+1)))),'r-')
    end
end

%Binarization
[m,n,s] = size(I);
temp=(1/9)*[1 1 1;1 1 1;1 1 1];
Im=double(I);
In=zeros(m,n);

for a=2:m-1;
    for b=2:n-1;
        In(a,b)=Im(a-1,b-1)*temp(1,1)+Im(a-1,b)*temp(1,2)+Im(a-1,b+1)
        *temp(1,3)+Im(a,b-1)*temp(2,1)+Im(a,b)*temp(2,2)
        +Im(a,b+1)*temp(2,3)+Im(a+1,b-1)*temp(3,1)+Im(a+1,b)*temp(3,2)
        +Im(a+1,b+1)*temp(3,3);
    end
end

I=In;
```

```
Im=zeros(m,n);
for x=5:m-5;
    for y=5:n-5;
        sum1=I(x,y-4)+I(x,y-2)+I(x,y+2)+I(x,y+4);
        sum2=I(x-2,y+4)+I(x-1,y+2)+I(x+1,y-2)+I(x+2,y-4);
        sum3=I(x-2,y+2)+I(x-4,y+4)+I(x+2,y-2)+I(x+4,y-4);
        sum4=I(x-2,y+1)+I(x-4,y+2)+I(x+2,y-1)+I(x+4,y-2);
        sum5=I(x-2,y)+I(x-4,y)+I(x+2,y)+I(x+4,y);
        sum6=I(x-4,y-2)+I(x-2,y-1)+I(x+2,y+1)+I(x+4,y+2);
        sum7=I(x-4,y-4)+I(x-2,y-2)+I(x+2,y+2)+I(x+4,y+4);
        sum8=I(x-2,y-4)+I(x-1,y-2)+I(x+1,y+2)+I(x+2,y+4);
        sumi=[sum1,sum2,sum3,sum4,sum5,sum6,sum7,sum8];
        summax=max(sumi);
        summin=min(sumi);
        summ=sum(sumi);
        b=summ/8;
        if  (summax+summin+4*I(x,y))> ...
           (3*(sum1+sum2+sum3+sum4+sum5+sum6+sum7+sum8)/8)
            sumf = summin;
        else
            sumf = summax;
        end
        if   sumf > b
            Im(x,y)=128;
        else
            Im(x,y)=255;
        end
    end
end

for i=1:m
    for j =1:n
        Icc(i,j)=Icc(i,j)*Im(i,j);
    end
end

for i=1:m
    for j =1:n
        if (Icc(i,j)==128)
            Icc(i,j)=0;
        else
            Icc(i,j)=1;
        end;
    end
```

```
end

figure,imshow(double(Icc))

Icc=bwareaopen(Icc,80);  %remove lake
Icc=¬Icc;
Icc=bwareaopen(Icc,80);  %remove island
Icc=¬Icc;

Icc=imdilate(Icc,[1 1; 1 1]);
figure,imshow(double(Icc))
Im=Icc;
In=Im;
for a=1:4
    for i=2:m-1
        for j=2:n-1
            if Im(i,j)==1
                if Im(i-1,j) + Im(i-1,j+1) +Im(i,j+1) + Im(i+1,j+1) ...
                    + Im(i+1,j) + Im(i+1,j-1) + Im(i,j-1) + ...
                    Im(i-1,j-1) ≤3
                     In(i,j)=0;
                end
            end
            if Im(i,j)==0
                if Im(i-1,j) + Im(i-1,j+1) +Im(i,j+1) + Im(i+1,j+1) ...
                    + Im(i+1,j) + Im(i+1,j-1) + Im(i,j-1) + ...
                    Im(i-1,j-1) ≥7
                     In(i,j)=1;
                end
            end
        end
    end
    Im=In;
end
figure,imshow(double(Im))

%Thinning
Icc=thinning1(Icc);
figure,imshow(Icc);

%Minutiae extraction
Mi=zeros(m,n);
Mi=minu(Icc);

```

```
%false minutiae deletion
for m=1:300
    for n=1:300
        if Mi(m,n)==1||Mi(m,n)==2
            d=20;
            if m<d||m+d>300||n<d||n+d>300;
              Mi(m,n)=0;
            else
            end
        else
        end
    end
end

for n=1:300
    for m=1:300
        if Mi(m,n)==1||Mi(m,n)==2
            for i=1:300
                for j=1:300
                   if Mi(i,j)==1||Mi(i,j)==2
                    if Mi(m,n)==Mi(i,j)
                       d=10;
                    else
                        d=5;
                    end
                    a=sqrt((m-i)^2+(n-j)^2);
                    if a<d&&a>0;
                     Mi(m,n)=0;
                     Mi(i,j)=0;
                    else
                    end
                   end
                end
            end
        else
        end
    end
end

figure(7)
hold on
for m=1:300
    for n=1:300
        if Mi(m,n)==1
```

```
            a=round((m-1)/12)+1;
            b=round((n-1)/12)+1;
            plot(1*(n++1i*(m)),'ro','LineWidth',2)

         elseif Mi(m,n)==2
            a=round((m-1)/12)+1;
            b=round((n-1)/12)+1;
            plot(1*(n++1i*(m)),'bo','LineWidth',2)

         end
     end
end
```

This is the thinning function:

```
function Y=thinning1(X)
[M,N]=size(X);
Y=zeros(M,N,20);
Z=zeros(M,N,20);
Y(:,:,1)=X;
Z(:,:,1)=X;
for k=2:20;
    Y(:,:,k)=Y(:,:,k-1);
    for i=2:M-1;
        for j=2:N-1;
% * * *     y11 y12 y13
% * @ *     y21 y22 y23
% * * *     y31 y32 y33
             y11=Y(i-1,j-1,k);
             y12=Y(i-1,j,k);
             y13=Y(i-1,j+1,k);
             y21=Y(i,j-1,k);
             y22=Y(i,j,k);
             y23=Y(i,j+1,k);
             y31=Y(i+1,j-1,k);
             y32=Y(i+1,j,k);
             y33=Y(i+1,j+1,k);
             if y22==0;
              if y11==1&&y12==1&&y13==1&&y31==0&&y32==0&&y33==0;
                   Y(i,j,k)=1;

              elseif y11==1&&y13==0&&y21==1&&y23==0&&y31==1&&y33==0;
                   Y(i,j,k)=1;
```

```
                elseif y11==0&&y12==0&&y13==0&&y31==1&&y32==1&&y33==1;
                     Y(i,j,k)=1;
                elseif y11==0&&y13==1&&y21==0&&y23==1&&y31==0&&y33==1;
                     Y(i,j,k)=1;
                elseif y12==1&&y13==1&&y21==0&&y23==1&&y32==0;
                     Y(i,j,k)=1;

                elseif y11==1&&y12==1&&y21==1&&y23==0&&y32==0;
                     Y(i,j,k)=1;
                elseif y12==0&&y21==1&&y23==0&&y31==1&&y21==1;
                     Y(i,j,k)=1;
                elseif y12==0&&y21==0&&y23==1&&y32==1&&y33==1;
                     Y(i,j,k)=1;

               %extra
                elseif y11==0&&y12==0&&y23==1&&y31==1&&y32==1&&y33==1;
                     Y(i,j,k)=1;

                end
               else
               end
           end
       end


       if Y(:,:,k)==Y(:,:,k-1);
           break;
       end
   end
   Y=Y(:,:,k);
```

This is the minutiae extraction function:

```
function Y=minu(X)
Y(:,:)=X;

[M,N]=size(X);
mi=zeros(M,N);
    for i=2:M-1;
        for j=2:N-1;
% * * *     y11 y12 y13
% * @ *     y21 y22 y23
% * * *     y31 y32 y33
```

```
            y11=Y(i-1,j-1);
            y12=Y(i-1,j);
            y13=Y(i-1,j+1);
            y21=Y(i,j-1);
            y22=Y(i,j);
            y23=Y(i,j+1);
            y31=Y(i+1,j-1);
            y32=Y(i+1,j);
            y33=Y(i+1,j+1);
            if y22==0;
             if y11+y12+y13+y23+y33+y31+y21+y32==7;
                  mi(i,j)=1;

             elseif y11==0&&y12==1&&y13==0&&y21==1
                 &&y23==1&&y31==1&&y32==0&&y33==1;
                 mi(i,j)=2;
             elseif y11==1&&y12==0&&y13==1&&y21==1
                 &&y23==1&&y31==0&&y32==1&&y33==0;
                 mi(i,j)=2;
             elseif y11==0&&y12==1&&y13==1&&y21==1
                 &&y23==0&&y31==0&&y32==1&&y33==1;
                 mi(i,j)=2;
             elseif y11==1&&y12==1&&y13==0&&y21==0
                 &&y23==1&&y31==1&&y32==1&&y33==0;
                 mi(i,j)=2;
             elseif y11==1&&y12==0&&y13==1&&y21==0
                 &&y23==0&&y31==1&&y32==1&&y33==1;
                 mi(i,j)=2;

             elseif y11==1&&y12==1&&y13==1&&y21==0
                 &&y23==0&&y31==1&&y32==0&&y33==1;
                 mi(i,j)=2;
             elseif y11==1&&y12==0&&y13==1&&y21==0
                 &&y23==1&&y31==1&&y32==0&&y33==1;
                 mi(i,j)=2;
             elseif y11==1&&y12==0&&y13==1&&y21==1
                 &&y23==0&&y31==1&&y32==0&&y33==1;
                 mi(i,j)=2;
             elseif y11==0&&y12==1&&y13==0&&y21==1
                 &&y23==1&&y31==0&&y32==1&&y33==1;
                 mi(i,j)=2;
             elseif y11==0&&y12==1&&y13==1&&y21==1
                 &&y23==1&&y31==0&&y32==1&&y33==0;
                 mi(i,j)=2;
```

```

        elseif y11==0&&y12==1&&y13==0&&y21==1
            &&y23==1&&y31==1&&y32==1&&y33==0;
            mi(i,j)=2;
        elseif y11==1&&y12==1&&y13==0&&y21==1
            &&y23==1&&y31==0&&y32==1&&y33==0;
            mi(i,j)=2;
        elseif y11==1&&y12==0&&y13==1&&y21==0
            &&y23==0&&y31==0&&y32==1&&y33==1;
            mi(i,j)=2;
        elseif y11==1&&y12==0&&y13==1&&y21==0
            &&y23==0&&y31==1&&y32==1&&y33==0;
            mi(i,j)=2;
        elseif y11==0&&y12==1&&y13==1&&y21==0
            &&y23==0&&y31==1&&y32==0&&y33==1;
            mi(i,j)=2;

        elseif y11==1&&y12==1&&y13==0&&y21==0
            &&y23==0&&y31==1&&y32==0&&y33==1;
            mi(i,j)=2;
        elseif y11==1&&y12==1&&y13==0&&y21==0
            &&y23==1&&y31==1&&y32==0&&y33==1;
            mi(i,j)=2;
        elseif y11==1&&y12==0&&y13==1&&y21==1
            &&y23==0&&y31==0&&y32==1&&y33==1;
            mi(i,j)=2;
        elseif y11==1&&y12==0&&y13==1&&y21==0
            &&y23==1&&y31==1&&y32==1&&y33==0;
            mi(i,j)=2;
        elseif y11==0&&y12==1&&y13==1&&y21==1
            &&y23==0&&y31==1&&y32==0&&y33==1;
            mi(i,j)=2;

        elseif y11==1&&y12==0&&y13==1&&y21==1
            &&y23==0&&y31==0&&y32==0&&y33==1;
            mi(i,j)=2;
        elseif y11==0&&y12==0&&y13==1&&y21==1
            &&y23==0&&y31==1&&y32==0&&y33==1;
            mi(i,j)=2;
        elseif y11==1&&y12==0&&y13==0&&y21==0
            &&y23==1&&y31==1&&y32==0&&y33==1;
            mi(i,j)=2;
        elseif y11==1&&y12==0&&y13==1&&y21==0
            &&y23==1&&y31==1&&y32==0&&y33==0;
```

```
                    mi(i,j)=2;
                end
              else
              end

        end
    end


Y=mi(:,:);
```