



Minutiae Extraction from Fingerprint with Neural Network and Minutiae based Fingerprint Verification

Josef Ström Bartůněk

Examensarbete
Teknologie Magisterexamen i Elektroteknik

Blekinge Tekniska Högskola
Mars 2005

Abstract

Human fingerprints are rich in details called minutiae, which can be used as identification marks for fingerprint verification. The goal of this thesis is to develop a complete system for fingerprint verification through extracting and matching minutiae. A neural network is trained using the back-propagation algorithm and will work as a classifier to locate various minutiae. To achieve good minutiae extraction in fingerprints with varying quality, preprocessing in form of binarization and skeletonization is first applied on fingerprints before they are evaluated by the neural network. Extracted minutiae are then considered as a 2D point pattern problem and an algorithm is used to determine the number of matching points between two point patterns. Performance of the developed system is evaluated on a database with fingerprints from different people and experimental results are presented.

Contents

1	Introduction	9
2	Binarization	15
2.1	Filter mask design	16
2.2	Orientation and mask image	19
2.3	Orientation image smoothing	22
2.4	Pixel-by-pixel enhancement filtering	24
2.5	Post processing	25
2.6	Summary	26
3	Skeletonization	29
3.1	Overview	29
3.2	Realization	31
4	Minutiae Extraction	35
4.1	Neural network	35
4.2	Back-propagation algorithm	36
4.3	Classifier realization	40
5	Minutiae Matching	45
5.1	Determination of the principal pair	46
5.2	Determination of the matching pairs	48
5.3	Testing the matching algorithm	51
6	Experimental Results	55
7	Summary and Conclusions	59
7.1	Further Work	59
	Bibliography	61

List of Figures

1.1	<i>left: different types of minutiae; right: real fingerprint</i>	11
1.2	<i>Fingerprints with varying quality.</i>	12
1.3	<i>Block-diagram of the total system design.</i>	13
2.1	<i>Detailed block-schema of the binarization process.</i>	16
2.2	<i>Horizontal oriented 11×11 filter mask for: $a_0 = 1000$, $R_{max} = 5$, $R_{min} = 3$, $V_{min} = 2$.</i>	18
2.3	<i>Estimated orientation weights $p(0^\circ)$, $p(60^\circ)$, $p(120^\circ)$ and mean value.</i>	20
2.4	<i>Example of mask image and how well it separates the fingerprint from the background.</i>	21
2.5	<i>Orientation image estimated from $p(0^\circ)$, $p(60^\circ)$ and $p(120^\circ)$ specified in degrees.</i>	22
2.6	<i>Decimated and smoothed orientation image to only 18 different levels, given in index numbers.</i>	23
2.7	<i>Orientation image expressed by lines, which represents the filter orientations. A closer inspection shows how well the lines aligns with the ridges.</i>	24
2.8	<i>Binarized fingerprint image.</i>	25
2.9	<i>Effect of the post processing.</i>	26
2.10	<i>Advantages and disadvantages of the binarization process. Rings in the two images indicate most unwanted effects and ellipse highlight most positive effect.</i>	27
3.1	<i>3×3 pixel mask of the nearest neighbors mapped around the P_i.</i>	29
3.2	<i>Examples of $0 \rightarrow 1$ patterns in the neighborhood.</i>	30
3.3	<i>Patterns there P_i is removed in the second skeletonization.</i>	31
3.4	<i>Flow diagram of the skeletonization process.</i>	32
3.5	<i>The steps between each iteration in the skeletonization process.</i>	33
3.6	<i>Intersection between the binarized fingerprint and its skeleton.</i>	34

4.1	<i>Simplified model of the neuron and the structure of the neural network.</i>	36
4.2	<i>The bipolar sigmoid activation function. The * marks $f(-1) = -1$ and $f(1) = 1$.</i>	39
4.3	<i>Different window sizes for the training data.</i>	40
4.4	<i>Examples of the training data. Upper row: termination patterns, middle row: bifurcation patterns and lower row: no minutiae patterns.</i>	41
4.5	<i>The "sum of squared errors" (SSE) error functions normalized with size of training data for three different neural networks. The "hl1 50" denotes one hidden layer with 50 neurons, "hl1 60" denotes one hidden layer with 60 neurons and "hl2 2525" denotes neural network with two hidden layers with 25 neuron in each layer.</i>	42
4.6	<i>MAXNET error function indicates how many patterns are left to be learned by the neural network. Error is 0 after 610 epochs.</i>	43
4.7	<i>Result of how well the neural network works as a classifier. Green boxes are the marked terminations. Red boxes are the marked bifurcations. Accuracy of the neural network can be seen in the figure b.</i>	44
5.1	<i>An example of point pattern matching between two fingerprints that are originating from the same person. A 58 and 63 point are searched for matching points. In the upper row are the point patterns before matching and in the lower row are the point patterns after matching. The blue boxes indicates the matching pairs and the principal pair is highlighted with a black box. Totally 48 pairs has been found which is 83% and $E_{ms} = 21$</i>	52
5.2	<i>An example of point pattern matching between two fingerprints that are originating from the different persons. A 54 and 63 point are searched for matching points. In the upper row are the point patterns before matching and in the lower row are the point patterns after matching. The blue boxes indicate the matching pairs and the principal pair is highlighted with a black box. Totally 25 pairs has been found which is 46% and $E_{ms} = 84$</i>	53
6.1	<i>Position of the examined fingerprint pairs for the two groups, plotted in the matched minutiae vs. mean squared error plane. The matched minutiae are given in the percentage.</i>	56

List of Tables

6.1	<i>Results for the matching of the identical fingerprint.</i>	55
6.2	<i>Results for the matching of the non-identical fingerprint. . . .</i>	56

Chapter 1

Introduction

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 not two persons have the same fingerprints, hence they are unique for every individual [1].

Above mentioned properties that human fingerprints have, made them 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. Today there exist many different automatic systems with various algorithms that solve this pattern recognition problem with very good results. However, this doesn't means that automatic fingerprint recognition is a fully solved problem. Designing algorithms that gives robust solutions to this particular problem is still a difficult and challenging task.

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. Also the many years of satisfying experience in law enforcement, using the fingerprints as identification marks. The observations showed that the fingerprints offer more secure and reliable person identification than keys,

passwords or id-cards can provide. With decreasing cost of fingerprint readers and cheap increasing computer power, automatic fingerprint recognition gives an efficient and inexpensive alternative to ordinary solutions in person identification [1]. 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.

Intention with this master thesis is to get deeper theoretical and practical understanding on how automatic fingerprint recognition is done. The goal is to have fully functional system that can distinguish if two fingerprints originate from the same person or not at end of the project, implemented in MATLAB. Because of big interest in neural networks and their application in real world problems, they will be used as at least part of the solution in the whole system. In the end, performance of developed system is evaluated on a database with fingerprints from different people and experimental results are presented.

There are many fingerprint matching methods that can be applied for fingerprint recognition, unfortunately the limited time do not allow examination of them all. Another limitation is that this thesis takes only to consideration so called *one-to-one* matching method, otherwise also called *verification*. This means that the comparison is performed only once and that is between template (the pre-stored fingerprint(s) on a safe place in the system) and fingerprint of the person to confirm that his identity is the same as the templates.

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. Most common minutiae that are found in fingerprints can be seen in **Figure 1.1** (on the left)[1]. To illustrate occurrence of those in a real fingerprint take a look at **Figure 1.1** (on the right). The dark (black) lines are the ridges and light (white) lines are the valleys. After a closer examination there can be seen that the two most important minutiae are *termination* and *bifurcation*, rest of the minutiae are only combinations of those two types!

So the fingerprint verification problem can be divided in two main tasks:

1. Minutiae extraction from fingerprints.
2. Minutiae matching.




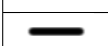
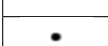

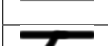
	Termination
	Bifurcation
	Lake
	Independent ridge
	Point or island
	Spur
	Crossover



Figure 1.1: *left: different types of minutiae; right: real fingerprint*

Quality of fingerprints can significantly vary, mainly due to skin condition and pressure made by contact of fingertip on sensing device some sort of preprocessing is needed to achieve good minutiae extraction. Some examples of different fingerprints with varying quality can be found in **Figure 1.2**. This problem can be handled by applying an enhancing algorithm that is able to separate and highlight the ridges from background; this type of enhancing is also called *binarization*. Detailed description on whole subject is described in chapter 2.

A more effective and faster minutiae extraction realization can be achieved by minimizing data that represents minutiae without corrupting it. Since minutiae are determined only by discontinuities in ridges, they are totally independent of ridges thickness. Thinning of the ridges to only 1-pixel wide lines also called skeletons, not only preserves minutiae but it does it with

minimum possible data usage. The thinning method is often called *skeletonization*. Detailed description on whole subject is described in chapter 3.



Figure 1.2: *Fingerprints with varying quality.*

Given the binarized and skeletonized image, can the task *minutiae extraction from fingerprint* be seen as a typical classification problem. This can be solved by training a neural network that works as a classifier. After the training the different types and shapes of various minutiae that has been learned can be recognized and classified by the neural network. Detailed description

on whole subject is described in chapter 4.

The second task, *minutiae matching* can be seen as a 2D point pattern problem. A matching algorithm is needed to determine the number of matching points between the two point patterns. The algorithm should be able to find optimal translation in x and y coordinates, rotation and possible scale difference between the two point patterns. Through those parameters the maximum matching points can be found which is essential to assure good accuracy and performance for fingerprint verification. Detailed description on whole subject is described in chapter 5.

Block-diagram of the automatic fingerprint verification system and its diverse parts can be seen in **Figure 1.3**. The template is a pre-stored point pattern of extracted minutiae from authentic fingerprint. It is produced in the same way as the point pattern of the fingerprint in the **Figure 1.3** on the left.

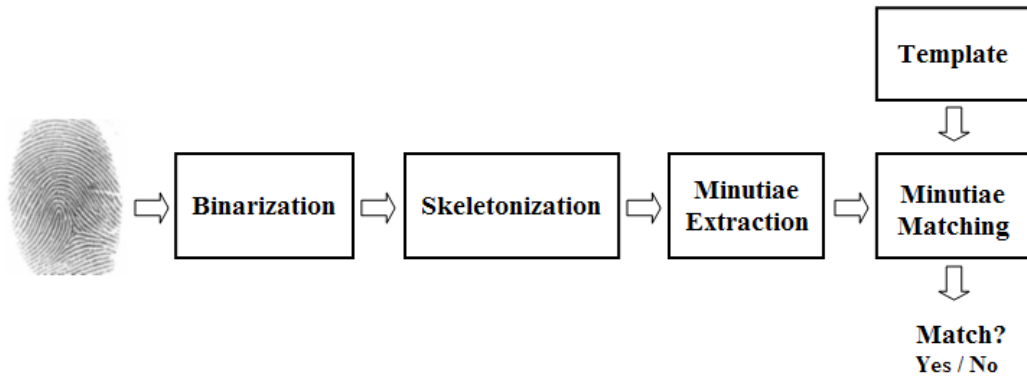


Figure 1.3: *Block-diagram of the total system design.*

Experimental results of designed system is presented and discussed in chapter 6.

Summary and conclusions with possible future work are being treated in chapter 7.

Chapter 2

Binarization

Binarization is process where a grayscale image is decimated or categorized into only two levels, black and white (0 and 1). Since quality of fingerprints is varying, the grayscale image of the fingerprint will be more or less disturbed and noisy. Therefore binarization is performed in a way that it has enhancing effects on the fingerprint image. The binarization algorithm described in this chapter is based on two scientific papers [2] and [3].

Binarization is done through filtering the fingerprint image in spatial domain with specially designed filters whose orientation matches the local ridges orientation in the fingerprint. A filter mask is designed out of user specified parameters derived from fingerprint. The whole process is done through five key steps:

1. Filter masks design out of user specified parameters.
2. Local ridge orientation and mask determination.
3. Smoothing of the orientation image.
4. Pixel-by-pixel enhancement filtering.
5. Post-processing to reduce background noise.

Detailed overview of whole binarization process and its diverse parts order and dependency on each other is shown in **Figure 2.1**.

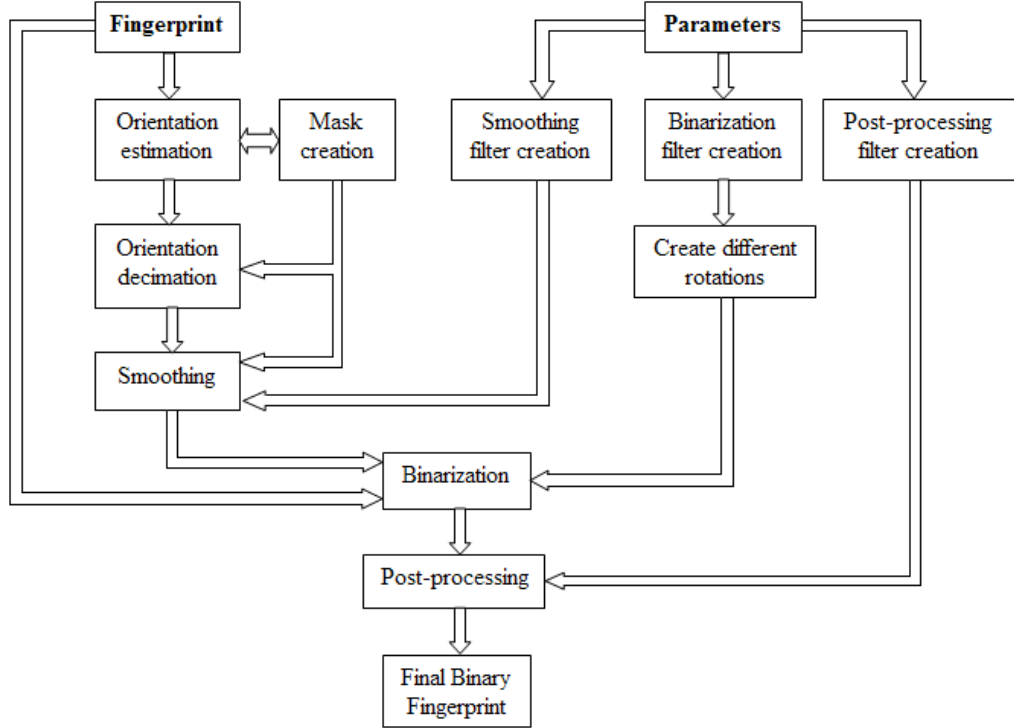


Figure 2.1: *Detailed block-schema of the binarization process.*

2.1 Filter mask design

Filter mask design is based on four input parameters from which the rest is automatically calculated so appropriate size and shape of filter is determined for optimal performance. The input parameters that needs to be specified are a_0 the peak of the filter, maximal and minimal widths of ridge R_{max} , R_{min} and also minimal width of valley V_{min} .

The filter is selected to be square with size of $k \times k$ where k is an odd number so every sample of the image is exact in the center. The size of the filter is big enough to hold at least one period of the signal which means one valley and one ridge. The filter has three sections, middle stripe with odd width that consists of coefficients that amplify ridges. The side stripe on each side of the middle stripe is chosen to negatively amplify the valleys. To reduce interference the middle and side stripes have widths equal or less than V_{min} . Depending on the widths of the middle and side stripes there may be a transition stripe in between them containing zeroes. Horizontally oriented

filter mask $f(i, j)$, where i is row index and j is column index, with center in $i = 0, j = 0$ has its stripes as following:

$$\begin{aligned} \text{middle stripe:} & \quad -h_m \leq j \leq h_m \\ \text{transition strips:} & \quad -h_t < j < -h_m; h_m < j < h_t \\ \text{side strips:} & \quad -h_f \leq j \leq -h_t; h_t \leq j \leq h_f \end{aligned} \quad (2.1)$$

where h_m , h_t and h_f are the half widths from the middle row to the end of the middle, transition and side stripes. Calculations of those are

$$h_m = \left\lfloor \frac{R_{min} - 1}{2} \right\rfloor_- \quad (2.2)$$

$$h_t = \left\lceil \frac{R_{max} - 1}{2} \right\rceil_+ \quad (2.3)$$

$$h_f = h_t - 1 + \left\lceil \frac{V_{min} - 1}{2} \right\rceil_+ \quad (2.4)$$

where $\lfloor \cdot \rfloor_-$ means, round down non-integer towards lower integer and $\lceil \cdot \rceil_+$ means, round up towards higher integer. The k can be now calculated as

$$k = 2h_f + 1. \quad (2.5)$$

Coefficients of the filter $f(i, j)$ are calculated in a way that average of those is zero so there is no dependence on average intensities in the region where the filter is applied. Filter's left and the right side and upper and lower side is made symmetric to each other. Filter shape is bell-formed to reduce holes in the ridges caused by pores and noise. The coefficients are first calculated for the middle column $f(0, j)$ where $f(0, 0)$ is set to a_0 and cosine tapered to half power $a_0/\sqrt{2}$ at $j = h_m$. The transition stripe is zero and the peak of side stripe $f(0, h_f)$ is set to b_0 and is cosine tapered to half the power of $b_0/\sqrt{2}$ at $j = h_t$. The calculations for the middle column $f(0, j)$ are

$$f(0, j) = f(0, -j) = \begin{cases} a_0 \cos \frac{j\pi}{4h_m} & 0 < j \leq h_m \\ 0 & h_m < j < h_t \\ b_0 \cos \frac{\pi(h_f-j)}{4(h_f-h_t)} & h_t \leq j \leq h_f \end{cases} \quad (2.6)$$

The value of b_0 is determined by setting the sum of all the coefficients in the column equal to zero,

$$a_0 + 2 \sum_{j=1}^{h_m} f(0, j) + 2 \sum_{j=h_t}^{h_f} f(0, j) = 0 \quad (2.7)$$

and substituted for $f(0, j)$ from equation (2.6)

$$b_0 = \frac{-a_0 \left[1 + 2 \sum_{j=1}^{h_m} \cos \frac{j\pi}{4h_m} \right]}{2 \sum_{j=h_t}^{h_f} \cos \left[\frac{\pi(h_f-j)}{4(h_f-h_t)} \right]}. \quad (2.8)$$

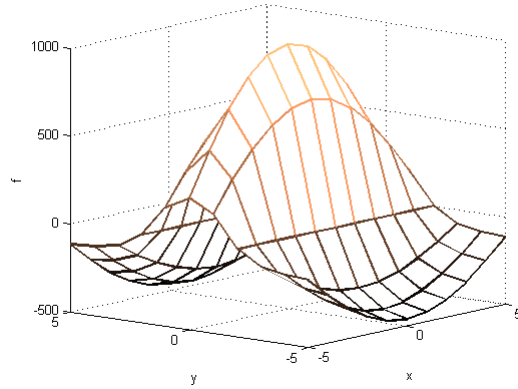
The other coefficients in the filter are then calculated by cosine taper the middle column to the side columns where they have only 1/4 of the power of the middle column. Calculation is done by following formula:

$$f(i, j) = f(-i, j) = f(i, -j) = f(-i, -j) = f(0, j) \cos \frac{i\pi}{2.383h_f}, 0 < |i| \leq h_f \quad (2.9)$$

In this thesis the values for the filter design was chosen as following $a_0 = 1000$, $R_{max} = 5$, $R_{min} = 3$ and $V_{min} = 2$ to best match the sensor with which the fingerprints has been obtained for optimal performance. The corresponding filter coefficients and 3-D image can be seen in **Figure 2.2**.

-115	-226	-323	-396	-443	-459	-443	-396	-323	-226	-115
-106	-209	-298	-366	-409	-424	-409	-366	-298	-209	-106
-81	-160	-228	-280	-313	-324	-313	-280	-228	-160	-81
0	0	0	0	0	0	0	0	0	0	0
177	349	497	611	683	707	683	611	497	349	177
250	494	703	864	965	1000	965	864	703	494	250
177	349	497	611	683	707	683	611	497	349	177
0	0	0	0	0	0	0	0	0	0	0
-81	-160	-228	-280	-313	-324	-313	-280	-228	-160	-81
-106	-209	-298	-366	-409	-424	-409	-366	-298	-209	-106
-115	-226	-323	-396	-443	-459	-443	-396	-323	-226	-115

(a) **Filter coefficients**



(b) **3-D image**

Figure 2.2: *Horizontal oriented 11×11 filter mask for: $a_0 = 1000$, $R_{max} = 5$, $R_{min} = 3$, $V_{min} = 2$*

Other filter orientations are generated by rotating the horizontal filter mask. The coefficient at the position (i', j') on the rotated mask is found by rotating with angle θ back to the position (i, j) on the original mask

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} i' \\ j' \end{bmatrix}. \quad (2.10)$$

Position (i, j) will often be a non-integer and therefore not fall exactly on the mask position. Fast and sufficient method to handle this kind of problem is to simply round off the non-integer to closest integer. Another problem is that the corners of the new position (i', j') can be mapped outside the region of the original filter mask. A simple and adequate method how to handle this problem is to extend the original mask by mirroring the mask on every side.

A diverse number of orientations have been tested to get the optimal result of binarized fingerprint image. A total of 18 different orientations of the filter have been found to give the best overall results. This number of orientations gives exactly 10° between every orientation.

2.2 Orientation and mask image

To be able to choose the proper oriented filter mask that aligns with ridge in the local area of the fingerprint, so called orientation image needs to be estimated. This image holds values of local ridge orientation for every position (x, y) in fingerprint image. Another important entity that needs to be calculated is the mask image that separates the fingerprint from background. This can be done by calculating so called orientation weights $p(\theta)$ perpendicular to three different angles $\theta = \{0^\circ, 60^\circ, 120^\circ\}$. Calculation of horizontal weight $p(0^\circ)$ for position (x, y) in the fingerprint image is given by

$$p(0^\circ) = \sum_{b=1}^k \left[\sum_{a=k-1}^1 |s(a, b) - s(a+1, b)| \right] \quad (2.11)$$

where s is a matrix with same size as filter mask, that is $k \times k$. Its mapping coordinates are $(1, 1)$ in the left upper corner and (k, k) in the right lower corner. It contains the position (x, y) along with the local area around it. The position (x, y) is placed in center of matrix s . By rotating the content in s with 60° and 120° degrees respectively, the same formula can be applied to calculate weights for $p(60^\circ)$ and $p(120^\circ)$. Rotation is done in the same way as the generation of the different orientations of the filter mask. To handle

the problem of mapping the corners outside of the s area for the 60° and 120° degree rotations, s has been widened out to size of the $2k \times 2k$. After the rotation has been performed s is resized back to size $k \times k$ holding the rotated data. The smaller value $p(\theta)$ have indicates how much the local ridge orientation diverge from that particular angle θ in that position (x, y) of the image.

When the $p(0^\circ)$, $p(60^\circ)$ and $p(120^\circ)$ are estimated, the mask image is easily determined by calculating mean value of the weights. Then by choosing a proper threshold, can the mask image be obtained. The estimation of the weights and the mean value are illustrated in **Figure 2.3**.

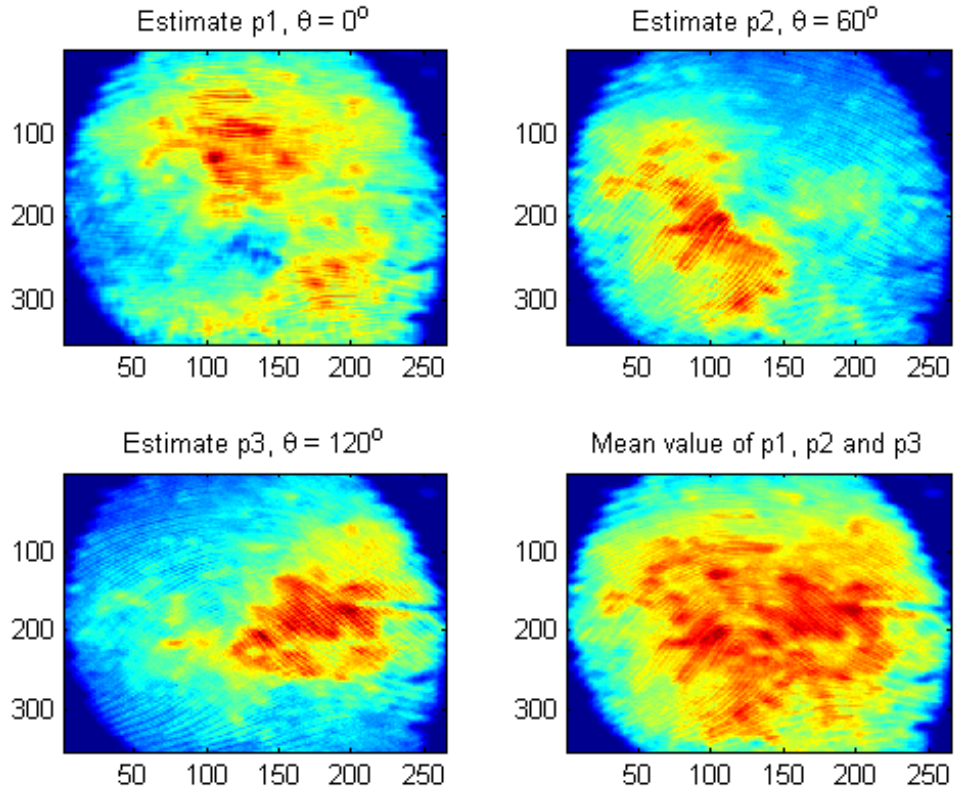


Figure 2.3: *Estimated orientation weights $p(0^\circ)$, $p(60^\circ)$, $p(120^\circ)$ and mean value.*

A threshold value that gives a good mask image is 800, everything that is under or equal to that number is belonging to the background=0 and every-

thing over it is part of the fingerprint=1. Representation of such a mask image and how good it separates fingerprint from background is demonstrated in **Figure 2.4**.

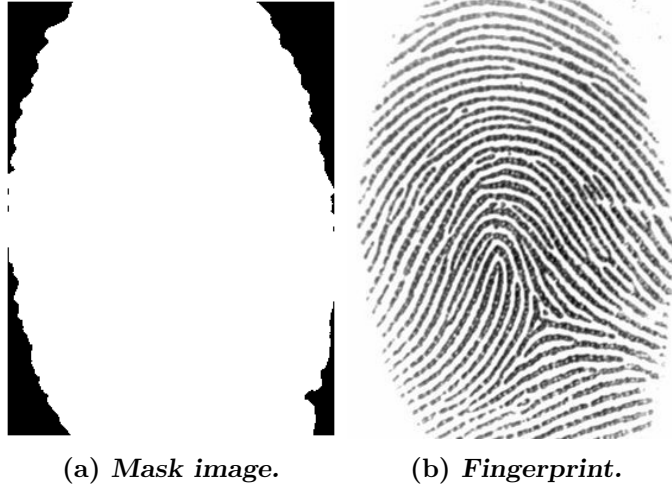


Figure 2.4: *Example of mask image and how well it separates the fingerprint from the background.*

The orientation image is found by calculating so called "circular average" from the three weights in every position as following, given the angles θ in radians:

$$\theta_{AVG} = \begin{cases} \theta_{max} + \frac{1}{2} \frac{p_{mid} - p_{min}}{p_{max} - p_{min}}, & \text{if } (\theta_{max} + \pi/3)_\pi = \theta_{mid} \\ \theta_{max} - \frac{1}{2} \frac{p_{mid} - p_{min}}{p_{max} - p_{min}}, & \text{if } (\theta_{max} - \pi/3)_\pi = \theta_{mid} \end{cases} \quad (2.12)$$

where $(\cdot)_\pi$ notation denotes the quantity in parentheses is modulo π . The weight that has the largest value is denoted as p_{max} and its angle as θ_{max} . The middle value of the weight and its angle is denoted p_{mid} and θ_{mid} . The smallest value of the weight and the angle is denoted p_{min} and θ_{min} . θ_{AVG} will be in range of 0 and π .

After all the points in the orientation image is computed, the values are recalculated back to degrees and rounded off to nearest integer. If the $\theta = 180^\circ$ it is changed to 0° since 180° and 0° degrees is in this case the same. At last the orientation image is multiplied with the mask image to get rid of the unwanted orientation estimates made in the background or the very

noisy ones on the edges of the fingerprint. Estimated orientation image is shown in **Figure 2.5**.

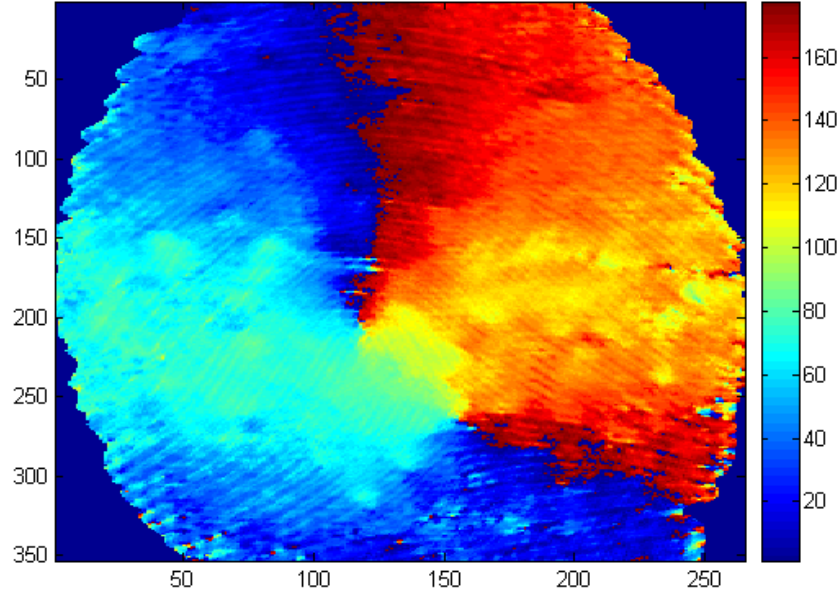


Figure 2.5: *Orientation image estimated from $p(0^\circ)$, $p(60^\circ)$ and $p(120^\circ)$ specified in degrees.*

2.3 Orientation image smoothing

Existing orientation image consists of values $0, 1, 2, \dots, 179$ representing the local ridge orientation in degrees. This orientation image can be noisy especially in the areas where the fingerprint is damaged or has relatively bad contrast between ridges and background. There is also no need for using 180 different filter mask orientations since a good binarization is obtained already when using only a few orientations.

It is desired to minimize both the amount of noise and the number of possible orientations in the orientation image. This is performed by first decimating the orientation image levels and then median filtering which takes away the extreme values in the region and therefore smoothes it out.

Decimation is performed according to the steps:

1. Choose number of orientation levels. In this case it is the same amount as the number of filter mask orientations, which is the number 18.
2.

```
if (outside of the mask)
    decimated orientation = 0
else
    index=orientation/18+1
    if (index == 19)
        decimated orientation = 1
    else
        decimated orientation = index
    end
end
```

This decimates nicely the orientation image to a new orientation image with 19 different levels. Number 0 indicates that we are outside of the mask and number 1 to 18 is an index that points to the position of the filter orientation. This new decimated orientation image is median filtered with window size of 7×7 to minimize the noise. The new orientation image is shown in **Figure 2.6**.

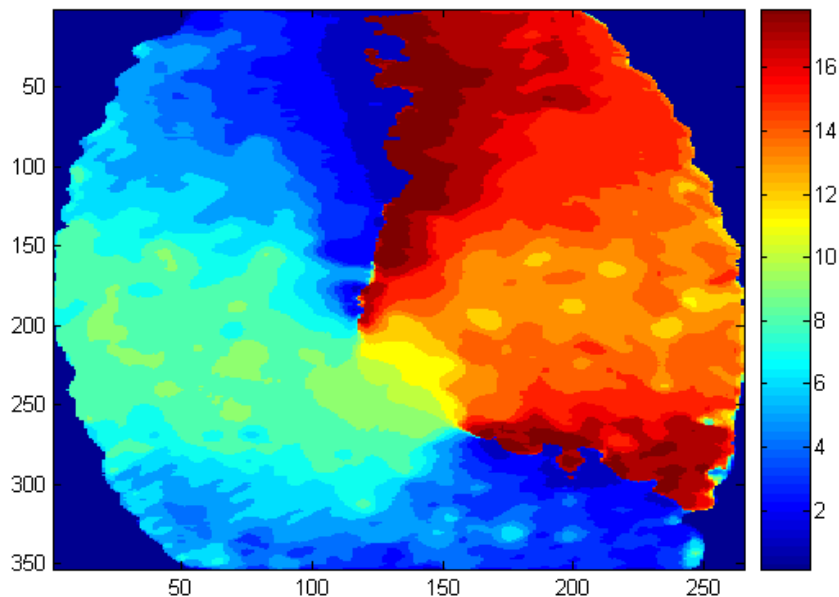


Figure 2.6: *Decimated and smoothed orientation image to only 18 different levels, given in index numbers.*

To show graphically how good the orientation image is, a mask with lines characterizing the filter mask orientations is created. The lines represent the most common orientation in the local areas with that same angle. The mask is then laid on top of the fingerprint image for comparison of how well they align with the ridges. This is exemplified in **Figure 2.7**.

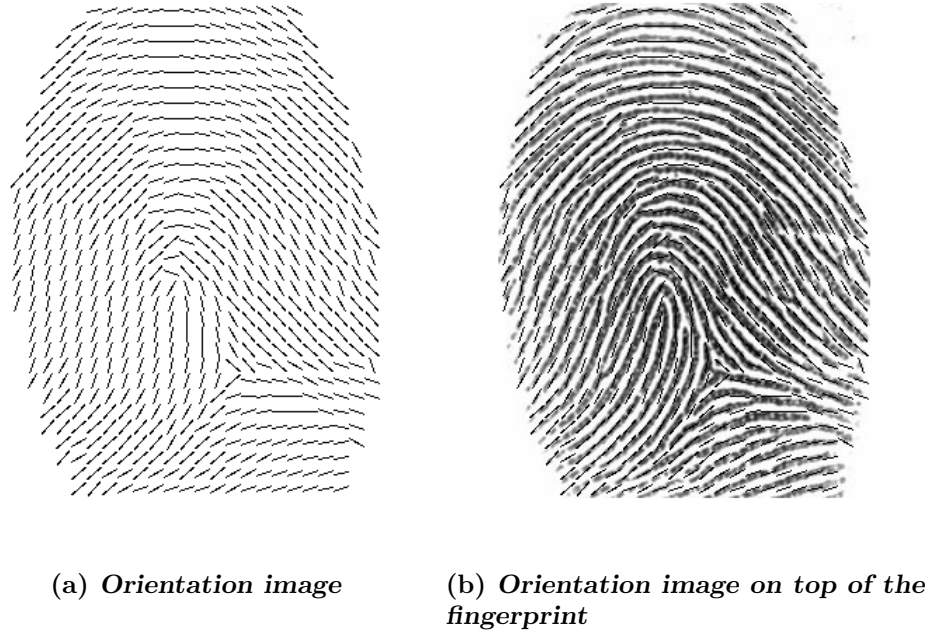


Figure 2.7: *Orientation image expressed by lines, which represents the filter orientations. A closer inspection shows how well the lines aligns with the ridges.*

2.4 Pixel-by-pixel enhancement filtering

When the filter mask orientations and orientation image is prepared the enhancement filtering is very easy to do. For every point in the fingerprint image $fp(x, y)$ we look up the local orientation at the same position in the orientation image $o(x, y)$. If the value is 0 it means that there is no local orientation and the point in the new enhanced fingerprint image $bifp(x, y)$ is set to value 1 which belongs to background. Otherwise if the value of the position in the orientation image $o(x, y)$ is non-zero it indicates that there is local orientation in that point and the filtering is then done as following:

$$bifp(x, y) = \sum_{j=-\frac{k-1}{2}}^{\frac{k-1}{2}} \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} fp(x+i, y+j)f(i, j, o(x, y)). \quad (2.13)$$

If the calculated value in point $bifp(x, y)$ is above the zero it is set to value 1 in the same position, otherwise it is set to value 0. The new enhanced and binarized fingerprint image can be found in **Figure 2.8**.



Figure 2.8: *Binarized fingerprint image.*

2.5 Post processing

There is a chance that there can be still noise in some parts of the orientation image which can cause noise in the binarized fingerprint image. Even the edges of the fingerprint can be more or less noisy because of the insufficient amount of information. This noise is being reduced through another median filtering, this time only with window size of 3×3 . Difference between binarized fingerprint image and the post processed binarized fingerprint image can be seen in **Figure 2.9**.

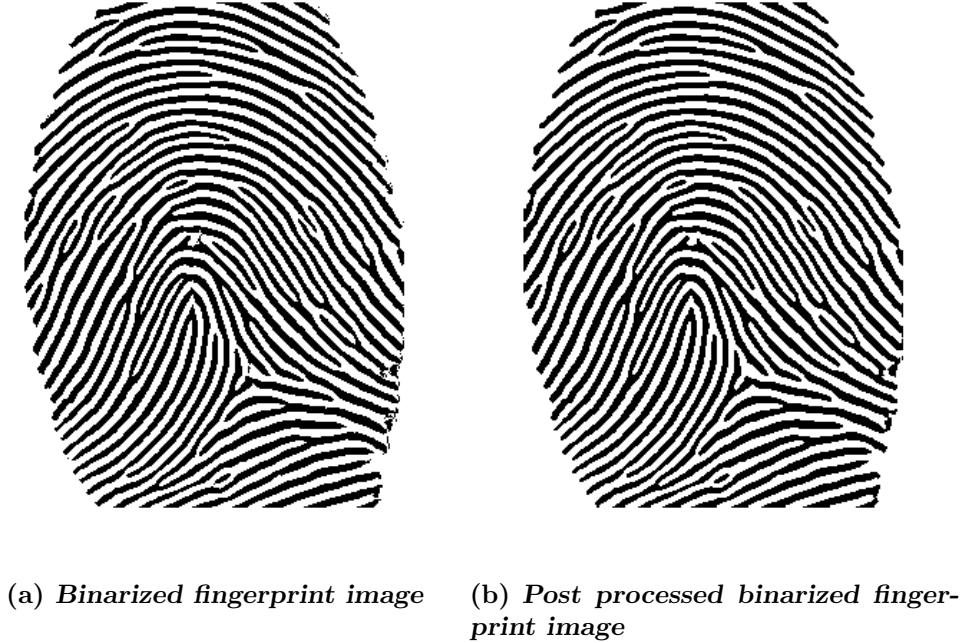


Figure 2.9: *Effect of the post processing.*

2.6 Summary

In this chapter the whole process of binarization, from filter mask design, orientation image estimation and smoothing to final enhancement filtering and post processing are presented. All grayscale fingerprint images used in this thesis and processes by this method are taken from the same database therefore originate from the same sensor. By carefully choosing the different parameters, the process is adjusted towards that certain fingerprint sensor and no further manipulation of the process is needed. In case of sensor change or applying this method on other fingerprint images originating from different sensor, new parameters needs to be tuned to get the optimal performance for those images taken with that particular sensor.

In the **Figure 2.10** is possible to see some advantages and disadvantages of this method. The binarization method is capable to filter out some small cuts and fills small gaps or holes in the ridges. The disadvantage is that some minutiae can be interchanged like termination to bifurcation and backwards. Also some small details in ridges can disappear.



Figure 2.10: *Advantages and disadvantages of the binarization process. Rings in the two images indicate most unwanted effects and ellipse highlight most positive effect.*

Chapter 3

Skeletonization

Skeletonization is a process mostly used on binarized images by thinning a certain pattern shape until it is represented by 1-pixel wide lines, the so called skeleton of that pattern. Since minutiae are determined only by discontinuities in ridges, they are totally independent of ridges thickness. By finding the skeleton of the binarized fingerprint image through thinning of the ridges to only one pixel wide lines, the minutiae are preserved with minimum possible data. This decimation of data offers more effective minutiae extraction realization. Skeletonization algorithm described in this chapter is based on a scientific paper [4].

3.1 Overview

Skeletonization is performed separately on every pixel P_i belonging to the ridges by examining the nearest neighbors around it. Take a look at the **Figure 3.1**. A couple of rules are set up to test if the pixel P_i can be taken away from the ridge without affecting the flow and connectivity of the lines and still preserv the endings of ridges.

$P1$	$P2$	$P3$
$P8$	P_i	$P4$
$P7$	$P6$	$P5$

Figure 3.1: 3×3 pixel mask of the nearest neighbors mapped around the P_i .

The following set of rules 1 have been set up to test if the pixel P_i can be taken away:

1. $2 \leq B(P_i) \leq 6$.

2. $A(Pi) = 1$.
3. $P2 * P4 * P6 = 0$.
4. $P4 * P6 * P8 = 0$.

The $B(Pi)$ is the number of neighbors with value 1 and is calculated as:

$$B(Pi) = P1 + P2 + \dots + P8 \quad (3.1)$$

and the $A(Pi)$ is the number of $0 \rightarrow 1$ patterns in the neighborhood and is calculated as:

$$A(Pi) = (P1 | P2) \& (\sim P1) + (P2 | P3) \& (\sim P2) + \dots + (P8 | P1) \& (\sim P8) \quad (3.2)$$

examples are showed in **Figure 3.2**.

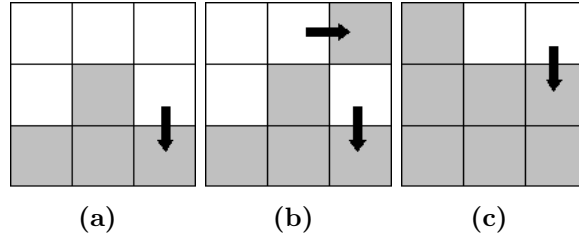


Figure 3.2: *Examples of $0 \rightarrow 1$ patterns in the neighborhood.*

Unfortunately the above given rules can't handle the slanting lines 2-pixel wide, those lines are erased. To handle that problem another set of rules has been set up to neutralize that effect.

The new set of rules 2 are:

1. $3 \leq B(Pi) \leq 6$.
2. $A(Pi) = 1$.
3. $P2 * P4 * P8 = 0$.
4. $P2 * P6 * P8 = 0$.

The new rules prevent erasing of the 2-pixel wide lines, but it can't produce 1-pixel wide lines. Therefore when the skeletonization according to above rules is done, another skeletonization is performed to handle the remaining 2-pixel wide lines and the lines are thinned to 1-pixel wide lines.

The *set of rules 3* for the second skeletonization are as following:

1. $P1 * P8 * P6 = 1 \ \& \ P3 = 0$.
2. $P3 * P4 * P6 = 1 \ \& \ P1 = 0$.
3. $P5 * P6 * P8 = 1 \ \& \ P3 = 0$.
4. $P4 * P6 * P7 = 1 \ \& \ P1 = 0$.

Patterns for which the pixels P_i are removed in the second skeletonization can be seen in **Figure 3.3**.

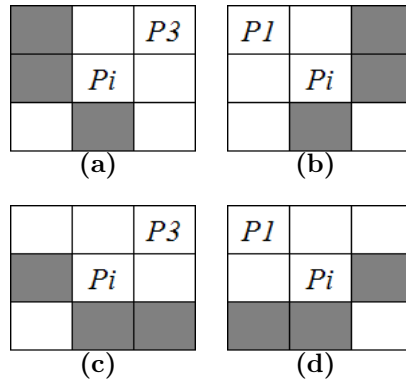


Figure 3.3: *Patterns there P_i is removed in the second skeletonization.*

3.2 Realization

Skeletonization is performed on so called "negative image" of the binarized fingerprint images since above specified rules uses 1 that represents the ridges and 0 for representing the background. The binarized fingerprint images uses exact the opposite signs. A negative image is simply formed by performing a logical NOT operation on the binarized fingerprint.

The examination of the pixels is done in iterations where the first two rule sets are applied in turns. The pixels that can be erased are marked and first at end of each iteration are removed from the image. This process is repeated until there are no more pixels that can be removed from the image. Then the second skeletonization process is started to remove the remaining pixels to produce the 1-pixel wide lines. This process takes only one iteration. After that the image is converted back and the skeleton of the binarized fingerprint image is found. The flow diagram is shown in **Figure 3.4**.

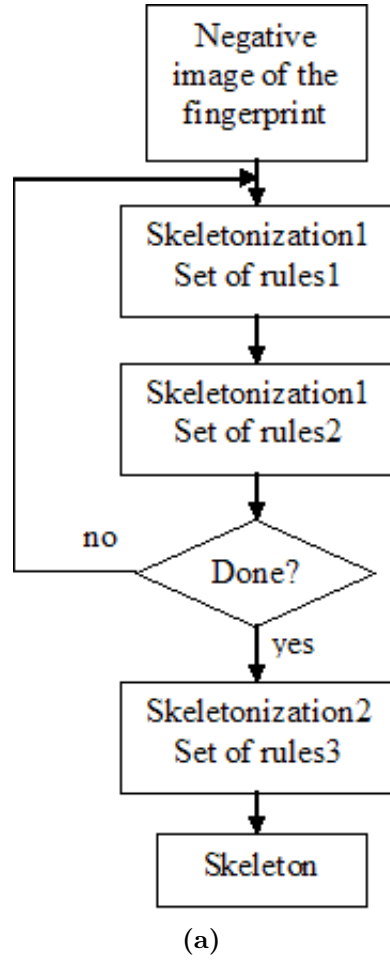


Figure 3.4: *Flow diagram of the skeletonization process.*

Following **Figure 3.5** displays how the different stages between each iteration look likes. From the binary fingerprint image to the final skeleton.

The illustration that the skeletonization process found the skeleton of the binarized fingerprint in the center of the ridges can be seen in **FigureX 3.6**. The fingerprint and skeleton is laid on top of each other and the intersection of them both is highlighted with red color.

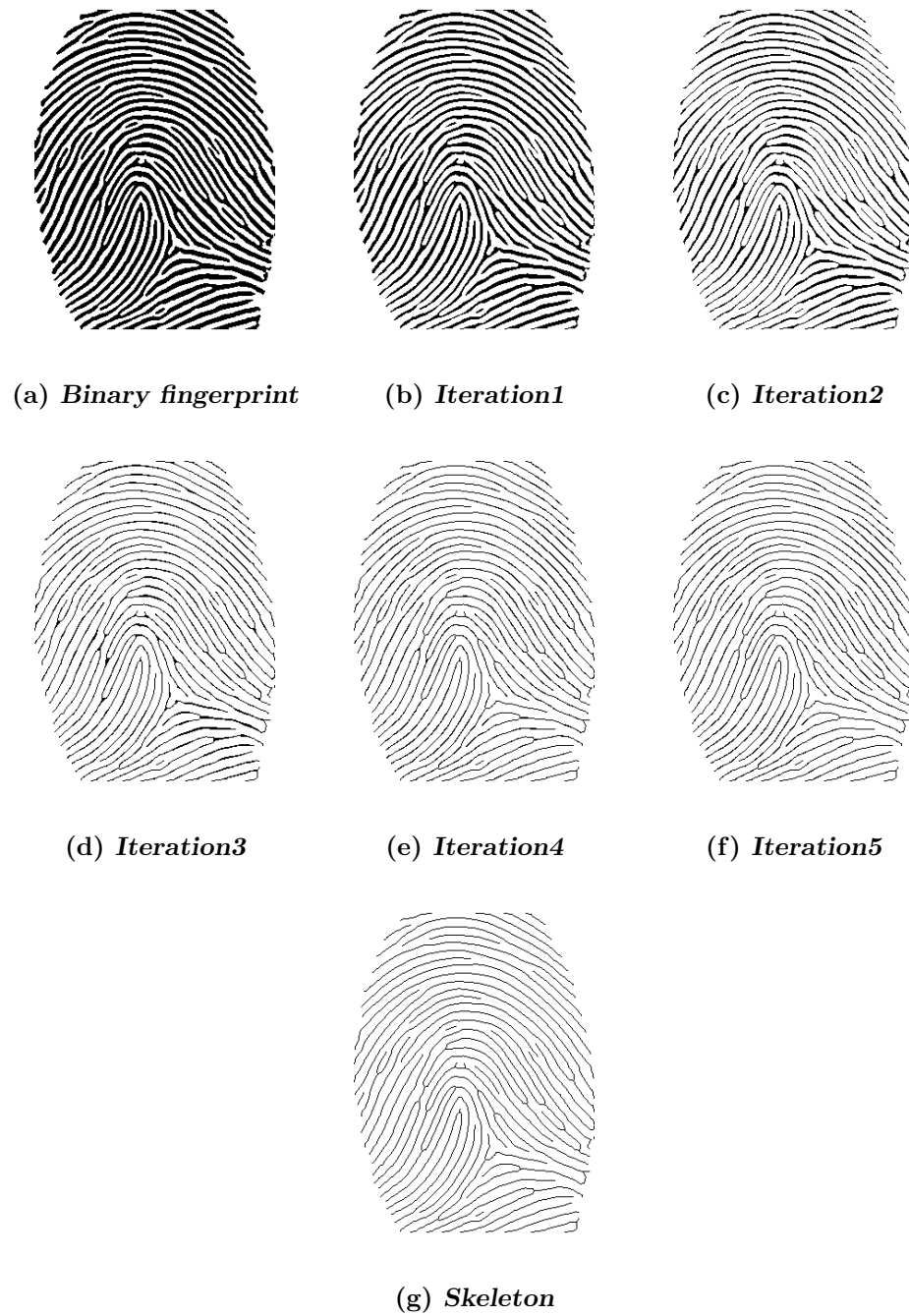


Figure 3.5: *The steps between each iteration in the skeletonization process.*



Figure 3.6: *Intersection between the binarized fingerprint and its skeleton.*

Chapter 4

Minutiae Extraction

Extracting minutiae from the skeleton of the fingerprint requires a method that is able to distinguish and categorize the different shapes and types of minutiae. This is a classification problem and can be solved by constructing and training a neural network which work as a classifier. Training of the neural network is conducted with the back-propagation algorithm. The method developed in this chapter is based on literature [5] and [6].

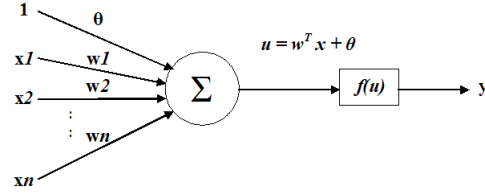
4.1 Neural network

Neural network is a nonlinear mapping system whose structure is loosely based on principles of the real brain. The whole network is build up with simple processing units, structures of those can be seen in **Figure 4.1 (a)**. The unit is a simplified model of a real neuron. Its parts are the input vector $\bar{\mathbf{x}}$ whose containing information is manipulated by the weighted nodes by weigh vector $\bar{\mathbf{w}}$. The node with weight θ and input 1 is a so called bias of the neuron which gives the freedom to the neuron to be able to shift the function $f(\cdot)$. Function $f(\cdot)$ is usually a nonlinear function and is mostly called activation function. Input to this function $f(\cdot)$ is a sum of all the nodes and is denoted as u .

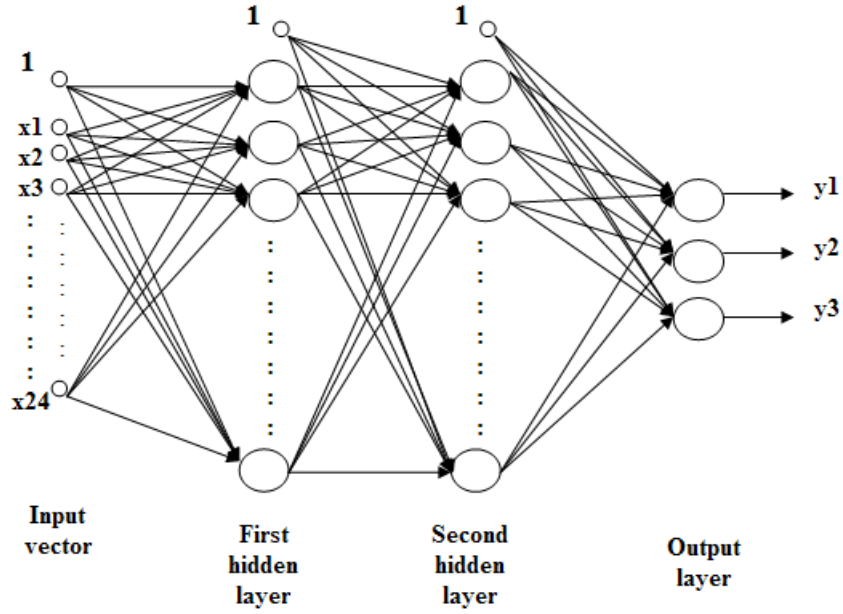
There are many different types of the neural networks, in this thesis there are used only so called *feedforward neural networks*. The neurons are structured in layers and connections are drawn only from the previous layer to the next one. A typical structure of this type of the neural network can be seen in **Figure 4.1 (b)**. Calculating the output vector $\bar{\mathbf{y}}$ is done by the formula

$$\bar{\mathbf{y}} = F^{[3]} \left[\bar{\mathbf{W}}^{[3]} \cdot F^{[2]} \left[\bar{\mathbf{W}}^{[2]} \cdot F^{[1]} \left[\bar{\mathbf{W}}^{[1]} \cdot \bar{\mathbf{x}} \right] \right] \right] \quad (4.1)$$

where $\overline{\mathbf{W}}$ is the networks weight matrix, F is the nonlinear activation function calculating on the vector and not only on a one value as $f(\cdot)$. The $[\cdot]$ indicates in which layer the calculations are done. The bias 1 with weight θ is a part of the input vector $\overline{\mathbf{x}}$ and weight matrix $\overline{\mathbf{W}}$ in the above formula.



(a) *The unit structure*



(b) *Feedforward neural network*

Figure 4.1: *Simplified model of the neuron and the structure of the neural network.*

4.2 Back-propagation algorithm

What is making the neural networks interesting is that the determination of its weights \mathbf{w} is done by learning. A back-propagation is one of many different learning algorithms that can be applied for neural network training

and has been used in this thesis. It belongs to a category of so called learning with the teacher. For every input vector $\bar{\mathbf{x}}$ that is presented to the neural network there is predefined desired response of the network in a vector $\bar{\mathbf{t}}$ (the teacher). The desired output of the neural network is then compared with the real output by computing an error $\bar{\mathbf{e}}$ of vector $\bar{\mathbf{t}}$ and neural network output vector $\bar{\mathbf{y}}$. The correction of the weights in the neural network is done by propagating the error $\bar{\mathbf{e}}$ backward from the output layer towards the input layer, therefore the name of the algorithm. The weight change in each layer is done with steepest descent algorithm. The back-propagation algorithm is carried out in the following steps:

Step1: Initialization values of the weight matrices $\bar{\mathbf{W}}^{[1]}$, $\bar{\mathbf{W}}^{[2]}$ and $\bar{\mathbf{W}}^{[3]}$ should be rather small in the range of $[-0.5, 0.5]$ and random. The following formulas have been used

$$\bar{\mathbf{W}}^{[1]} = \frac{(rand(m, n + 1) - 0.5)}{n + 1} \quad (4.2)$$

$$\bar{\mathbf{W}}^{[2]} = \frac{(rand(k, m + 1) - 0.5)}{m + 1} \quad (4.3)$$

$$\bar{\mathbf{W}}^{[3]} = \frac{(rand(p, k + 1) - 0.5)}{k + 1} \quad (4.4)$$

where rand has uniform distribution in $[0, 1]$.

Step2: Randomly select the $(\bar{\mathbf{x}}_i, \bar{\mathbf{t}}_i)$ pair from the training data and compute the output $\bar{\mathbf{y}}_i$.

$$\bar{\mathbf{y}}_i = F^{[3]} \left[\bar{\mathbf{W}}^{[3]} \cdot F^{[2]} \left[\bar{\mathbf{W}}^{[2]} \cdot F^{[1]} \left[\bar{\mathbf{W}}^{[1]} \cdot \bar{\mathbf{x}}_i \right] \right] \right] \quad (4.5)$$

Step3: Find the weight correction $\Delta \bar{\mathbf{W}}$ for each layer.

First calculate the error $\bar{\mathbf{e}}_i$ according to

$$\bar{\mathbf{e}}_i = \bar{\mathbf{t}}_i - \bar{\mathbf{y}}_i \quad (4.6)$$

then the local error vector $\delta^{[3]}$ is computed

$$\delta^{[3]} = diag(\bar{\mathbf{e}}_i) \frac{\partial F^{[3]}}{\partial \mathbf{u}^{[3]}} \quad (4.7)$$

and the computation of the correction of $\Delta \bar{\mathbf{W}}^{[3]}$ is

$$\Delta \bar{\mathbf{W}}^{[3]} = \eta \delta^{[3]} (\bar{\mathbf{o}}^{[2]})^T \quad (4.8)$$

where the η is the step size and $\bar{\mathbf{o}}^{[2]}$ is the output vector from the second layer in the neural network. For the rest of the layers of the neural network the calculations

$$\delta^{[2]} = \text{diag}(\bar{\mathbf{W}}^{[3]} \delta^{[3]}) \frac{\partial F^{[2]}}{\partial \mathbf{u}^{[2]}} \quad (4.9)$$

$$\Delta \bar{\mathbf{W}}^{[2]} = \eta \delta^{[2]} (\bar{\mathbf{o}}^{[1]})^T \quad (4.10)$$

$$\delta^{[1]} = \text{diag}(\bar{\mathbf{W}}^{[2]} \delta^{[2]}) \frac{\partial F^{[1]}}{\partial \mathbf{u}^{[1]}} \quad (4.11)$$

$$\Delta \bar{\mathbf{W}}^{[1]} = \eta \delta^{[1]} (\bar{\mathbf{x}})^T \quad (4.12)$$

are performed. **OBS!** The $\bar{\mathbf{o}}^{[2]}$ and $\bar{\mathbf{o}}^{[1]}$ are the extended vectors and $\bar{\mathbf{W}}^{[3]}$ and $\bar{\mathbf{W}}^{[2]}$ are without the bias.

Step4: Update the weight matrices

$$\bar{\mathbf{W}}^{[3]} = \bar{\mathbf{W}}^{[3]} + \Delta \bar{\mathbf{W}}^{[3]} \quad (4.13)$$

$$\bar{\mathbf{W}}^{[2]} = \bar{\mathbf{W}}^{[2]} + \Delta \bar{\mathbf{W}}^{[2]} \quad (4.14)$$

$$\bar{\mathbf{W}}^{[1]} = \bar{\mathbf{W}}^{[1]} + \Delta \bar{\mathbf{W}}^{[1]} \quad (4.15)$$

Return to the **step2**, stop when the conditions for aborting the training are fulfilled.

The above described algorithm is using so called on-line learning which means that the weights are updated after each pattern presented to the network. The patterns are selected randomly so the neural net doesn't learn the order of the patterns in the training data. To get a good realization of the neural network the learning step η is made adaptive. The adaptation is done by so called *schedule annealing* or also called *search then converge*. The formula is given as following:

$$\eta(\text{epoch}) = \frac{\eta_0}{1 + \text{epoch}/\tau}. \quad (4.16)$$

Epoch or batch as it is also called is when all patterns in the training data have been presented to the neural network. After every epoch the learning

step is decreased by above stated formula so it can settle down in a good minima. τ is a constant and is chosen relatively large. When the training is in the beginning the *epoch* number is small. In this case it is as if the learning step η is constant. As the *epoch* number increases with the time of the training the learning step η approaches the asymptotic behavior. For this thesis the τ was chosen to value 5000 and the η_0 to 0.01.

A bipolar sigmoid function has been chosen as the activation function. It is named after its *S* shaped form and because it can have both positive and negative values. The formula for the bipolar sigmoidal function is

$$f(u) = \alpha \tanh(\beta u) \quad (4.17)$$

The constants α and β are chosen as:

$$\alpha = 1.7159 \quad (4.18)$$

and

$$\beta = \frac{2}{3} \quad (4.19)$$

which gives the function $f(u)$ good characteristics as $f(-1) = -1, f(1) = 1$ and linear transition with slope near 1 in between. Graf of such a function can be seen in **Figure 4.2**.

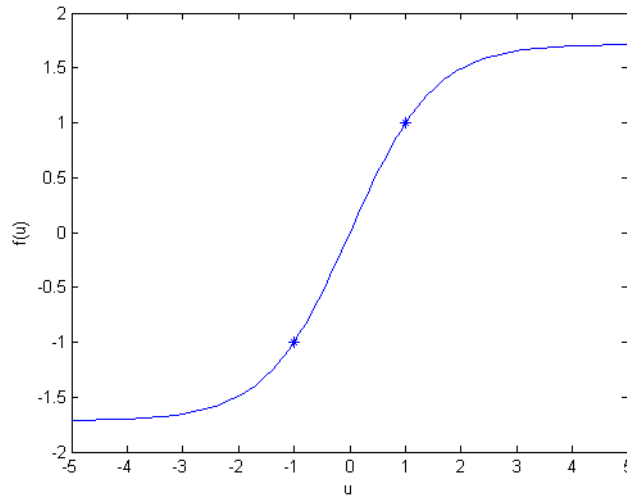


Figure 4.2: *The bipolar sigmoid activation function. The * marks $f(-1) = -1$ and $f(1) = 1$.*

The derivative of the $f(u)$ function is

$$\frac{df}{du} = (\alpha\beta)(1 - \tanh^2(\beta u)) \quad (4.20)$$

and is used in $F(\cdot)$ as

$$\frac{\partial F}{\partial \mathbf{u}} = \left[\frac{df}{du_1}; \frac{df}{du_2}; \dots; \frac{df}{du_N} \right]^T. \quad (4.21)$$

4.3 Classifier realization

Training data is divided into three different pattern classes; *termination*, *bifurcation* and *no minutiae*. The neural network output layer consists of three neurons, each representing one class of the training data. By training the neural network so it activates the right neuron corresponding to the pattern class, the classification of the input patterns can be achieved.

Size of the training data is chosen to 5×5 windows. The 3×3 window doesn't view much information and the 7×7 window show too much information which is unnecessary. Example on how the different sizes of the window results on the training patterns are shown in **Figure 4.3**. The size of the window is deliberately odd so there is exactly one pixel in the center.

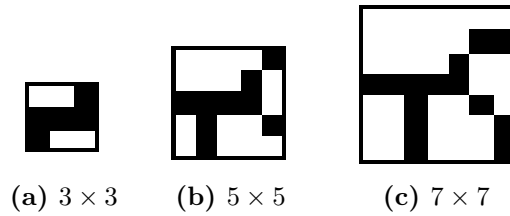


Figure 4.3: *Different window sizes for the training data.*

A total of 23 different fingerprint skeletons have been used to collect the necessary patterns for the three classes. A total of 1951 different patterns have been gathered. The different classes had 84 termination patterns, 388 bifurcation patterns and 1479 no minutiae patterns. Selection of the patterns was carefully done so the different minutiae types are found in the center. Even patterns with minutiae slightly off center are classified as no minutiae to avoid false classification towards that particular minutia type nearby. Examples of some patterns for each class can be seen in **Figure 4.4**.

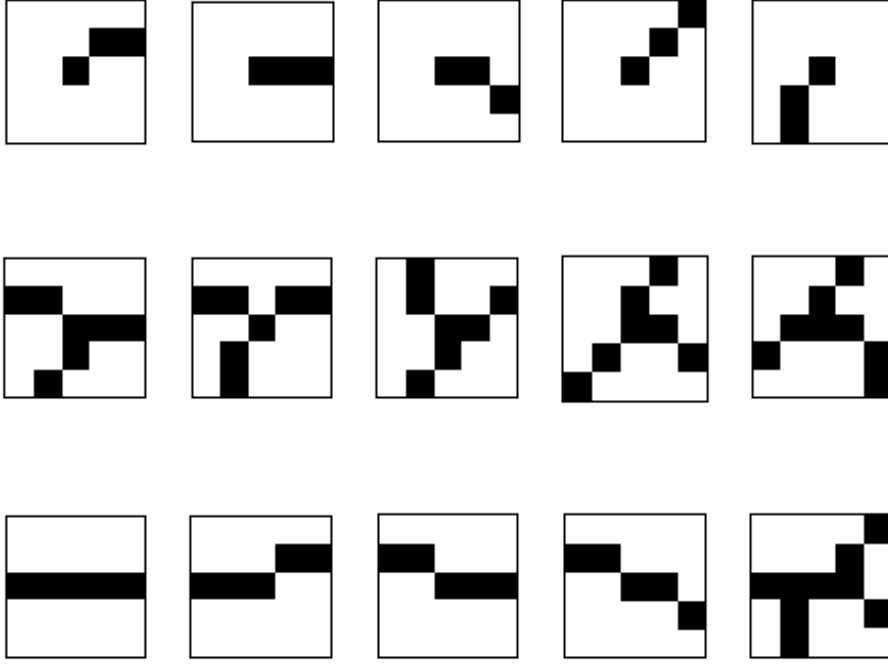


Figure 4.4: *Examples of the training data. Upper row: termination patterns, middle row: bifurcation patterns and lower row: no minutiae patterns.*

To find an optimal size and shape of the neural network for good realization, training test on various nets was conducted. To measure how fast the different neural networks learns a total error in each epoch has been calculate. The convergence of this error reveals how fast and good each neural network is. The error function that has been used is

$$\mathbf{E}_{\text{SSE}} = \frac{1}{2P} \sum_{i=1}^P \bar{\mathbf{e}}_i^T \bar{\mathbf{e}}_i \quad (4.22)$$

where $\bar{\mathbf{e}}_i$ is the error vector of the particular patter i in the training data set and P is total number of patterns in the training data set. This function is called *sum of squared errors* (SSE) and is normalized with size of the training set.

Three neural networks were tested, two with single hidden layer and one with two hidden layers. The single hidden layers had 50 and 60 neurons respectively and two hidden layers had 25 neurons each. All neural networks was trained for 5000 epochs and then compared how fast and low the error

falls. A graph with the error function calculated for each neural network is shown in **Figure 4.5**. The "hl1 50" denotes one hidden layer with 50 neurons, "hl1 60" denotes one hidden layer with 60 neurons and "hl2 2525" denotes neural network with two hidden layers with 25 neuron in each layer. It can be clearly seen that the neural network with two hidden layers converges fastest and lowest. The two single hidden layered are about the same. The interesting thing is to see how much improvement the extra hidden layer does.

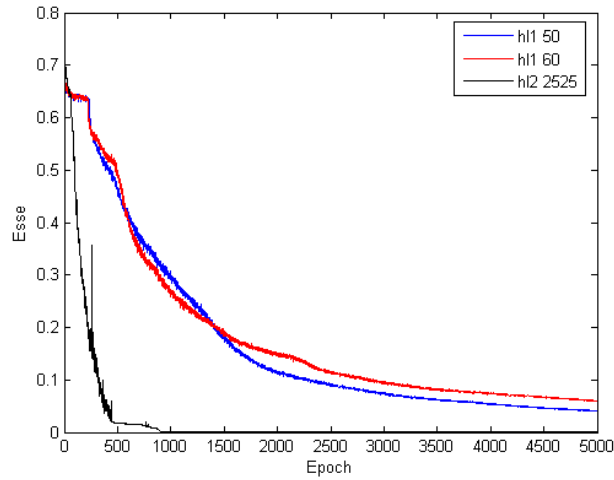


Figure 4.5: The "sum of squared errors" (SSE) error functions normalized with size of training data for three different neural networks. The "hl1 50" denotes one hidden layer with 50 neurons, "hl1 60" denotes one hidden layer with 60 neurons and "hl2 2525" denotes neural network with two hidden layers with 25 neuron in each layer.

The problem with two hidden layered neural network is that it can be easily over-trained; therefore training should not be longer then necessary. A so called *MAXNET* has been added on the output of the neural network. What *MAXNET* does is that it compares the three outputs from the neural network. The output from the neuron that has the biggest value is taken as the activated neuron. So the input pattern to the neural network is classified to the class that is represented by activated neuron in the output layer. The error function of the *MAXNET* under the training of the neural network indicates how many patterns are left to be learned. When *MAXNET* error is 0 all patterns have been learned by the neural network, the training can now be stopped to avoid possible over-training. An error function of the *MAXNET* can be seen in **Figure 4.6** for the two hidden layered neural

network. The trained neural network has 25 neurons in each hidden layer. It took 610 epochs to learn all the patterns in the training set.

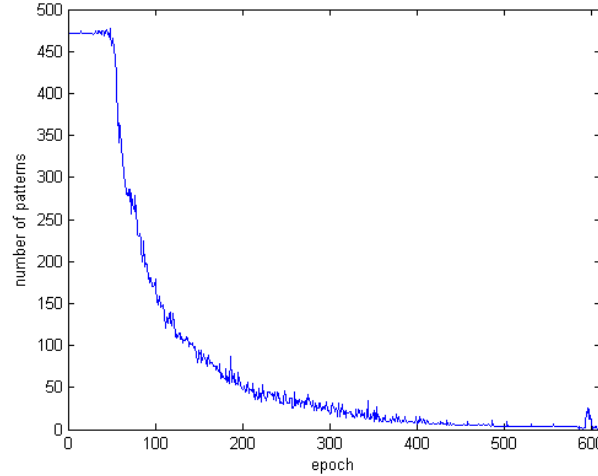


Figure 4.6: *MAXNET* error function indicates how many patterns are left to be learned by the neural network. Error is 0 after 610 epochs.

Finally the extraction of the minutiae from the fingerprint's skeleton is done by sliding the 5×5 window though the image. Each portion is then presented to the trained neural network which classifies it into one of the three classes. If the data sent to the neural network is a minutia coordinates (x, y) of the pixel in the picture are stored in a vector. The coordinates (x, y) represents the middle pixel in the window corresponding to the global coordinates in the picture.

To speed up and make things easier in the extraction of the minutiae, an assumption is made. The portions of data that are viewed by the neural network are only those who have a black pixel in the middle. Since the minutiae are made only from the thinned lines there is no need to examine data where lines are off center in the data portion. This assumption gives three very good improvements.

1. A skeleton image has only about 30% of the black pixels. This means that 70% of the data portions are not examined by the neural network. This gives very fast and accurate minutiae extractions since the window can be slide pixel-by-pixel through the image.

2. Since lines are centered in the data portions the patterns where lines are off centered does not need to be included in the training set. The training set is much smaller than it needs to be.
3. Because the middle pixel in the data portion is always black it does not give any new information to the neural network. Therefore the middle pixel can be discarded from the input to the neural network and the input vector is made smaller.

In this thesis a neural network with following characteristics was used for minutiae extraction. Input is made by stapling columns in the data portion into one column vector. Since the middle pixel is discarded the column has 24 bit plus a 1 bit as a bias. The neural network has two hidden layers with 25 neurons in each. Output layer consists of 3 neurons, each representing one class of the patterns. The shape of this neural network is actually the same as the neural network in the **Figure 4.1 (b)** with an extent by the *MAXNET*. Example of how well the neural network works as a classifier is shown in the next **Figure 4.7**. Notice the accuracy of the finding the minutiae in the center in the zoomed section of the image in the **Figure 4.7 (b)**.

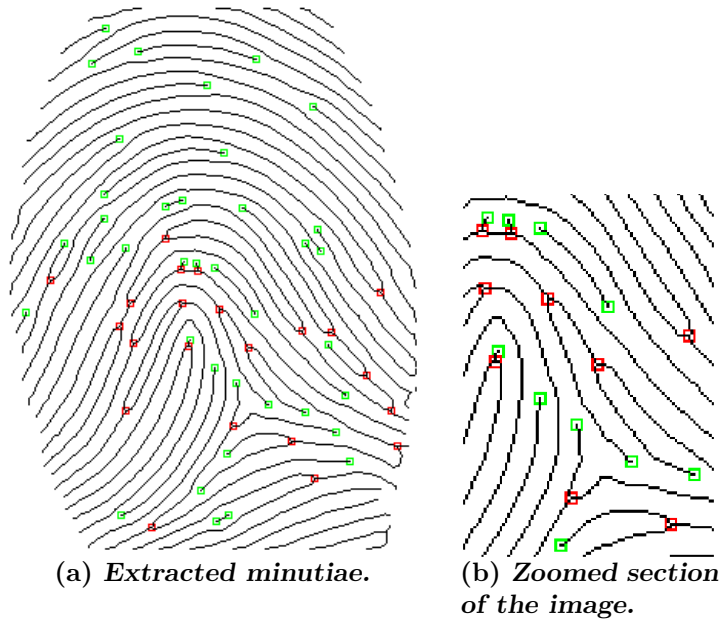


Figure 4.7: *Result of how well the neural network works as a classifier. Green boxes are the marked terminations. Red boxes are the marked bifurcations. Accuracy of the neural network can be seen in the figure b.*

Chapter 5

Minutiae Matching

Extracted minutiae from the fingerprint are together forming a point pattern in plane. Therefore matching two minutiae point patterns with each other are considered as a 2D point pattern problem. An algorithm is needed that localize the maximum number of mutual points in the two point patterns. The algorithm described in this chapter is based on literature [1] and a scientific paper [7].

The point patterns are constructed only on positions (x, y) of minutiae in the plane. The minutiae type and orientation which provides extra information are disregarded due to possible type alternation and noise in orientation. The alternation can be caused by varying pressure between fingertip and the sensor and also by binarization process. Low pressure can cause that bifurcation minutiae appear as termination minutiae in the fingerprint image. On other hand high pressure can cause termination minutiae appear as bifurcation minutiae in the fingerprint image. Alternations of minutiae types by the binarization process has been pointed out and described in chapter 2. Bad quality of fingerprint image gives noisy orientation image and therefore false minutiae orientation. Alternation and false orientation of the minutiae gives higher risk that not all mutual pointes are detected. Since point patterns are based on positions of minutiae in fingerprint they form distinctive patterns. With enough points in each pattern the positions (x, y) of the minutiae are the only information that is needed for good matching results. By using only (x, y) coordinates of minutiae yields that less memory is needed for implementation of this algorithm.

The matching is performed on the two point patterns P with m number of pointes $\{p_1, p_2, \dots, p_m\}$ and Q with n number of points $\{q_1, q_2, \dots, q_n\}$. The algorithm is made invariant towards possible translation and rotation

among P and Q so the maximum number of matching points is found. The matching is performed in two essential phases. First, a so called principal pair $p_{pair_i} \leftrightarrow q_{pair_a}$ is identified. Second, the matching pairs $p_i \leftrightarrow q_a$ are found.

5.1 Determination of the principal pair

The principal pair is the two points $p_{pair_i} \leftrightarrow q_{pair_a}$ corresponding to each other in the P and Q . It is the pair of points under which the maximum number of pairs $p_i \leftrightarrow q_a$ is found. Identification of $p_{pair_i} \leftrightarrow q_{pair_a}$ is done by examining the scale s and rotation θ differences of the vectors. The values are calculated from points p_i, p_j, q_a and q_b where $i \neq j$ and $a \neq b$.

$$|\overrightarrow{p_i p_j}| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad (5.1)$$

$$\theta_{\overrightarrow{p_i p_j}} = \begin{cases} \arctan \frac{y_j - y_i}{x_j - x_i}, & \text{for } (x_j - x_i) > 0 \\ \arctan \frac{y_j - y_i}{x_j - x_i} + \pi, & \text{for } (x_j - x_i) < 0 \\ \frac{\pi}{2}, & \text{for } (x_j - x_i) = 0, y_j - y_i > 0 \\ -\frac{\pi}{2}, & \text{for } (x_j - x_i) = 0, y_j - y_i < 0 \end{cases} \quad (5.2)$$

$$|\overrightarrow{q_a q_b}| = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \quad (5.3)$$

$$\theta_{\overrightarrow{q_a q_b}} = \begin{cases} \arctan \frac{y_b - y_a}{x_b - x_a}, & \text{for } (x_b - x_a) > 0 \\ \arctan \frac{y_b - y_a}{x_b - x_a} + \pi, & \text{for } (x_b - x_a) < 0 \\ \frac{\pi}{2}, & \text{for } (x_b - x_a) = 0, y_b - y_a > 0 \\ -\frac{\pi}{2}, & \text{for } (x_b - x_a) = 0, y_b - y_a < 0 \end{cases} \quad (5.4)$$

$$s = \frac{|\overrightarrow{q_a q_b}|}{|\overrightarrow{p_i p_j}|} \quad (5.5)$$

$$\theta = \theta_{\overrightarrow{q_a q_b}} - \theta_{\overrightarrow{p_i p_j}}. \quad (5.6)$$

The search for principal pair $p_{pair_i} \leftrightarrow q_{pair_a}$ is conducted by testing each point p_i toward all points q_a . For every pair $p_i \leftrightarrow q_a$ the so called *Matching Pairs Support* (MPS) is calculated. The MPS value w_{ia} is the number of most common θ between the vectors with $S_{MIN} \leq s \leq S_{MAX}$ where $S_{MIN} = 0.98$ and $S_{MAX} = 1.02$. The value s is chosen around 1 because if the point patterns P and Q are originating from the same person and sensor the scale

should be 1. Due to plasticity of the skin the points can shift the position to some extent and introduce noise to the coordinates. Therefore some variation in s is needed to be taken in a consideration. After each calculation towards pair $p_j \leftrightarrow q_b$, the cumulative sum $M(\theta)$ is updated. The $M(\theta)$ is array where θ denotes the position in the array that is increased with 1. To make things easier the θ is converted from radians to degrees and quantized to 0.25 precision and is denoted θ^+ . The original θ is in the interval $[-\pi, \pi]$, the converted and quantized θ^+ is in the interval $[0, 359.75]$ degrees with steps of 0.25 in-between. When all pairs $p_j \leftrightarrow q_b$ has been exploited, the accumulator sum is searched for the peak. The θ^+ that has the biggest peak in the $M(\theta^+)$ is the corresponding rotation between P and Q for the $p_i \leftrightarrow q_a$ pair. The following pseudo code shows how the above described MPS works.

```

Set the accumulator sum  $M(\theta^+) = 0$  for all  $\theta^+$ 
 $S_{MIN} = 0.98$ 
 $S_{MAX} = 1.02$ 
for  $j = 1, \dots, m, i \neq j$ 
    for  $b = 1, \dots, n, a \neq b$ 
         $s = \frac{|\vec{q_a q_b}|}{|\vec{p_i p_j}|}$ 
        if  $S_{MIN} \leq s \leq S_{MAX}$ 
             $\theta = \theta_{\vec{q_a q_b}} - \theta_{\vec{p_i p_j}}$ 
             $\theta^+ = \text{quantize}(\theta \frac{180}{\pi})$ 
             $M(\theta^+) = M(\theta^+) + 1$ 
        end
    end
end
Search the  $M(\theta^+)$  for which  $\theta^+$  it has the biggest peak. Return the
biggest value labeled as  $w_{ia}$  together with the according  $\theta^+$ .

```

To locate the best suitable pair $p_i \leftrightarrow q_a$ as principal pair the MPS value w_{ia} is compared to its previous value. If the new w_{ia} is bigger, then the new pair $p_i \leftrightarrow q_a$ is chosen as a new principal pair. The following pseudo code shows how the most suitable $p_i \leftrightarrow q_a$ is selected.

```

Set  $pmax = 0$ 
for  $i = 1, \dots, m$ 
    for  $a = 1, \dots, n$ 
        Calculate the MPS value  $w_{ia}$  for  $p_i \leftrightarrow q_a$ .
        if  $pmax < w_{ia}$ 
             $pmax = w_{ia}, \theta_0 = \theta^+, pair\_i = i, pair\_a = a$ 
    end
end

```

```

        end
    end
end

```

Pair $p_{pair_i} \leftrightarrow q_{pair_a}$ is taken as the principal pair and indicates the rotation difference θ_0 between P and Q .

5.2 Determination of the matching pairs

Localization of the matching pairs $p_j \leftrightarrow q_b$ among P and Q is executed in three stages. The first stage is based on the principal pair and the parameters S_{NORM} and θ_0 where $S_{NORM} = 1$. This method is actually very similar to the principal pair determination method. Calculations of s and θ^* are now performed for vectors $\overrightarrow{p_{pair_i}p_j}$ and $\overrightarrow{q_{pair_a}q_b}$. Those values are then compared toward S_{NORM} and θ_0 respectively. If there is only one vector in P and Q that satisfy the following rules $|s - S_{NORM}| \leq \Delta s$ and $|\theta^* - \theta_0| \leq \Delta\theta$ then the pair $p_j \leftrightarrow q_{ib}$ is matching pair. The matched pair can be collected to the point set called G . However, sometimes there are two or more points closer together and more then one vector satisfies the above rules. In that case identification of the matching pair is impossible to do by above mentioned method. Those pointes have to be stored in another point set called F and localization of the matching pairs is done in the second stage. Following pseudo code demonstrates the categorization of the different points to set G and F .

```

 $S_{NORM} = 1$ 
Put  $p_{pair\_i} \leftrightarrow q_{pair\_a}$  to the set  $G$ 
Set  $matching\_flag[b] = 1$  for all  $b$ .
for  $j = 1, \dots, m, j \neq pair\_i$ 
Set  $ib = 0, count = 0, Temp = []$ 
for  $b = 1, \dots, n, b \neq pair\_a$ 
    if  $matching\_flag[b] = 1$ 
         $s = \frac{|\overrightarrow{q_{pair\_a}q_b}|}{|\overrightarrow{p_{pair\_i}p_j}|}$ 
         $\theta^* = (\theta_{\overrightarrow{q_{pair\_a}q_b}} - \theta_{\overrightarrow{p_{pair\_i}p_j}}) \frac{180}{\pi}$ 
        if  $|s - S_{NORM}| \leq \Delta s \ \& \ |\theta^* - \theta_0| \leq \Delta\theta$ 
             $ib = b, count = count + 1$ 
            Collect the pair  $p_j \leftrightarrow q_b$  as a possible matching pair into
            temporally set  $Temp$ .
        end
    end
end

```

```

    end
  end
  if count = 1
    The  $p_j \leftrightarrow q_{ib}$  is taken as a matching pair and is collected in set  $G$ .
     $matching\_flag[ib] = 0$ 
  elseif count > 1
    Collect the points in set  $Temp$  to the set  $F$ .
  end
end
end

```

Observe that the θ^* is not quantized to get more accurate difference measurement between θ^* and θ_0 . However, it is recalculated to the interval of $[0, 359.75]$. The $matching_flag[b]$ serves as an indicator for points q_b that has yet not been uniquely matched. After the categorization of the points, set G holds the pairs that have been uniquely matched. Those pairs in set G are taken as the matching pairs. The pairs that are stored in set F are the pairs $p_j \leftrightarrow q_b$ where p_j have more then one q_b points to form a possible matching pair with.

An important aspect when categorizing the pairs into set G and F are the values of thresholds Δs and $\Delta \theta$. If those values are chosen too small the risk is that above mentions rules will not be fulfilled. Therefore high possibility that no pairs will be chosen to set G and F , even if the P is corresponding pattern to Q . If the threshold values are chosen to large there will be no unique pairs and all points will be collected to the set F . With empty set G the localization of matching pairs in set F can't be done. This means that the threshold values Δs and $\Delta \theta$ has to be carefully chosen.

The second stage of localizing the matching pairs is done by finding the unique pairs $p_j \leftrightarrow q_b$ in the set F . To do that the points p_j in set F are first transformed

$$p_{transf_j} = T_r(p_j) \quad (5.7)$$

to align with points q_b . Then the matching is done by measuring the distance between the point p_{transf_j} and q_b . The point q_b that is closest to the point p_{transf_j} with distance smaller then threshold d_1 as

$$\| T_r(p_j) - q_b \| \leq d_1 \quad (5.8)$$

is a matching pair $p_j \leftrightarrow q_b$. The matched pair $p_j \leftrightarrow q_b$ is taken away from set F and is collected to the set G .

Transformation of the p_j to $p_{transf-j}$ is done as following

$$p_{transf-j} = T_r(p_j) \Rightarrow \begin{pmatrix} x_{p_{transf-j}} \\ y_{p_{transf-j}} \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + \begin{pmatrix} s \cos \theta & -s \sin \theta \\ s \sin \theta & s \cos \theta \end{pmatrix} \begin{pmatrix} x_{p_j} \\ y_{p_j} \end{pmatrix} \quad (5.9)$$

where the t_x and t_y denotes the translation in x and y coordinates between P and Q point patterns. Estimation of the transformation parameters $r = [t_x, t_y, s \cos \theta, s \sin \theta]^T$ for the function $T_r(p_j)$ are based on matching pairs in set G . The following calculations are done to estimate the parameters

$$r = \frac{1}{det} \begin{pmatrix} l_P & 0 & -\mu_{x_P} & \mu_{y_P} \\ 0 & l_P & -\mu_{y_P} & -\mu_{x_P} \\ -\mu_{x_P} & -\mu_{y_P} & k & 0 \\ \mu_{y_P} & -\mu_{x_P} & 0 & k \end{pmatrix} \begin{pmatrix} \mu_{x_Q} \\ \mu_{y_Q} \\ l_{P+Q} \\ l_{P-Q} \end{pmatrix} \quad (5.10)$$

where

$$\mu_{x_P} = \sum_i^k x_{\tilde{P}_i}, \quad \mu_{x_Q} = \sum_i^k x_{\tilde{Q}_i} \quad (5.11)$$

$$\mu_{y_P} = \sum_i^k y_{\tilde{P}_i}, \quad \mu_{y_Q} = \sum_i^k y_{\tilde{Q}_i} \quad (5.12)$$

$$l_{P+Q} = \sum_i^k (x_{\tilde{P}_i} x_{\tilde{Q}_i} + y_{\tilde{P}_i} y_{\tilde{Q}_i}) \quad (5.13)$$

$$l_{P-Q} = \sum_i^k (x_{\tilde{P}_i} x_{\tilde{Q}_i} - y_{\tilde{P}_i} y_{\tilde{Q}_i}) \quad (5.14)$$

$$l_P = \sum_i^k (x_{\tilde{P}_i}^2 + y_{\tilde{P}_i}^2) \quad (5.15)$$

and

$$det = k * l_P - \mu_{x_P}^2 - \mu_{y_P}^2. \quad (5.16)$$

Note the denotation of \tilde{P} and \tilde{Q} in above equations. It is only used to emphasize the coordinates (x, y) of point belonging to the patterns P and Q . Therefore $x_{\tilde{P}_i}$ and $y_{\tilde{P}_i}$ denotes the (x, y) coordinates for points from pattern P found in set G . The $x_{\tilde{Q}_i}$ and $y_{\tilde{Q}_i}$ denotes the (x, y) coordinates for points from pattern Q found in set G . The k is the number of matching pairs located in set G and has to be $k \geq 2$. Meaning that set G needs to contain at least two matching pairs, otherwise parameters in r can't be estimated. The more matching pairs there is in the set G the more reliable the estimation gets.

The third stage of localization of matching pairs are done on the remaining points in P and Q . Depending on values of Δs and $\Delta\theta$ some pairs will not be selected to set G or F . Those remaining points are compared to each other only on measuring the distance between them. The rest of the points p_j in pattern P are first transformed with the same function $T_r(p_j)$. The closest points p_j and q_b whose distance to each other is under the threshold d_1 is selected as the matching pair. All remaining pairs $p_j \leftrightarrow q_b$ that are identified as matching pairs are collected to the set G . In this and previous stage it is important to decide appropriate value for d_1 . The bigger d_1 is the more distant points can be chosen as matching pairs. The risk is that some points are falsely selected as matching pair.

When all matching pairs between P and Q have been detected an error of Δx_i and Δy_i between pairs $p_i \leftrightarrow q_i$ in set G is calculated. This error is a measure on how well the matching pairs align with each other. The new transformation parameters are calculated based on all the matching pairs that have been detected. Therefore the new estimated vector r has optimal values $r_{opt} = [t_x, t_y, s \cos \theta, s \sin \theta]^T$. The error e_i for each matching pair $p_i \leftrightarrow q_i$ is calculated by

$$e_i = \begin{pmatrix} \Delta x_i \\ \Delta y_i \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + \begin{pmatrix} s \cos \theta & -s \sin \theta \\ s \sin \theta & s \cos \theta \end{pmatrix} \begin{pmatrix} x_{p_i} \\ y_{p_i} \end{pmatrix} - \begin{pmatrix} x_{q_i} \\ y_{q_i} \end{pmatrix} \quad (5.17)$$

and a sum of the squared errors is calculated, normalized with k which is the number of pairs in set G . The r_{opt} minimize the error E in a mean square sense. The equation is

$$E_{ms} = \frac{1}{k} \sum_{i=1}^k e_i^T e_i = [e_1^T e_2^T \dots e_k^T] \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_k \end{bmatrix} \frac{1}{k}. \quad (5.18)$$

5.3 Testing the matching algorithm

The developed matching algorithm is tested on two pairs of minutiae point patterns. The thresholds are chosen as following $\Delta s = 0.01$, $\Delta\theta = 0.5^\circ$ and $d_1 = 16$.

The first pair of patterns P with 58 points and Q with 63 points is originating from fingerprints belonging to the same person. One of the fingerprints

is rotated to test the ability of rotation invariance build in to the algorithm. The result can be seen in **Figure 5.1** where in the upper row are the point patterns before matching. In the lower row are the point patterns after the matching has been completed and the boxes that has turned blue indicate found matching pair. The principal pair is marked with black box. Totally 48 points has been determined as matching pairs which is about 83% and the $E_{ms} = 21$.



Figure 5.1: *An example of point pattern matching between two fingerprints that are originating from the same person. A 58 and 63 point are searched for matching points. In the upper row are the point patterns before matching and in the lower row are the point patterns after matching. The blue boxes indicates the matching pairs and the principal pair is highlighted with a black box. Totally 48 pairs has been found which is 83% and $E_{ms} = 21$*

In the second pair of patterns P with 54 points and Q with 63 points is originating from two different people. The Q pattern is the same as in the

previous example, only the P pattern has been changed. Selected from the fingerprints similar looking as the fingerprint in pattern Q . Test is performed to see how many matching pairs the algorithm is able to detect. The result can be seen in **Figure 5.2** where in the upper row are the point patterns before matching. In the lower row are the point patterns after the matching has been completed and the boxes that has turned blue indicate found matching pair. The principal pair is marked with black box. Totally 25 points has been determined as matching pairs which is about 46% and the $E_{ms} = 84$.

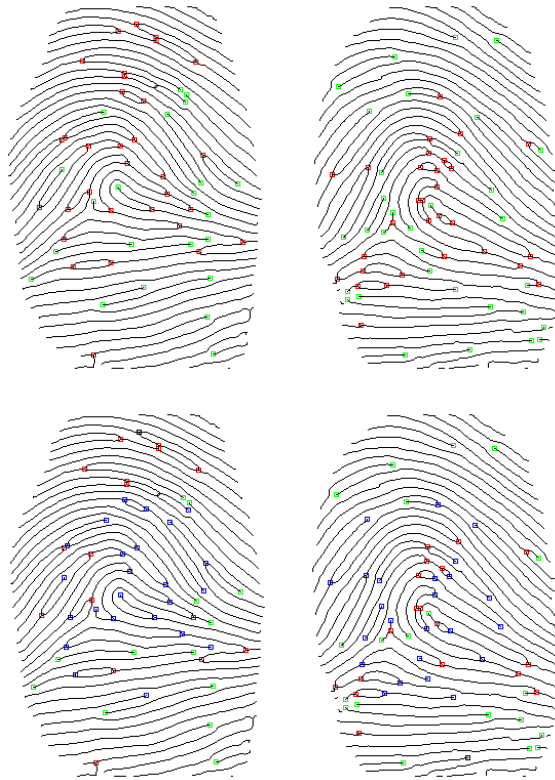


Figure 5.2: An example of point pattern matching between two fingerprints that are originating from the different persons. A 54 and 63 point are searched for matching points. In the upper row are the point patterns before matching and in the lower row are the point patterns after matching. The blue boxes indicate the matching pairs and the principal pair is highlighted with a black box. Totally 25 pairs has been found which is 46% and $E_{ms} = 84$

Chapter 6

Experimental Results

In previous chapters the diverse parts of the fingerprint verification system has been in depth explained. In this chapter the performance evaluation of the developed system is in detail described and the experimental results are presented.

A database was assembled from pre-stored fingerprints in FVC2002/Db1_a found on DVD accompanying the book [1]. Two fingerprints from 20 different people of good quality with varying rotation and translation were collected into the database. This database was then used to evaluate the performance of the fingerprint verification system. The matching is divided into two groups; examination of the identical fingerprints and examination of the non-identical fingerprints. The data of interest is the percentage of matched minutiae and mean squared error.

Table 6.1 shows the results for the matching of the identical fingerprint only. The values are the minimum, maximum and average of percentage matched minutiae and mean squared error.

	Min.	Max.	Average
E_{ms}	5.9720	41.0103	19.0690
[%] matched minutiae	52.3810	93.4783	81.1289

Table 6.1: *Results for the matching of the identical fingerprint.*

The results for the matching of the non-identical fingerprints can be found in **Table 6.2**. It is obvious that the two groups are mainly separated in the amount of matched minutiae given in percent.

	Min.	Max.	Average
E_{ms}	0	80.3645	43.7337
[%] matched minutiae	2.1739	50.9804	31.7193

Table 6.2: *Results for the matching of the non-identical fingerprint.*

To get a better view on how the two groups are separated, the data is plotted in the matched minutiae vs. mean squared error plane. The plotted data is viewed in **Figure 6.1**. The two groups are more or less forming clusters around the calculated average values.

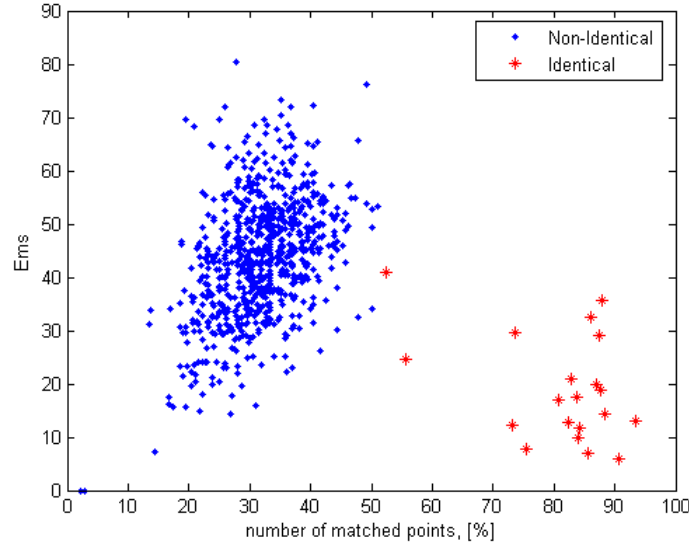


Figure 6.1: *Position of the examined fingerprint pairs for the two groups, plotted in the matched minutiae vs. mean squared error plane. The matched minutiae are given in the percentage.*

It can be seen that the cluster of the non-identical examples are somewhat directed. The smaller number of matched minutiae are, the smaller E_{ms} gets. While with increasing number of matched minutiae, the E_{ms} gets bigger. This is because the smaller number of matched minutiae offers better chance to be found closer to each other. While greater number of matched minutiae is often obtained by the minutiae distance matching method. The E_{ms} is in this case strongly related to the value chosen for the distance threshold d_1 in the matching algorithm. The bigger d_1 is the bigger E_{ms} gets for the larger

number of matched minutiae between the two non-identical fingerprints. This also explain why the non-identical cluster has greater overall E_{ms} . Notice also the two examples of the non-identical fingerprints matching which has the $E_{ms} = 0$ with very little percentage of the matched minutiae. This is due to the fact that fingerprints in those examples are that much different, that the matching algorithm found only one point pair in each.

The cluster of the identical examples is rounder except the two examples closer to the 50% of the matched minutiae boundary. Usually this is because of the two identical fingerprints have smaller mutual area and therefore less minutiae are being matched together. The overall E_{ms} is lower due to mutuality of the minutiae patterns that are being matched together. The matched minutiae aligns better to each other and the E_{ms} gets smaller.

Analysis of the experimental results shows that the developed system is capable of separating the identical from non-identical fingerprints. The separation is mainly in the percentage of the matched minutiae part. By choosing $\Theta_{mp} = 52$ the percentage of matched minutiae is classified as identical match if the value is larger then Θ_{mp} . The Θ_{mp} is a threshold for the matched minutiae in percent. It should be pointed out that this type of separation is quit dangerous! The separation between the non-identical and identical clusters is marginal of only about 1.5%. There should be at least one more checkup in form of examining the levels of the E_{ms} . A threshold Θ_{ms} is introduced to specify the maximum tolerance of E_{ms} that identical matched fingerprints can have. The threshold is chosen to $\Theta_{ms} = 42$. This method should be considered as a minimum of requirements for separating the two clusters apart from each others. Furthermore, good separation of the two groups is highly depending on the threshold values chosen for the diverse algorithms. In the test most of the parameters specified during the thesis reminds unchanged. The only parameters which has been modified are $\Delta s \Rightarrow 0.03$, $\Delta \theta \Rightarrow 0.7^\circ$ and the $d_1 \Rightarrow 13$.

The two fingerprints that are being matched should also have less then 50% of the difference in the number of minutiae. The larger difference in number of minutiae in the two point patterns is the higher chance there is to have higher number of matched minutiae. This is due to way the percentage of the matched minutiae is calculated. The formula is

$$percentage = number_of_matched_minutiae / \min([m \ n]) \quad (6.1)$$

where m and n are the sizes of the minutiae patterns. Therefore, fingerprints with higher difference in number of minutiae gives better chance for higher percentage of the matched minutiae.

Chapter 7

Summary and Conclusions

The theory behind the fingerprint verification based on minutiae matching, was in detail studied. With obtained knowledge the complete system has been designed and implemented in MATLAB. The performance of the developed system was evaluated on database with 2 fingerprints from 20 different people. The database was assembled from pre-stored fingerprints in FVC2002/Db1_a found on DVD accompanying the book [1]. The test showed that the system is fully capable of distinguishing the related fingerprints apart from the non-related fingerprints. The system has proved to be robust towards translation, rotation and/or missing minutiae between the matched fingerprints.

7.1 Further Work

The additional suggestions which can further improve or extent the system:

- The *Binarization* process can be improved by automating the filter customization. This enables adaptation of the filter to better suit each fingerprint that is being processed.
- Extension with "enrolment failed" due to the bad quality of the input fingerprint (to dry, wet, small area, etc...).
- Expansion to the fully identification system is possible by adding an algorithm that speeds up the searching time through the database. This is the only difference between the verification and identification.
- Because the fingerprint verification system is implemented in MATLAB it executes quite slow. By porting the system to the C or assembler language its run time can be speeded up.

- The fingerprint verification system could be build as a self-standing module. By implementing it in the FPGA using VHDL or Verilog, the system would be fast due to the parallelism that this technology provides.

Bibliography

- [1] D. Maltoni, D. Maio, A.K. Jain and S. Prabhakar, *Handbook of Fingerprint Recognition*, Springer, 2003, ISBN 0-387-95431-7.
- [2] O’Gorman L. and Nickerson J.V., “Matched Filter Design for Fingerprint Image Enhancement,” in *Proc. Int. Conf. on Acoustic Speech and Signal Processing*, pp. 916-919, 1988.
- [3] O’Gorman L. and Nickerson J.V., “An Approach to Fingerprint Filter Design,” *Pattern Recognition*, vol. 22, no. 1, pp. 29-38, 1989.
- [4] Kwon J.S., Gi J.W. and Kang E.K., “An Enhanced Thinning Algorithm using Parallel Processing,” in *Proc. Int. Conf. on Image Processing*, pp. 752-755, 2001.
- [5] S. Haykin, *Neural Networks: A comprehensive foundation*, Prentice-Hall, second edition, 1999, ISBN 0-13-273350-1.
- [6] R.D. Reed and R.J. Marks II, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, The MIT Press, 1999, ISBN 0-262-18190-8.
- [7] Chang S.H., Cheng F.H., Hsu W.H. and Wu G.Z., “Fast Algorithm for Point Pattern-Matching: Invariant to Translations, Rotations and Scale Changes,” *Pattern Recognition*, vol. 30, no. 2, pp. 311-320, 1997.