Low Complexity Embedded Fingerprint Verification and Identification System

Vorgelegt von
M.Sc.
Wei Sun

Von der Fakultät IV – Elektrotechnik und Informatik –
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr.-Ing. Reinhold Orglmeister, TU-Berlin
Gutachter:	Prof. Dr.-Ing. Thomas Sikora, TU-Berlin
Gutachter:	Prof. Dr. Atta Badii, University of Reading
Gutachter:	Dr.-Ing. Ivo Keller, TU-Berlin

Tag der wissenschaftlichen Aussprache: 24.November 2015

Berlin 2016
D83

Erklärung

Hiermit versichere ich an Eides statt, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner Prüfungsbehörde vorgelegt.

Berlin, den 9. Sep. 2013 _____

(Wei Sun)

Acknowledgement

This research work would not have been possible without the continuous support and assistance of some people.

First of all, I would like to thank my mentor Mr. Prof. Sikora, who offered me this opportunity to pursue this dissertation research. I am very grateful for his wise guidance, advice and suggestions all the time.

I would also like to thank my boss and supervisor Mr. Dipl. Zeng. Through several years' team work, I have learned a lot from him not only in researching field. Moreover many troublesome problems could almost be smoothly resolved after each discussion.

My sincere thanks go also to all the members of my family. Without my wife's great support, understanding and love I could not finish this work. To my son, your smile is the best encouragement and power for me.

Abstract

A qualified fingerprint verification and identification system should resolve most of the problems of fingerprint image enhancement and fingerprint matching. In practice, the fingerprint images are not ideal and may have totally different qualities when the fingers are too moist too dry, improperly placed, or from some users who are engaged in heavy manual work. Furthermore nonlinear deformations in fingerprint images by reason of the elasticity of the skin as well as the pressure and the movement of the finger during image acquisition, result in bad matching results in verification and identification even from the same finger.

In this thesis I designed a fingerprint-based biometric system in order to solve most of the above problems in both software and hardware application. Fingerprint identification and verification algorithms were first developed for PC, then optimized and implemented in an ARM9-based embedded system.

Zusammenfassung

Ein gutes Fingerabdruck Verifikations- und -Identifikationssystem sollte die meisten Probleme bei Verbesserung und Vergleich von Fingerabdruckbildern lösen. In der Praxis sind Fingerabdruckbilder jedoch nicht ideal und können völlig unterschiedliche Eigenschaften haben, wenn die Finger zu feucht oder zu trocken sind, falsch auf den Sensor gelegt wurden oder abgenutzt sind. Zusätzlich treten durch die Elastizität sowie Druck und Bewegung des Fingers während der Bilderfassung nichtlineare Verzerrungen in Fingerabdruckbildern auf, die zu schlechten Ergebnissen bei Identifizierung und Verifikation führen.

In dieser Arbeit wurde ein Fingerabdruck-basiertes biometrisches System entworfen, das die meisten der genannten Probleme lösen soll. Algorithmen zur Identifizierung und Überprüfung von Fingerabdrücken wurden zuerst auf dem PC entwickelt, und dann optimiert und auf einem ARM9-basierten eingebetteten System implementiert.

Contents

1 Introduction 1

1.1 Biometrics 1

1.2 Embedded System 7

1.3 Overview of the Thesis 8

2 Fingerprint Image Enhancement 10

2.1 Introduction 10

2.2 State of the Art 11

2.2.1 Segmentation 11

2.2.2 Normalization 12

2.2.3 Ridge Orientation Estimation 13

2.2.4 Ridge Frequency Estimation 16

2.2.5 Filtering 18

2.3 Experimental Improvements 20

2.3.1 Directional Image Filtering 20

2.3.2 Noise Image Evaluation 24

2.3.3 Frequency Image Improvement 26

2.4 Summary 28

3 Fingerprint Feature Extraction 31

3.1 Introduction 31

3.2 State of the Art 33

3.2.1 Binarization 33

3.2.2 Thinning 35

3.2.3 Minutiae Extraction 37

3.2.3.1 Extraction with Unthinned Binarized Images 38

3.2.3.2 Skeletonization-based Methods 40

3.2.3.3 Extraction with Grey-Level Images 43

3.3 Application and Optimization 44

3.3.1 Grey-Level Image Binarization 44

3.3.2 Minutiae Extraction 49

3.3.3 Minutiae Post-processing 51

3.4 Experimental Improvements 54

3.5 Summary 61

4 Fingerprint Matching 62

4.1 Introduction 62

4.2 State of the Art 65

4.3 Application and Optimization 68

4.4 Experimental Results 89

4.5 Summary 99

5 Fingerprint Embedded System 100

5.1 Hardware Architecture 100

5.1.1 Introduction 100

5.1.2 Sensor Techniques 102
5.1.3 ARM Processor 105
5.1.4 SPI and Flash Organization 107
5.1.5 Serial Interface 109
5.1.6 I^2C 114
5.1.7 Infrared Keypad 116
5.2 Software Implementation and Optimization 119
5.2.1 System Boot 119
5.2.2 MMU and Caches 121
5.2.3 Interrupt Handling 124
5.3 Summary 125
6 Conclusions and Future Work 126
6.1 Conclusions 126
6.2 Future Work 128
Appendix A: FVC – Fingerprint Verification Competition 129
Bibliography 133

List of Figures

Figure 1.1: Deployment of biometrics systems at border crossings for immigration control: a) face recognition system (Smart Gate) at Sydney airport; b) iris recognition system at Amsterdam Schiphol airport; c) at Manchester airport (UK); d) at UAE airport; e) fingerprint recognition using index fingers at airports in Japan; f) ten fingerprint acquisition at airports in the United States; g) fingerprint based immigration clearance for passengers at Hong Kong airport; h) vehicular clearance using fingerprint and face in Hong Kong.[3] 4
Figure 1.2: The relations between EER, FAR and FRR. 6
Figure 1.3: The ARM and Cortex families. 8
Figure 2.1: Fingerprint images with various qualities from FVC2006_DB3_A. a) A good quality fingerprint; b) A medium quality fingerprint; c) A poor quality fingerprint. 11
Figure 2.2: Image normalization. a) Original image; b) Image after normalization. 13
Figure 2.3: Ridge orientation estimation. a) Original image; b) Orientation image. 14
Figure 2.4: A 9×9 window in slit-based approach. 15
Figure 2.5: Ridge frequency estimation. a) Original image; b) Frequency image. 17
Figure 2.6: Ridge frequency computation. 17

Figure 2.7: Image filtering. a) Original image; b) Filtered image. 18
Figure 2.8: Gabor Filter with the parameters $\theta = 135°, f = 1/5, \sigma_x = \sigma_y = 4$. 20
Figure 2.9: Image direction layout.....21
Figure 2.10: Computation window $(w = 12)$. 21
Figure 2.11: Directional image filtering. a) Fingerprint grey-scale images; b) Images before directional image filtering; c) Images after directional image filtering. 23
Figure 2.12: Noise image evaluation. a) Original noise image; b) Optimized noise image; c) Original binary image; d) Optimized binary image. 25
Figure 2.13: 33×33 windows with different local frequencies. 26
Figure 2.14: Frequency image improvement. a) Original frequency image; b) Filtered frequency image; c) Quantized frequency image. 28
Figure 2.15: Results of fingerprint image enhancement with optimized algorithm: a) Input image; b) Directional filtered image; c) Directional filtered image with ridge estimation. 29
Figure 3.1: Various singularities of fingerprint in FVC2006_DB3_A. 31
Figure 3.2: Termination minutia $x0, y0, \theta 0$ and bifurcation minutia $x1, y1, \theta 1$.32
Figure 3.3: Efficiency of binarization-based method for images with different quality and acquisition condition: a) A medium quality fingerprint with scratches and ridge breaks; b) A poor quality fingerprint with too much acquisition noise. 32
Figure 3.4: Fingerprint images with different contrast: a) Appropriate for global threshold; b) Appropriate for local threshold. 34
Figure 3.5: Binarized image thinning. a) Binarized image; b) Thinning image. 35
Figure 3.6: Neighbour of P5 in OPTA. 36
Figure 3.7: Restoring patterns in OPTA. 37
Figure 3.8: Thinning patterns in OPTA. 37
Figure 3.9: Classifications of minutiae extraction techniques. 38
Figure 3.10: Minutiae extraction using contour tracing in binary image. 39

Figure 3.11: 3×3 window for Crossing Number scanning. 41
Figure 3.12: Types of Crossing Number. 42
Figure 3.14: The structure element sequence for ridge bifurcations. 43
Figure 3.15: Stages of the proposed grey-level image binarization algorithm. 45
Figure 3.16: Results of various stages in grey-level image binarization. 48
Figure 3.17: Thinning patterns improvements.....50
Figure 3.18: The proposed thinning patterns. 51
Figure 3.19: One minutia pair for post-processing. 52
Figure 3.20: Comparison of binarization from a fingerprint of good quality with various methods. 55
Figure 3.21: The sample set of fingerprints referred in [8] and used in our comparison. All the fingerprints (except no. 6, 7 and 8) have dimensions 256×256 with 256 grey levels. The fingerprints no. 6, 7 and 8 have dimensions 190×250 with 256 grey levels. 56
Figure 3.22: Automatic minutiae detection in fingerprint no.1 - class good using approach A, the proposed, B and E (from left to right). The first row is the input image successively with its smoothing, binarization, thinning and minutiae extraction results. 57
Figure 3.23: Automatic minutiae detection in fingerprint no.13 - class poor using approach A, the proposed, B and E (from left to right). The first row is the input image successively with its smoothing, binarization, thinning and minutiae extraction results. 58
Figure 4.2: Feature vector between two minutiae. 71
Figure 4.3: Minutiae filtering using block direction image. 73
Figure 4.4: Minutiae filtering using block direction image. 73
Figure 4.5: Minutiae similarity decided with singularity. 76
Figure 4.6: Minutiae similarity decided with midpoint. 77
Figure 4.7: Minutiae local orientation descriptor. 80
Figure 4.8: Minutiae local neighbour descriptor. 80

Figure 4.9: Comparisons of feature vectors before and after optimization for Finger Distortion: a) Original grey fingerprint image; b) Feature vectors before optimization; c) Feature vectors after optimization. 83
Figure 4.10: Correlations between a selected minutia and one singularity or two singularities. 85
Figure 4.11: Minutiae selection for transformation parameters using spatial distance, minutia score and curvature. 88
Figure 4.12: Fingerprint examples from FVC databases. 90
Figure 4.13: Experimental results of our algorithm on FVC2004 DB3_A. Part d) shows the ROC results from the participants in FVC2004..... 91
Figure 4.14: Experimental results of our algorithm on FVC2006 DB3_A. 92
Figure 4.15: Matching failed due to insufficient overlap. 97
Figure 4.16: Matching failed due to global distortion..... 98
Figure 5.1: Hardware architecture based on ARM9G20. 101
Figure 5.2: FingerChip packages of AT77C102B. 103
Figure 5.3: AT77C102B pin description. 103
Figure 5.4: Flow diagram of fingerprint image acquisition..... 104
Figure 5.5: AT91SAM9G20 block diagram. 106
Figure 5.6: SPI data transmission. 107
Figure 5.7: SPI connections between MX25L1605 and AT91SAM9G20. .. 107
Figure 5.8: SPI data transmission with DMA..... 109
Figure 5.9: Serial flash organizations. 109
Figure 5.10: USART to RS-485 transceiver. 110
Figure 5.11: Flow diagrams of USART communications. 112
Figure 5.12: Data structure of USART. 112
Figure 5.13: Schematic diagram of the I^2C relay board. 115
Figure 5.14: Infrared receiver and remote controller. 117
Figure 5.15: Remote output waveform of NEC format. 118
Figure 5.16: Interrupt vector table. 120
Figure 5.17: MMU section descriptor. 121
Figure 5.18: Enabled interrupts in BioKey4000..... 124

List of Tables

Table 3.1: Minutiae extraction comparison 59
Table 3.2: Average error percentage relative to the fingerprint of class good. 59
Table 3.3: Average error percentage relative to the fingerprint of class poor. 60
Table 3.4: Average error percentage relative to the fingerprint of the whole sample set. 60
Table 4.1: Performance comparison with FVC2004 DB3_A 93
Table 4.2: Performance comparison with FVC2006 DB3_A 93
Table 4.3: Comparison of the high-level description of the algorithms between the 29 participants in FVC2004 and the proposed algorithm . 94
Table 4.4: Hard difference between FVC (right) and us (left) 95
Table 5.1: Internal memory mapping 120
Table 5.2: Translation table definition 122
Table 5.3: Comparisons of the encoding performance with or without ICache and DCache..... 123

Chapter 1

Introduction

1.1 Biometrics

Biometrics is the technology that uses unique and measurable physiological and behavioural characteristics to distinguish one person from the others. From these unique features many different research fields have been expanded, the major of which are fingerprint recognition, hand geometry recognition, facial recognition, iris recognition, voice recognition, keystroke recognition and signature recognition.

These biometric technologies can be also separated into two categories: behavioural biometrics and physical biometrics [1]. Although behavioural biometrics includes all of the mannerisms or behaviour from an individual, it cannot be captured by a biometric system. Signature as well as keystroke recognition belongs to this category. On the other side physical biometrics has so many biological and physiological features, which can be captured by a biometric system. Fingerprint recognition, hand geometry recognition, facial recognition, iris recognition, voice recognition belong to this category.

Fingerprint recognition has been used for over a century and given criminal investigations a great aid. Every fingerprint has its own ‘ridges’ and ‘valleys’, which compose the loops, arches and swirls in some special positions. The ridges and valleys have different kinds of breaks and discontinuities. They are called ‘minutiae’, most of which will not change during a person’s life. The direction, location and type information of the reliable minutiae will be computed and used to compare with these from another finger.

Hand geometry recognition is based on the structure of the hand such as its shape, size of palm, the length, width and thickness of the finger, the distances between finger joints. Hand geometry-based verification systems are simple, inexpensive and relatively easy to use. Dry weather or individual anomalies such as dry skin will not make the verification accuracy of the hand geometry-based systems worse. But its physical size limits its use. It cannot be used in many embedded systems such as laptop, cell phone etc.

Facial Recognition uses facial images as biometric characteristic to distinguish one person from another. This biometric characteristic is perhaps the most prominent and intuitive of all recognition methods. Two basic clues of this approach are that: 1) the location and shape of facial attributes such as the eyes, ears, nose, lips and chin, and their spatial relationships, or 2) the global analysis of the face image that represents a face as a weighted combination of a number of standard faces. The result of facial recognition depends much on the capture angle of facial images. [2]

Iris recognition relies on the distinctive features of the iris. The iris, which is one colour region, is located between the pupil and the sclera. Its texture such as freckles, furrows, rings, and the corona is stable and unique to an individual, which are also very difficult to tamper.

Voice recognition utilizes the characteristic of the sound, which is different from person to person, to identify an individual. The vibration of the words from the vocal tract as well as the shapes and sizes of the articulators change at all times. By this mean it is also one of the behavioural biometrics to some extent. Voice-based recognition is sensitive to the background noise.

Keystroke recognition is applied by comparing the particular way in which an individual types on a computer keyboard. Each individual has its own typing speed, the favourite time of some keys down and the interval between two continuous keystrokes.

Signature recognition makes records of the way and the manner in which we sign our name. Signature recognition system analyses the timing, pressure and speed during the signing process and surely the signature itself. But the signature may change at every time even by the same person. Moreover, professional forgers can simulate the signature precisely.

According to the development of biometrics technologies, they can be separated into several generations. Before the first generation of automatic fingerprint identification system (AFIS) came into use in 1970s, the earlier generation has limited performance and lack of interconnectivity (standalone). The first generation systems including all the technologies mentioned above have then widely used in civilian and commercial systems. Some examples are shown in Figure 1.1, which are mostly used at international border crossings. In spite of the advancement in sensor technology, computer speed and memory storage, the first generation still cannot meet the request of real-time computation in some cases and vulnerable to be attacked. Therefore the second generation biometrics systems in the present aim at firstly solving such left problems and then developing more techniques to improve such as data acquisition environment, user convenience, data acquisition quality and the capability to handle poor quality data, and to protect user privacy more efficiently.[3]

a) b) c) d)

e) f)

g) h)

Figure 1.1: Deployment of biometrics systems at border crossings for immigration control: a) face recognition system (Smart Gate) at Sydney airport; b) iris recognition system at Amsterdam Schiphol airport; c) at Manchester airport (UK); d) at UAE airport; e) fingerprint recognition using index fingers at airports in Japan; f) ten fingerprint acquisition at airports in the United States; g) fingerprint based immigration clearance for passengers at Hong Kong airport; h) vehicular clearance using fingerprint and face in Hong Kong.[3]

In each kind of biometrics people are concerned with two problems: identification and verification. In identification, the enrolled templates try to tell the biometric system whether they are authenticated members by being compared with the other templates in the database, which is a one-to-N process. In verification, the enrolled templates with definite identification (ID) codes are compared with the templates with the same ID in the database in order to verify whether these newly enrolled templates are agreeable with the ID that they have, which is a one-to-one process. In contrast with popular identification methods, such as identification cards, personal identification numbers (PINs), or passwords, the most evident advantage of biometrics is that it is more secure and difficult to forger. Furthermore, it is more convenient for the users who utilize such biometric systems every day.

Both identification and verification must supply a YES or NO as the answer, which is based on some matching algorithm. The latter generates a score which represents the similarity between the template to identify or verify and the matched template in the database. Each biometric system has at least one threshold for this matching score. If the score is lower than this threshold, the template will not be accepted by this biometric system.

In order to give a standard performance evaluation of these matching algorithms, three important factors are introduced – the false acceptance rate (FAR), the false rejection rate (FRR) and the equal error rate (EER).

The FAR is defined as the percentage of individuals who are not correctly identified:

$$FAR(\%) = \frac{num\ of\ incidents\ of\ false\ acceptance}{total\ number\ of\ samples\ presented} \times 100\% \tag{1.1}$$

The FRR is defined as the percentage of individuals who should be identified, but are not:

$$FRR(\%) = \frac{num\ of\ incidents\ of\ false\ rejections}{total\ number\ of\ samples\ presented} \times 100\% \tag{1.2}$$

The EER which is also called crossover rate is the point where the FAR and the FRR have the same value, see Figure 1.2.

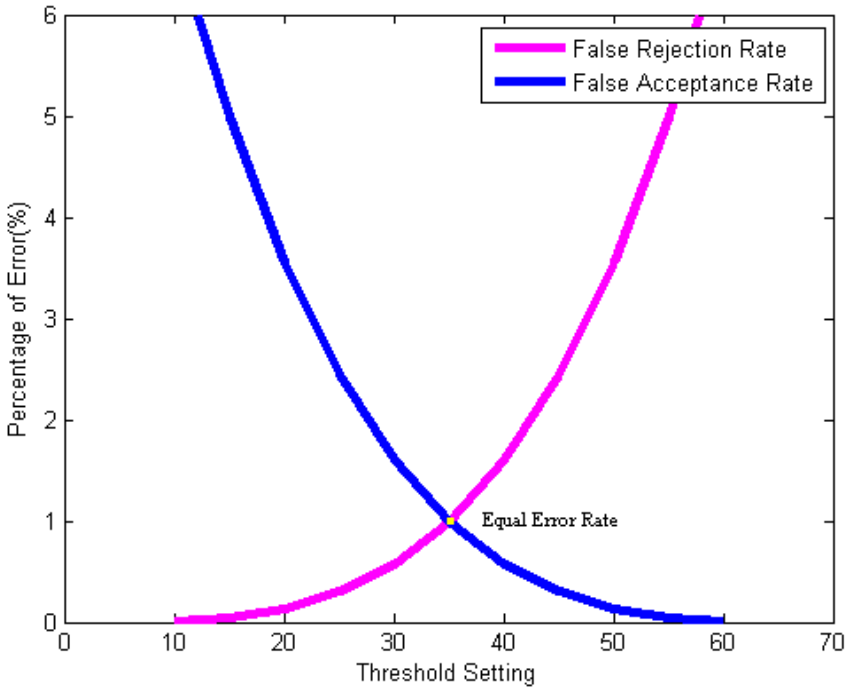


Figure 1.2: The relations between EER, FAR and FRR.

In addition, there are some other standards such as the failure to enroll rate (FTER) and the ability to verify rate (ATV). The FTER is defined as the statistical probability that an individual is unable to enroll successfully in a biometric system:

$$FTER(\%) = \frac{num\ of\ incidents\ of\ unsuccessful\ enrollment}{total\ number\ of\ enrollment\ attempts} \times 100\% \tag{1.3}$$

The ATV implies the overall percentage of how difficult users can be verified by a biometric system. It is also related with the FTER and the FRR described above:

$$ATV = [(1 - FTER) \times (1 - FRR)] \tag{1.4}$$

1.2 Embedded System

Embedded system is actually such a software-hardware system which faces on practical application and puts emphasis on its main processor. The hardware as the basic platform of the whole embedded system provides exhaustive physical communicating interfaces, on the other side the software including operating system and application software, which are the controlling centre of this system, provides the operation of Human-Machine Interaction. Therefore, the development of embedded system includes two different parts. Hardware design contains the selection of processor, memory, communication interfaces, I/O and power supply. Software design refers to the programming of operating system and application program, especially software-interrupt debugging, code optimization and robustness.

The aim of hardware selection is actually to search suitable microprocessor. The ARM processor, which is a 32-bit reduced instruction set computer (RISC) instruction set architecture (ISA), is considered as the most popular and universal embedded processor at the moment. The ARM processor has various families with different performance, the principal of which are the following: ARM7, ARM9, ARM11 and ARM Cortex.

The ARM7 processor family was introduced in 1994 and continuous to be used today for simple 32-bit devices. Its successor is the ARM9 processor family, which enables single processor solutions for microcontroller, DSP and Java

applications, offering savings in chip area and complexity, power consumption etc. The representative core of the ARM9 processor family is ARM926EJ-S with enhanced 32-bit RISC CPU, flexible size instruction and data caches, tightly coupled memory (TCM) interfaces and memory management unit (MMU). Parallel with classic arm families, ARM Cortex family has its special advantages such as lower power consumption, higher code density etc. The ARM11 family and other later ARM Cortex processors are used for higher performance application and will not be discussed in this thesis.

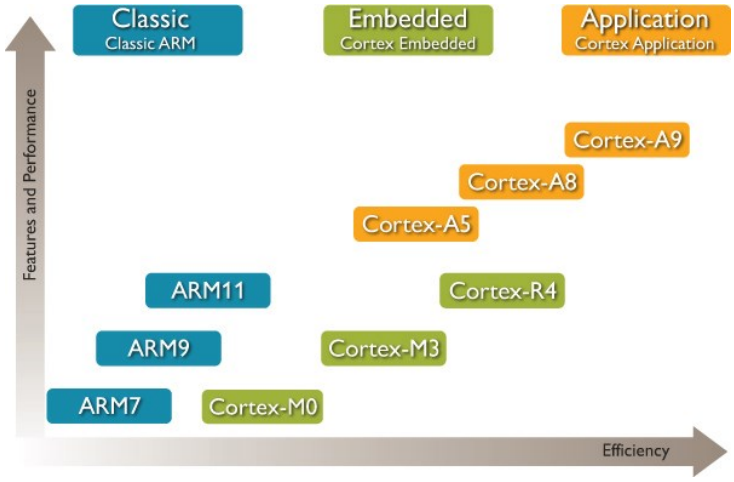


Figure 1.3: The ARM and Cortex families.

1.3 Overview of the Thesis

In this thesis a minutiae-based fingerprint encoding and matching approach is proposed and implemented in ARM9-based embedded system. Through further optimizations of the generation of binarized- and skeletonized- fingerprint image, more reliable minutiae are extracted and unreliable minutiae are rejected. And then a novel multiple-matching algorithm is developed to improve traditional minutiae-based fingerprint matching approaches. The main contribution of this thesis is that it provides a reliable and fast fingerprint identification and verification method applicable not only on PC platform but also on embedded hardware system.

This thesis is organized as follows. Chapter 2 and Chapter 3 give a general review of image enhancement, feature extraction methods and our improvements. Chapter 4 introduces various fingerprint matching algorithms and our hardware oriented fingerprint matching. Chapter 5 emphasizes the implementation of the fingerprint feature extraction and matching on embedded hardware platform. Chapter 6 summarizes the main work of thesis and gives some suggestions of future work.

Chapter 2

Fingerprint Image Enhancement

2.1 Introduction

During the acquisition of fingerprint images some undesired factors such as moist or dry fingers, cuts, bruises and incorrect finger pressure will lead the input images to bad quality. In contrast with an ideal fingerprint, a practical fingerprint has discontinuous ridges and valleys with unequalled width. In Figure 2.1 different fingerprints from the same Atmel senor indicate great difference of quality.

The degradation of fingerprint images makes the ridge extraction and minutiae extraction very unreliable. As a consequence many spurious minutiae instead of genuine minutiae are extracted. Furthermore even the extracted genuine minutiae may have false position and orientation.

The enhancement algorithm aims to find various methods to recover the ridges and valleys as much as possible. Some classic image enhancement techniques, such as contrast stretching, histogram manipulation, normalization[4], have been used as the initial preprocessing. Besides more sophisticated enhancement algorithms have also been developed for better results.

a) 79_6.bmp b) 64_9.bmp c) 20_1.bmp

Figure 2.1: Fingerprint images with various qualities from FVC2006_DB3_A. a) A good quality fingerprint; b) A medium quality fingerprint; c) A poor quality fingerprint.

2.2 State of the Art

2.2.1 Segmentation

Segmentation, as the initial step of fingerprint image preprocessing, aims to separate fingerprint area (foreground) from the original image background, which makes the preprocessing only valid for the foreground and reduces the preprocessing time quite a lot. Moreover the extraction of false features in noisy areas, which will greatly degrade the latter enhancement, can be avoided.

The classic segmentation algorithms are based on the variance of each block and oriental information from the directional image. Ridge orientation is estimated at each pixel and a histogram is computed for each 16 × 16 block[5]. According to the distribution of the signal in the histogram, oriented ridges and valleys can be located at the peak signal and isotropic areas are represented as a flat region. When a block has no ridge information such as a background block, grey-scale

variance of this block is needed to be computed[6, 7] since the background area has uniform grey value and also low-variance. A more efficient method proposed in [8] is to use the average magnitude of the gradient in each image block. Since gradient response uses the differential between the grey value of the two neighbor pixels in horizontal and vertical orientation, it can differentiate the fingerprint area and background area faster than grey-scale variance method.

Apart from this, some other theories are also applied in this field such as Gabor-filters-based[9], Fourier-spectrum-based[10] and Harris-corner-detector-based[11]. And in [12-15] more accurate segmentation approaches based on feature value threshold are proposed, whereas their computational complexity is relatively higher.

2.2.2 Normalization

In the normalization, input fingerprint images are normalized to limit the dynamic range of the grey scale between ridges and valleys of the images, which facilitates the processing of the following steps such as orientation image estimation and filtering. Figure 2.2 shows an image before and after normalization.

The input image is divided into sub-blocks with the size $N \times M$. As described in [4], a fingerprint image I is defined as $N \times M$ matrix and $I(i,j)$ is the grey value of the pixel at the ith row and jth column:

$$I'(i,j) = \begin{cases} M_0 + \sqrt{\dfrac{VAR_0(I(i,j) - \widehat{M})^2}{\widehat{VAR}}} & I(i,j) > \widehat{M} \\ M_0 - \sqrt{\dfrac{VAR_0(I(i,j) - \widehat{M})^2}{\widehat{VAR}}} & otherwise \end{cases} \tag{2.1}$$

In (2.1) M_0 and VAR_0 are the desired mean and variance values; $\widehat{M}$ and $\widehat{VAR}$ are the computed mean and variance of the given image. The desired M_0 and VAR_0 should be pre-computed from the origin image, which is reliable but maybe also of poor quality.

a) FVC2000_DB2_B:104_1.tif b)

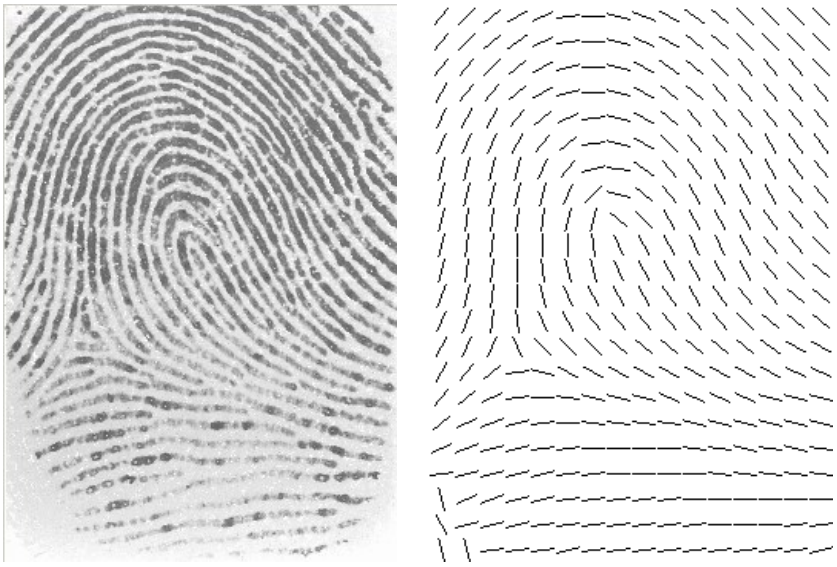
Figure 2.2: Image normalization. a) Original image; b) Image after normalization.

2.2.3 Ridge Orientation Estimation

The fingerprint orientation image, which is also called directional image, demonstrates the local orientation of the fingerprint ridges. The orientation of the ridges supply important clues to the completeness of each ridge, with which most of the cuts, creases and bruises can be found and repaired. In Figure 2.3 a block-wise orientation image is shown with its corresponding original input image.

The prominent approach to determine local ridge orientation is from the computation of gradients in the fingerprint images. The gradient $\nabla(x_i, y_i)$ is one two-dimensional vector $[\nabla_x(x_i, y_i), \nabla_y(x_i, y_i)]$ at point $[x_i, y_i]$ in the image, where ∇_x and ∇_y denote the derivatives with reference to the x and y directions,

respectively. The gradient orientation is actually the direction of the maximum pixel-intensity at point $[x_i, y_i]$. Therefore its orthogonal angle represents the direction of the ridges at point$[x_i, y_i]$.



a) FVC2000_DB2_B: 104_1.tif b)

Figure 2.3: Ridge orientation estimation. a) Original image; b) Orientation image.

In [7, 16] such gradient-based method is proposed within each 17×17 block centered at point$[x_i, y_i]$:

$$\theta_{ij} = 90° + \frac{1}{2}\arctan\left(\frac{2G_{xy}}{G_{xx} - G_{yy}}\right)$$

$$G_{xy} = \sum_{h=-8}^{8}\sum_{k=-8}^{8} \nabla_x\left(x_i + h, y_j + k\right) \cdot \nabla_y\left(x_i + h, y_j + k\right)$$

$$G_{xx} = \sum_{h=-8}^{8}\sum_{k=-8}^{8} \nabla_x\left(x_i + h, y_j + k\right)^2$$

$$G_{yy} = \sum_{h=-8}^{8}\sum_{k=-8}^{8} \nabla_y(x_i + h, y_j + k)^2$$

(2.2)

where ∇_x and ∇_y are the x- and y-gradient components computed with 3×3 Sobel or Prewitt masks.

Another novel one called slit-based approach has been proposed by [5, 17, 18] which is to select the best slit based on the pixel grey-values along the slits. It was firstly used in local threshold binarization and then in computation of local ridge orientation with some revision of this threshold. The algorithm of this method is that pixels along the ridge orientation obtain small sum of grey-values, whereas along the valley orientation relative high. Its computational complexity is relatively higher than gradient-based approach.

The basic principle is shown in Figure 2.4. The center C is the pixel whose direction is to be computed. The 9×9 window which is also variable is selected around C. The sum of the four pixel values s_i along the direction marked as i will be firstly computed $(i = 1 \cdots 8)$. The direction is then selected based on the following formula:

$$Direction(C) = \begin{cases} Direction(s_{max}) \;\; if \; 4C + s_{max} + s_{min} > \dfrac{3}{8}\displaystyle\sum_{i=1}^{8} S_i \\ Direction(s_{min}) \;\; otherwise \end{cases} \tag{2.3}$$

7		8		1		2		3
6		7	8	1	2	3		4
		6				4		
5		5		C		5		5
		4				6		
4		3	2	1	8	7		6
3		2		1		8		7

Figure 2.4: A 9×9 window in slit-based approach.

The direction of the local ridges and valley is computed at each pixel where the direction of the slit with the minimal sum of s_i is assigned if this pixel is defined as a black ridge point and otherwise the direction of slit with the maximal sum of s_i is assigned for the pixel as a white valley point.

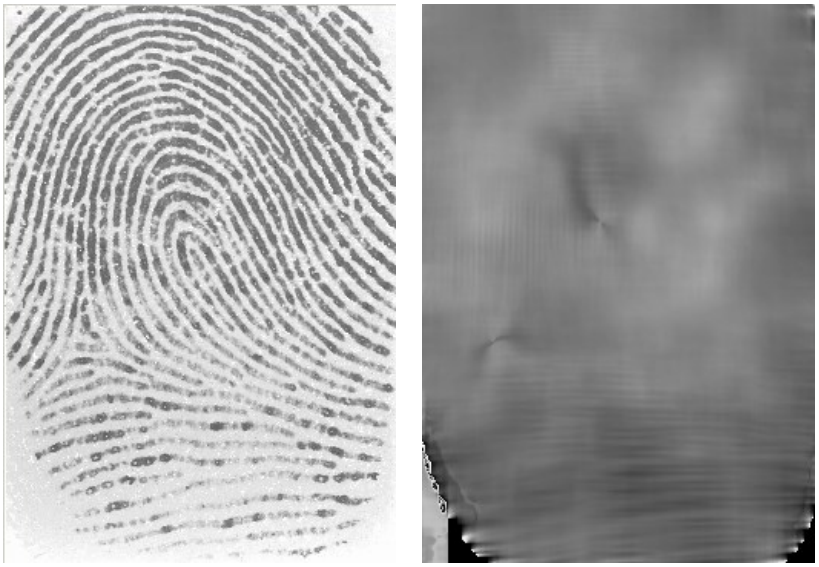
Another projection-based approach proposed in [19] is one relative intuitive method. Similar to the following slit-based approach, a rectangle window is defined around one pixel and then rotated to 8 or 16 directions. At each direction this window is projected to y axis of the image. When the x axis of this window is perpendicular to the local ridge, this projection obtains the maximal change. When the x axis of this window is parallel to the local ridge, this projection obtains the minimal change. But its computational complexity is usually higher than gradient-based methods.

2.2.4 Ridge Frequency Estimation

Alternate ridges and valleys constitute a line-shaped fingerprint image. In some certain local area of a fingerprint image, the grey values are distributed approximately in sinusoidal shape along the direction orthogonal to local ridge orientation. The frequency of this sinusoid can be considered as the frequency of the ridges in this local area, with which some latter enhancement can be executed such as filtering. In Figure 2.5 ridge frequency image is shown with an original input image.

In Figure 2.5 b) light areas signify higher frequencies. One popular approach for ridge frequency estimation proposed in [4] is to estimate local ridge frequency by analyzing the period of the ridges along the direction orthogonal to local ridge orientation. In Figure 2.6 the frequency f_{ij} at $[x_i, y_i]$ can be obtained by accumulating the grey values along a rotated orientation window (32×16 in this figure). Between the two dash lines a valley of grey values can be computed,

with which the period of ridges and valleys within this window is computed as the frequency at $[x_i, y_i]$.



a) FVC2000_DB2_B: 104_1.tif b)

Figure 2.5: Ridge frequency estimation. a) Original image; b) Frequency image.

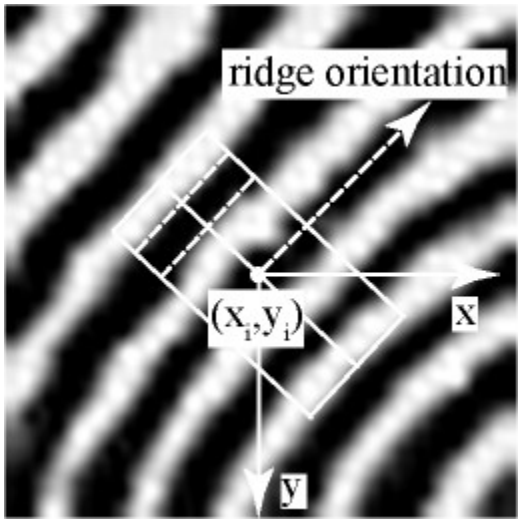


Figure 2.6: Ridge frequency computation.

Besides this approach some other methods such as in [20] with a high-order spectrum technique, [21] with scale space theory are proposed to locally estimate ridge width.

2.2.5 Filtering

Normalization introduced in section 2.2.2 belongs to pixel-wise method. In practice this is far not enough to represent a perfect fingerprint image. Another kind of enhancement based on section 2.2.3 and 2.2.4 which is called contextual filters. These filters execute convolution operations with different parameters based on local information of ridge direction and frequency. Furthermore, in order to get better computation efficiency such convolution matrixes are pre-computed. The principal thought of this enhancement is to perform low-pass (averaging) filtering along the ridge direction to connect the discontinuous line or to perform band-pass filtering along the direction orthogonal to the ridges to separate neighbour ridges where they are too close to each other. In Figure 2.7 filtered image is shown with its original view.

a) FVC2000_DB2_B: 104_1.tif b)

Figure 2.7: Image filtering. a) Original image; b) Filtered image.

Some earlier approaches[22, 23] are proposed which defined a set of bell-shaped filters (16 sub-filters along each 22.5°) to convolute each point of the

image with one from this filter-set which has the best matched direction with the local ridge and in [24] which processed the fingerprint image in the frequency domain using Fourier transform under the help of the local ridge orientation to select the best enhancement.

Subsequently Gabor filters became the most popular approach in contextual filters because of its frequency-selective and orientation-selective properties[25]. After Gabor proposed 1-dimention Gabor-transform in 1946 and Daugman 2-dimention Gabor-transform in 1985, [4] developed a method based on this theory. Such Gabor filters, whose spatial extent, frequency and orientation as well as bandwidths can be configured with the parameters generating themselves, can reach the lower bound of the uncertain relation simultaneously in space and frequency fields[26]. In [4] a Gabor filters is defined in the following form and shown in Figure 2.8.

$$g(x, y: \theta, f) = exp\left(-\frac{1}{2}\left[\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right]\right) \cdot \cos(2\pi f \cdot x_\theta) \tag{2.4}$$

where θ is the orientation of the filter, $[x_\theta, y_\theta]$ are the coordinates of $[x, y]$ after a clockwise rotation of the Cartesian axes by an angle of $(90° - \theta)$, f is the frequency of a sinusoidal plane wave, and σ_x, σ_y are the standard deviations of the Gaussian envelop along the x- and y-axes, respectively.

$$\begin{bmatrix} x_\theta \\ y_\theta \end{bmatrix} = \begin{bmatrix} \cos(90° - \theta) & \sin(90° - \theta) \\ -\sin(90° - \theta) & \cos(90° - \theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sin\theta & \cos\theta \\ -\cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.5}$$

However, when local ridge orientation and frequency are not reliable in the fingerprint image with many bad areas, convolution with Gabor filters will bring side effect because of unsuitable parameters based on this local ridge's information. Some other approaches based on Gabor filters proposed in [19, 27-

29] etc. aim to optimize the above parameters - σ_x, σ_y in order to obtain better tolerate errors or avoid generating spurious ridges and valleys.

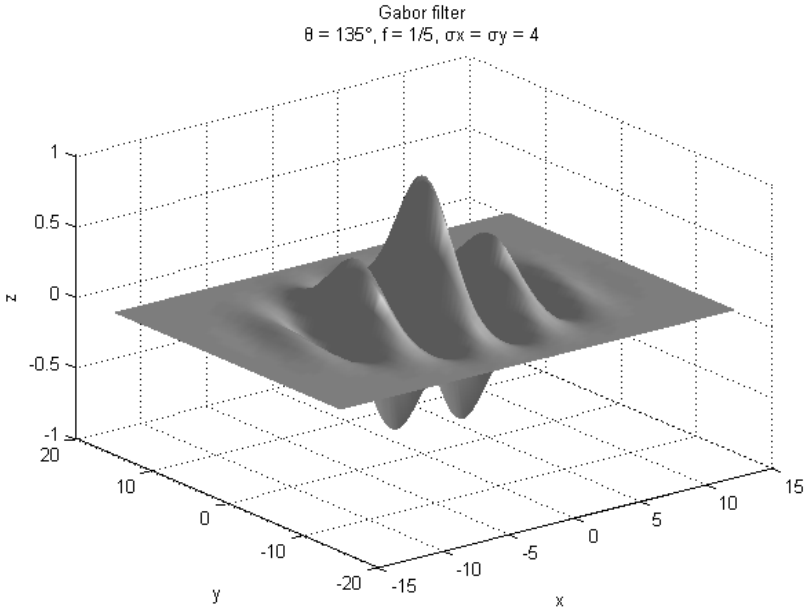


Figure 2.8: Gabor Filter with the parameters $\theta = 135°, f = 1/5, \sigma_x = \sigma_y = 4$.

2.3 Experimental Improvements

Besides some enhancement approaches mentioned above, some other attempts have also been adopted by us in order to obtain a more qualified fingerprint image for minutiae extraction.

2.3.1 Directional Image Filtering

The fingerprint directional image extracts the characteristics of the distribution of ridges and valleys and provides the image's textural structure, with which images can be directionally filtered. To obtain more reliable filtered image, we firstly use the intermediate noise image, to be discussed in next section, to restrain the

direction image to the valid range. Then Gabor filter is used to filter the fingerprint image.

The direction corresponds to horizontal direction that goes to the left and increment of the direction gives counterclockwise turn. But in fact to simplify the computation, each direction and its reverse direction are regarded as the same direction, so that only four main directions (0°, 45°, 90°, 135°.) are counted to represent the direction info at each pixel. The direction layout is shown in Figure 2.9.

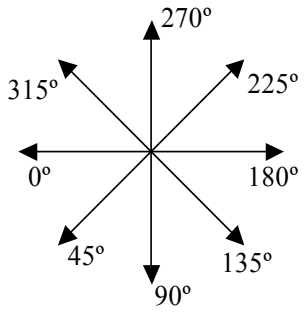


Figure 2.9: Image direction layout.

In directional image computation the sum of pixel values around each pixel along different directions should be firstly obtained within a pre-defined window shown in Figure 2.10.

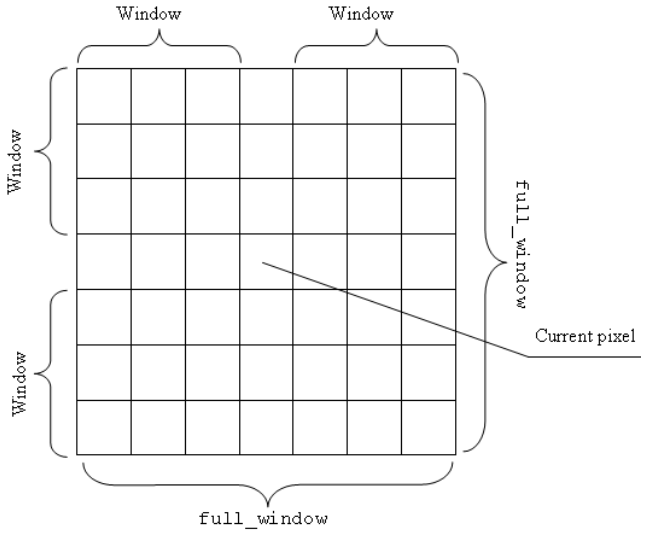


Figure 2.10: Computation window ($w = 12$).

$$S_{ij} = \sum_{y=i-window}^{i+window} \cdot \sum_{x=j-window}^{j+window} \begin{pmatrix} \left|I_{y\ \ x-1} - I_{yx}\right| + \left|I_{y\ \ x+1} - I_{yx}\right| \\ \left|I_{y-1\,x-1} - I_{yx}\right| + \left|I_{y+1\,x+1} - I_{yx}\right| \\ \left|I_{y-1\,x\ \ } - I_{yx}\right| + \left|I_{y+1\,x\ \ } - I_{yx}\right| \\ \left|I_{y-1\,x+1} - I_{yx}\right| + \left|I_{y+1\,x-1} - I_{yx}\right| \end{pmatrix} \tag{2.6}$$

From (2.2) and (2.6) the orientation of each pixel can be computed. In practice this direction image is not quite acceptable because of noise which may disturb the orientation of the local ridges. A low-pass filter which is performed in each 3×3 block is used to smooth this directional image shown in the following (2.7).

$$\begin{aligned} S_x &= \sum_{u=i-1}^{i+1} \sum_{v=j-1}^{j+1} \cos 2\theta(u,v) \\ S_y &= \sum_{u=i-1}^{i+1} \sum_{v=j-1}^{j+1} \sin 2\theta(u,v) \\ \theta(i,j) &= \frac{1}{2} tan^{-1}\left(\frac{S_y}{S_x}\right) \quad if\ \theta(i,j) < 0, \theta(i,j) = \theta(i,j) + \pi \end{aligned} \tag{2.7}$$

According to (2.4), a set of horizontal Gabor filter templates $\left(\omega_{-N/2}, \cdots, \omega_{-1}, \omega_0, \omega_1, \cdots, \omega_{N/2}\right)$ are firstly generated and then rotated to all of the other pre-defined directions, where N is the template size. If the input fingerprint image is defined as $I(x,y)$ and the filtered image is $O(x,y)$, the directional image filtering can be described in (2.8) and (2.9).

$$O(x,y) = \varphi_0 I(x,y) + \sum_{s=1}^{\frac{N}{2}} \varphi_j\big(I(x-s,y-s) + I(x+s,y+s)\big) \tag{2.8}$$

$$O(x,y) = \varphi_0 I(x,y) + \sum_{s=1}^{N/2} \varphi_j (I\left(x - \frac{s}{\mu}, y - s\right) + I\left(x + \frac{s}{\mu}, y + s\right)) \tag{2.9}$$

When the direction is along the x-axis at $[x, y]$, the convolution filtering is implemented with (2.8). Otherwise the input fingerprint image is filtered with (2.9) at $[x, y]$ along $y = \mu x, where\ \mu = tan\theta$. If the coordinates in (2.9) fall between two neighbor pixels, linear interpolation will be used to obtain its approximate coordinates.

FVC2006_DB3_A: 4_3.bmp

FVC2006_DB3_A: 9_12.bmp

a) b) c)

Figure 2.11: Directional image filtering. a) Fingerprint grey-scale images; b) Images before directional image filtering; c) Images after directional image filtering.

It is evident from Figure 2.11 that most undesired noise is removed and the ridges and valleys are preserved and improved.

2.3.2 Noise Image Evaluation

In fingerprint images regular ridges and valleys are regarded as the foreground and the other areas filled with smudge or shadow generated from sensor as the background which is also called noisy regions. To separate noisy regions from the original fingerprint image or other intermediate images is very gainful such as computational complexity and precision. The basic assumption for this segmentation is that in a given block noisy regions have no dominant direction while clear regions lie in a particular direction[30]. Moreover, foreground regions have a very high variance in a direction orthogonal to the orientation of the pattern and a very low variance along its dominant ridge direction.

We obtain the original noise image during directional image computation according to the following assumption. There should be a dominant direction at each foreground pixel. This direction has a nearest neighbor direction and a furthest neighbor direction among the four main directions listed in Figure 2.9. The difference between the gradients along these two directions represents the ridge quality at this pixel. Since they are orthogonal, the direction at this pixel is not evident and the ridge quality is low if they are close and vice versa. The noise image records such difference at each pixel. Afterwards the obtained noise image is optimized using the values at each pixel and a finite 33×33 window around each pixel in order to divide those areas with poor ridge quality into background quickly.

According to the judgment that whether a dominant direction at each pixel exists, two rules are proposed in this thesis to differentiate background and foreground in a noise image:

1) At $[x_i, y_i]$, if the maximal gradient f_{max} in all directions is below a pre-defined threshold f_{thres1}, this point will be set to background.
2) In a 31×31 window centered at $[x_i, y_i]$, if the sum of the pixels with unreliable directions, i.e. their respective quality in noise image is below a pre-defined threshold f_{thres2}, is more than 50% and in addition it is not a singular point, this point will be reset to background.

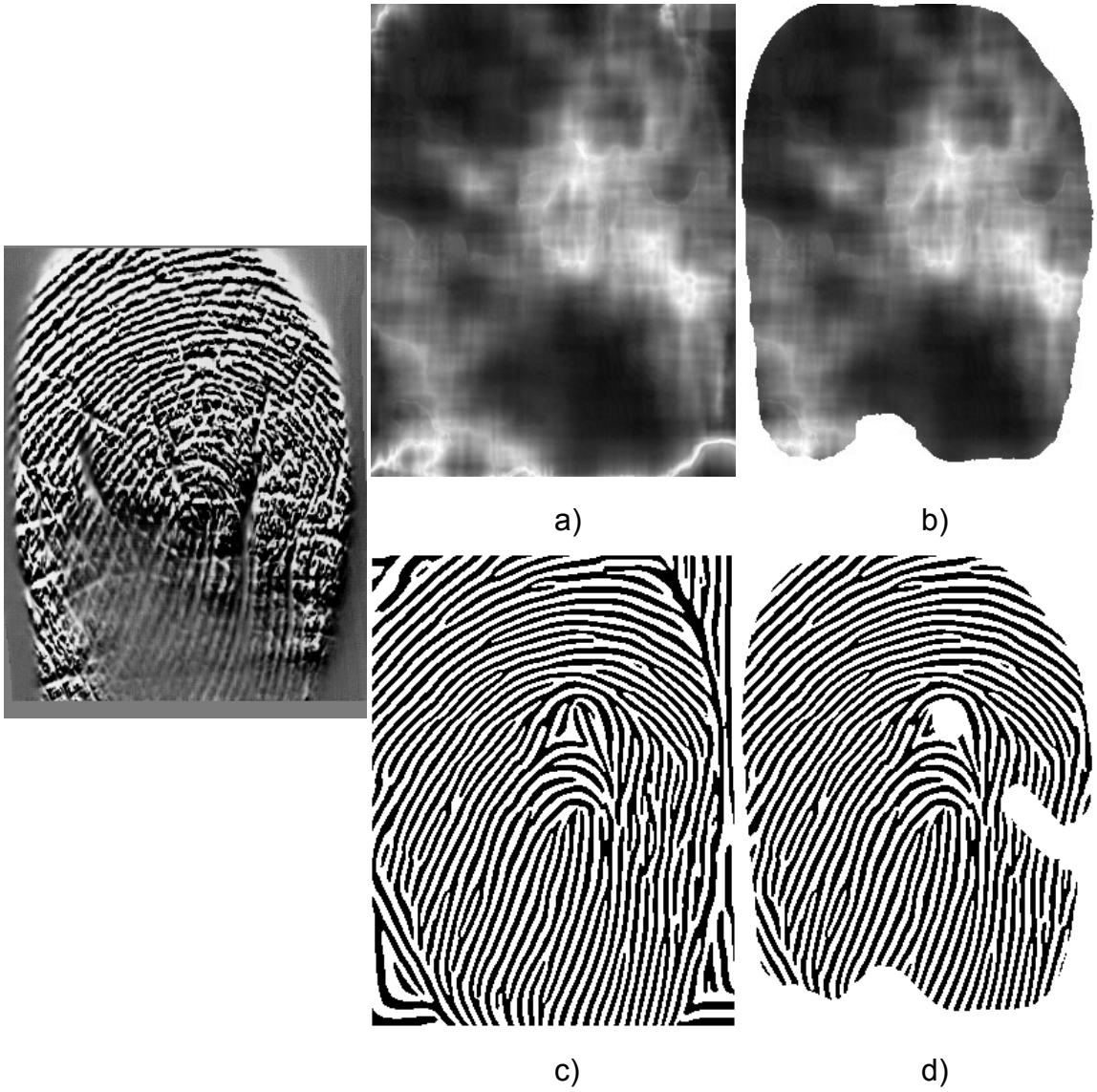
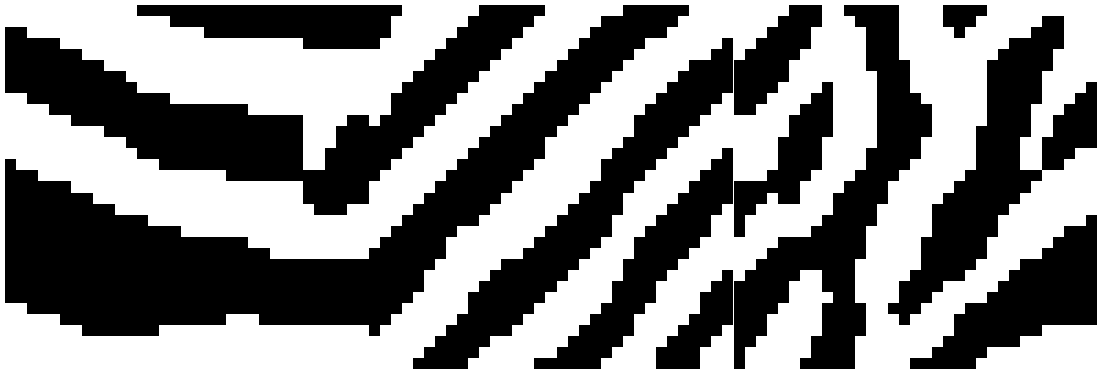


Figure 2.12: Noise image evaluation. a) Original noise image; b) Optimized noise image; c) Original binary image; d) Optimized binary image.

If $[x_i, y_i]$ in a noise image is marked as the background, this pixel is considered as the unreliable position. The following operation such as grey-image filtering, binarization and minutiae extraction will ignore such area (Figure 2.12).

2.3.3 Frequency Image Improvement

The fingerprint ridge distributes with special orientation feature. Particularly perpendicular to the local ridge orientation, the grey values alter with the period of approximate to some kind of sine wave[31]. Besides the indication of the period of ideal parallel ridges, the local ridge frequency (density) image provides the ridge's filling rate inside one window. Figure 2.13 shows three 33×33 windows in a pre-binarized image with different local density. To define the difference of these windows can help for the selection of the threshold for binarization for the reason that if the density is too high, a relative small binarization window should be selected to avoid interacting between two neighboring ridges and if the density is too low, a relative big binarization window can include more ridge orientation information.



a) Low b) Middle c) High

Figure 2.13: 33×33 windows with different local frequencies.

We propose a method to find this local ridge frequency by accumulating the edge pixels of the ridges inside this window. The frequency f_{ij} is computed as follows.

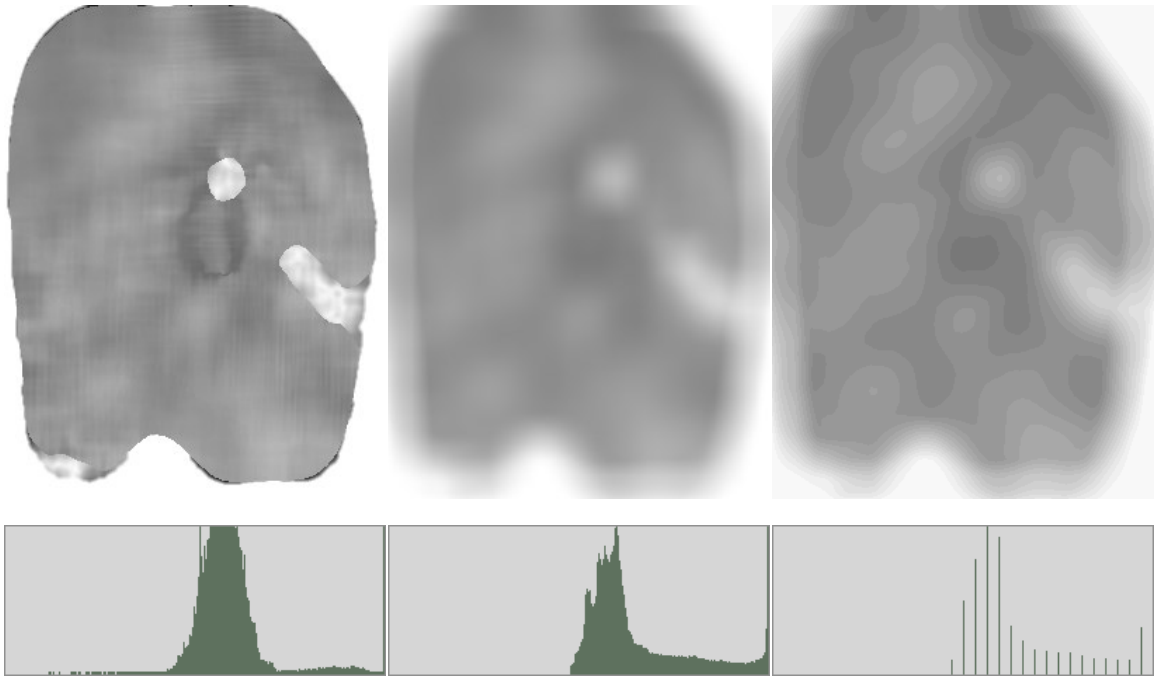
1) A filtered input image is firstly pre-binarized with the simple algorithm:

$$
\begin{aligned}
&if \quad O(x,y) \geq \left(\frac{1}{(2m+1)^2} \sum_{y=-m}^{m} \sum_{x=-m}^{m} O(x,y) + \frac{1}{(2n+1)^2} \sum_{y=-n}^{n} \sum_{x=-n}^{n} O(x,y) \right) / 2 \\
&\qquad B(x,y) = 255 \\
&else \\
&\qquad B(x,y) = 0
\end{aligned}
\tag{2.10}
$$

where $O(x,y)$ is the filtered input image, $B(x,y)$ is the pre-binarized image and m, n is the window size.

2) A 36×36 window centered at $[x_i, y_i]$ is defined in horizontal-vertical coordinate system in this pre-binarized image.
3) Another 3×3 window is scanned from left-top to right-bottom in the above 36×36 window. If the sum of the grey values in this 3×3 window $S_{3\times3}$ is between S_{min} and S_{max}, the window counter $nS_{3\times3}$ will be accumulated.
4) If $[x_i, y_i]$ in noise image is not in background field, $nS_{3\times3}$ from step 2 will be limited to $[0 \cdots 255]$ and then projected to horizontal or vertical axis using directional image.
5) A local mean filter is performed after frequency estimation in step 3 to reduce noise.
6) The filtered frequency image is then quantized with a finite quantization step (in this thesis step = 8) if $[x_i, y_i]$ is not a singular point.

Since frequency image describes the density of the ridges, the binarization presented in the next chapter will be more effectively performed with this information. Some intermediate results for frequency image improvement are shown in Figure 2.14.



a) b) c)

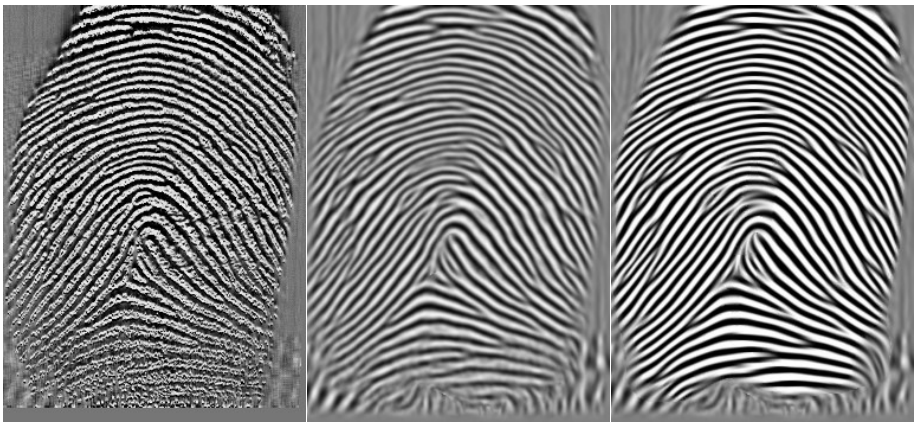
Figure 2.14: Frequency image improvement. a) Original frequency image; b) Filtered frequency image; c) Quantized frequency image.

2.4 Summary

Some earlier work in fingerprint enhancement is introduced in this chapter, which includes general-purpose image processing techniques such as segmentation, normalization, image smoothing and also special-purpose method for fingerprint such as contextual filtering, ridge frequency estimation. Based on the previous work, we have optimized the fingerprint enhancement algorithm and the new algorithm will be more robust to noise. Figure 2.15 presents some enhancement results using our optimized algorithm. Images are from FVC2006 DB3 with the resolution 500 dpi and the size 400×500.

FVC2006_D3_A: 1_3.bmp

FVC2006_D3_A: 6_5.bmp



FVC2006_D3_A: 124_2.bmp

a) b) c)

Figure 2.15: Results of fingerprint image enhancement with optimized algorithm: a) Input image; b) Directional filtered image; c) Directional filtered image with ridge estimation.

We note that our algorithm can find and repair most of the cuts, dryness but for those extreme unrecoverable regions such as the creases in the image of the first row not very well. These regions have to be marked and not used for the next fingerprint feature extraction.

Chapter 3

Fingerprint Feature Extraction

3.1 Introduction

Fingerprint features include minutiae, singularity (see Figure 3.1) and local ridge alignment etc. Minutiae, which exist where the ridges become somehow discontinuous, constitute the unique characteristic of one fingerprint. According to the structure of this discontinuity, minutiae can be classified into terminations and bifurcations (see Figure 3.2)[32]. The x- and y- coordinates and the angle between the tangent to the ridge line at the minutia position and the horizontal axis.

2_12.bmp 3_6.bmp 6_3.bmp 62_11.bmp 80_10.bmp

Figure 3.1: Various singularities of fingerprint in FVC2006_DB3_A.

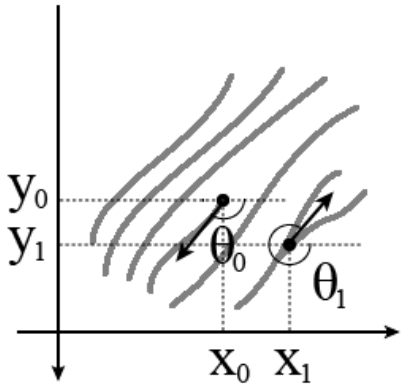


Figure 3.2: Termination minutia (x_0, y_0, θ_0) and bifurcation minutia (x_1, y_1, θ_1).

a) (FVC2006_DB3_A: 31_12.bmp)

b) (FVC2006_DB3_A: 4_6.bmp)

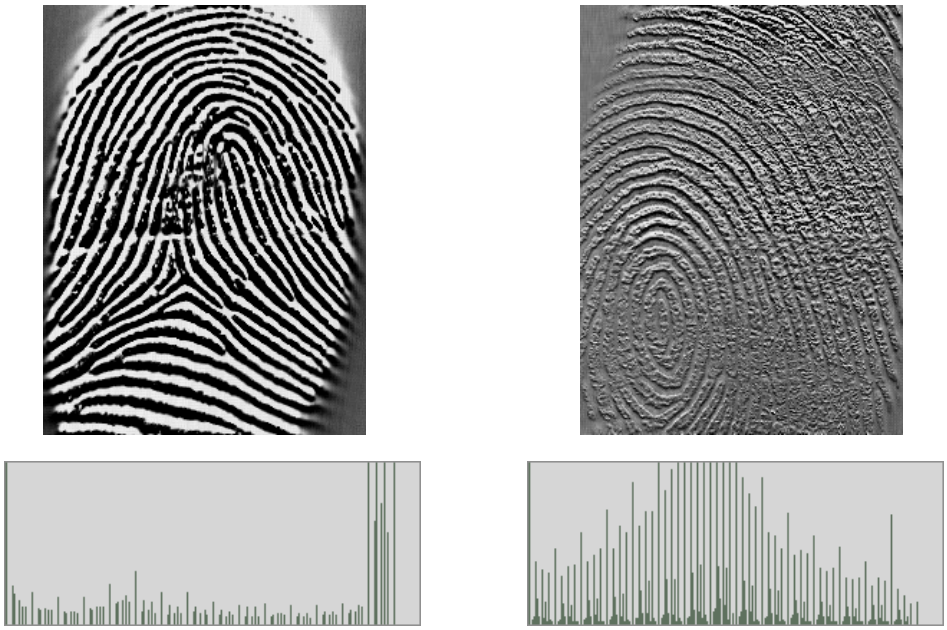
Figure 3.3: Efficiency of binarization-based method for images with different quality and acquisition condition: a) A medium quality fingerprint with scratches and ridge breaks; b) A poor quality fingerprint with too much acquisition noise.

Most of the past research on fingerprint feature extraction is based on minutiae, which is still the principle criterion in fingerprint comparison. Therefore, to obtain the credible minutiae is the key to fingerprint matching. Various approaches have been proposed in minutiae detection, which can be concluded into two types: grey-scale based and binarization-based. Grey-scaled based method aims to extract minutiae directly from grey-scale images where image information keeps unchanged and more reliable especially for low-quality images. However binarization-based method has to process enhanced even skeletonized image. The advantage of this method is that the background noise because of smudges or acquisition and the discontinuity of the ridges because of bruises and cuts can be better improved (see Figure 3.3).

3.2 State of the Art

3.2.1 Binarization

The purpose of the binarization is to convert grey-value fingerprint image to binary-value image with the pixel value of “0” and “255”. Thus the geometric characteristic of the image is only concerned with the distribution of “0” and “255”, which simplifies the latter processes. One simple approach is to set a global threshold t, with which the pixels whose grey-level is lower than t are set to 0 and the others to 255. When a fingerprint image has good contrast and the statistic histogram can acquire two apparent peaks (see Figure 3.4.a), this approach is feasible. Nevertheless it is very sensible to noises. Moreover, this threshold is not easy to decide and maybe inexistent for the reason that the contrast and intensity in local areas of one image may have great difference (see Figure 3.4.b). From the point of this view, local-threshold-based approach has been widely used only to improve the quality of the binarization image.



a) (FVC2006_DB3_A:48_3.bmp) b) (FVC2006_DB3_A: 15_3.bmp)

Figure 3.4: Fingerprint images with different contrast: a) Appropriate for global threshold; b) Appropriate for local threshold.

A binarization approach using Laplacian operator and dynamic thresholds was proposed in [33, 34]. The image is convolved with a Laplacian operator and the intensity of each pixel is computed. If this intensity value is not between the two pre-defined thresholds, the relative pixel will be set to 0 or 255 according to the above comparison.

An approach which uses the edges information around each ridge combined with the grey-scale image was designed in [35]. An edge image is firstly computed and then the fingerprint ridges are tracked separately in grey-scale image and edge image. Finally these two tracked image will be combined with the OR operation.

Another one based on peak detection from grey-scaled image is from [7]. After the computation of the orientation field, the input image is divided into 16×16 pixel blocks. For each block the maximum variance of the grey level projection

appears along the orientation perpendicular to the local ridge orientation. Thus the peaks are obtained with the variance and ridges are located and thinned along the same orientation with an adaptive morphological filter. Meanwhile the peaks and its two neighbouring pixels are set to 0 and the remaining to 255.

One topological approach proposed in [36] considers a fingerprint image as continuous surface of noise, which is generated from Chebyshev polynomials. From the curvatures of this surface ridges and valleys areas can be determined with the help of its second-order derivatives.

Some other documents related to fingerprint image binarization can be found in [4, 24, 30, 37, 38].

3.2.2 Thinning

a) FVC2000_DB2_B: 104_1.tif b)

Figure 3.5: Binarized image thinning. a) Binarized image; b) Thinning image.

Thinning, also called skeletonization, aims to extract one-pixel-width ridge structure from binarized image which is composed of multi-pixel-width ridges and

valleys (see Figure 3.5). A qualified thinning method should preserve the profile and connectivity of the ridges, the distribution of the minutiae and locate each skeleton in the centre of its relational ridge. Some research has been done in [7, 39-46] by filling holes, eliminating small breaks, deleting bridges between ridges and smudges. For example, in [45, 46] some arithmetic-logic operations are used in elimination and relative smooth skeletons can be obtained, however [45] loses some ridge structure and [46] generates much deformation for those slanting ridges.

The approach OPTA (One-pass Parallel Thinning Algorithm) proposed in [44] is a popular method for thinning. It uses eight 3×3 thinning patterns (see Figure 3.8) to remove edges pixels and two restoring patterns (1×4 and 4×1) (see Figure 3.7) to preserve continuity. For one binarized fingerprint image, the pixel of background will be set to 0 and foreground to 1. This algorithm starts from the left-top of the image, extracts neighbour pixels around each foreground element (see Figure 3.6, P5). These neighbour pixels are then compared with the eight thinning patterns. If one from the eight thinning patterns coincides with these neighbour pixels, this pixel will be marked as “deleted”, otherwise as “restored”. In view of the continuity, this pixel will also be compared with the other two restoring patterns. If the second row in Figure 3.6 matches with the second row of a) in Figure 3.7 or the second column in Figure 3.6 matches with the second column of b) in Figure 3.7, this pixel will be marked as “restored”, otherwise as “deleted”. However, this algorithm raises such a problem that it generates biased skeletons and is more easily to remove convex corners than concave corners [47].

P_1	P_2	P_3	×
P_4	P_5	P_6	P_{14}
P_7	P_8	P_9	×
×	P_{11}	×	×

Figure 3.6: Neighbour of P5 in OPTA.

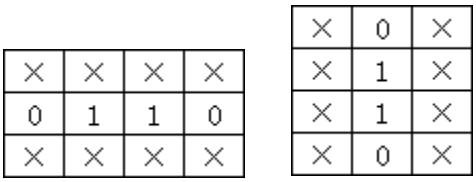


Figure 3.7: Restoring patterns in OPTA.

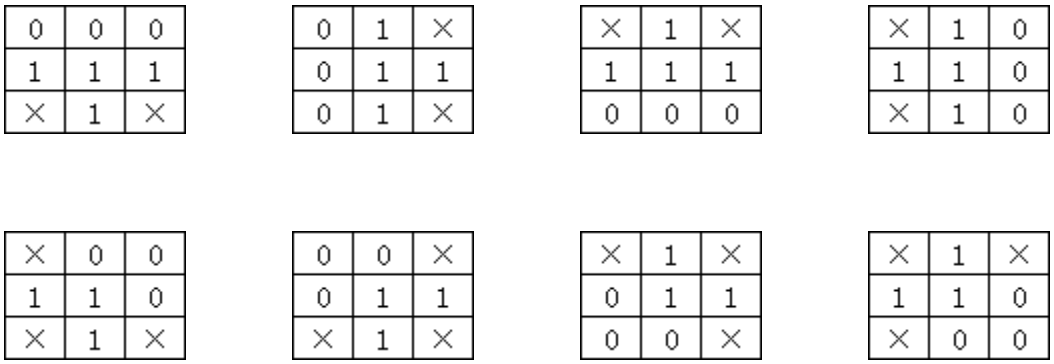


Figure 3.8: Thinning patterns in OPTA.

In [48, 49] this algorithm is optimized by using more reliable restoring and thinning patterns. These new patterns can remove edge pixels including convex corner pixels, but not very helpful to those concave corner pixels.

3.2.3 Minutiae Extraction

Many different minutiae extraction methods have been proposed in the past years (see Figure 3.9). And the effective detection approaches can be on the whole concluded into the following two categories based on image domain for detection. The first group extracts minutiae using binarized images. The second group extracts minutiae directly from grey-scale images [7], and ignores the above section 3.2.1 and 3.2.2 mentioned binarization and thinning processes.

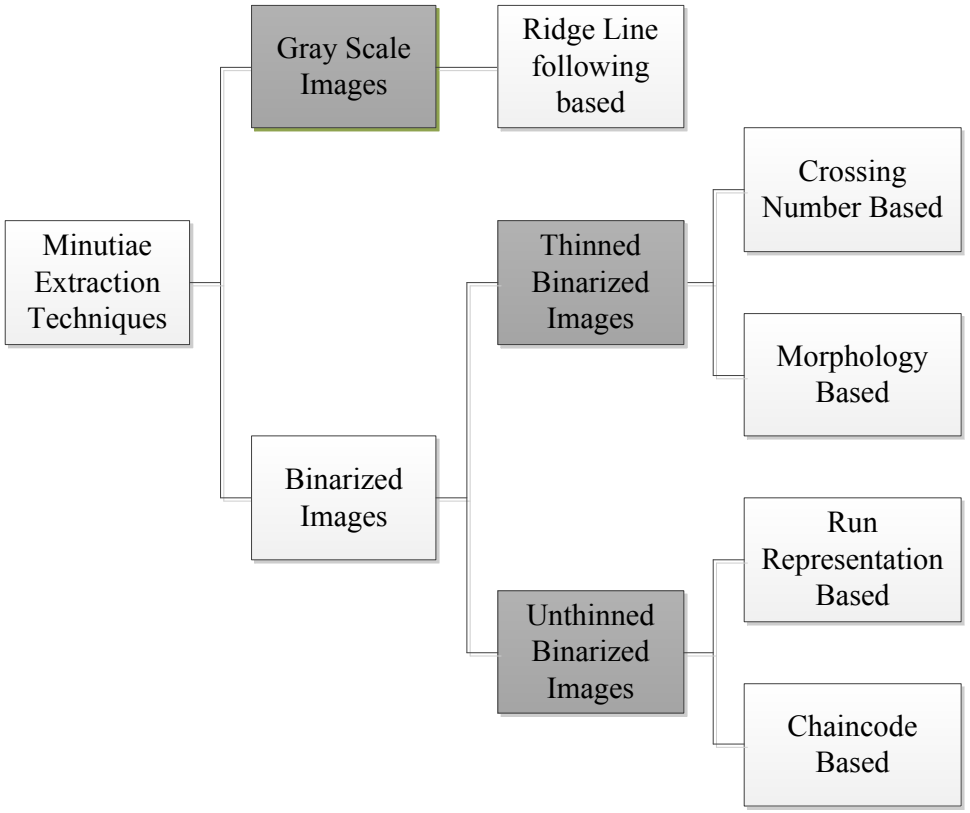


Figure 3.9: Classifications of minutiae extraction techniques.

3.2.3.1 Extraction with Unthinned Binarized Images

One extraction algorithm depended on unthinned binarized images makes use of the closed contours of the ridges in fingerprint images [17, 50, 51]. These ridge contours will be consistently traced in a counter clockwise fashion and marked with a special grey value except 0 (foreground) and 255 (background). When we arrive at one point where exists a sharp turn, we mark this point as a candidate for a ridge-ending or bifurcation point.

A binary image (0 – foreground, 255 – background) is scanned from left-top to right-down row by row. In order to avoid data overflow and keep all ridges closed, pixels at the outermost rows and columns will be omitted in scanning. The scanning procedure is described as follows:

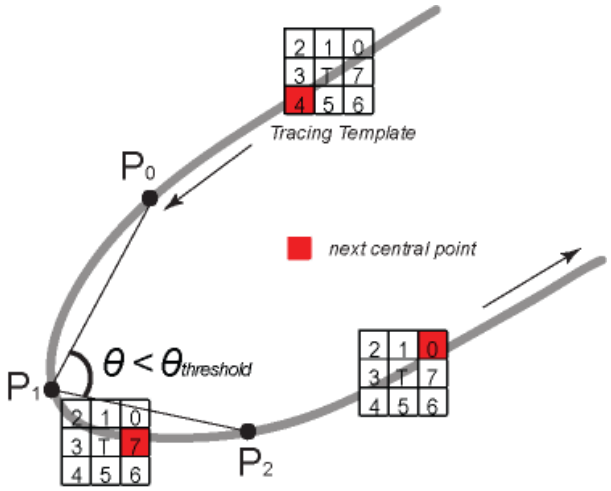


Figure 3.10: Minutiae extraction using contour tracing in binary image.

1) If the first scanned foreground pixel at $[x_i, y_i]$ is not inside one ridge field, this pixel will be set to the starting point of one contour tracing.
2) A 3×3 window is defined where the central point is T, see Figure 3.10.
3) Set $[x_i, y_i]$ as the coordinate of the central point T, and trace the pixels around T from index 0 to 7. The first non-background pixel will be assigned as the next neighbouring central point, whose value will also be modified to a special value (e.g. 253).
4) From the new neighbouring central point step 3 will be repeated until the next central point reaches the starting point at step 1 where a complete ridge contour is obtained.
5) Along the computed contour of the ridge three points P_0, P_1 and P_2 with the same step length are defined in order to compute the local minimum angle θ. P_1 will be marked as a candidate of the expected minutiae when the following conditions are met:

$$S = (x_0 - x_1) \times (x_2 - x_1) \times (y_0 - y_1) \times (y_2 - y_1)$$
$$S > S_{threshold} \tag{3.1}$$

6) From all above candidates the pixel $P_{max}$with S_{max} is selected as the minutiae position. And the minutiae angle can be extracted from the vector $\vec{P}_{max}$ in (3.2):

$$\vec{P}_{max} = \overrightarrow{P_1P_0} + \overrightarrow{P_1P_2} \tag{3.2}$$

Another effective algorithm utilizes the horizontal and vertical run-length encoding from binary images instead of a computationally expensive thinning process[52, 53]. After run-length encoding Fingerprint images are represented as a cascade of runs. In this method, fingerprint images should also be pre-processed: using Gabor filters to enhance the ridges oriented in the direction of local orientation at each pixel and to reduce noise with parameters for the orientation and the frequency; and using adaptive image binarization with an optimal threshold. After the binary image is generated, a run-length encoding of this binary image is executed horizontally to define successive black pixels as a run. Each run has its starting location and ending location or running length. Moreover each run has its special case as follows:

1) No adjacent runs both on previous and the next scan line.
2) Two adjacent runs on previous and the next scan line.
3) One adjacent runs on either the previous or the next scan line.
4) Two adjacent runs on either the previous or the next scan line.
5) More than two adjacent runs on either the previous or the next scan line.

From the case of each run, the characteristics of these successive black pixels can be concluded as 1) one pixel spot or one pixel width line; 2) ridgeline; 3) ridge termination, i.e. the starting or ending point of one ridge; 4) ridge bifurcation; 5) not yet considered according to the features of ridge's distribution.

3.2.3.2 Skeletonization-based Methods

Skeletonization-based methods require not only the procedures described in last section such as image enhancement and binarization, but only one additional

process – thinning. The thinning algorithm reduces pixels along the ridges up to one pixel width (see section 3.2.2). In this category two principal approaches have been widely utilized to process such thinned binary images, which are crossing number based and morphology based.

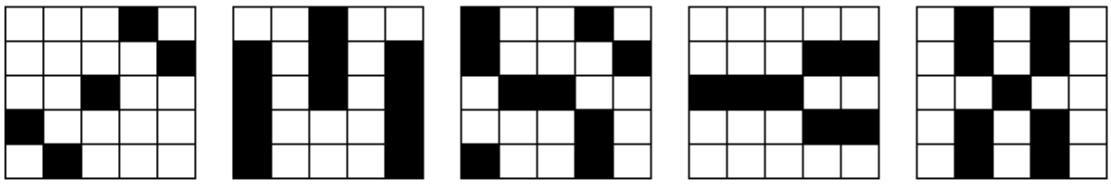
The Crossing Number (CN) method is a popular skeletonization-based method. In the literature [34, 54-59] this method is used for minutiae extraction. The advantage of this method lies in its computational efficiency and theoretical simplicity. Since the ridge flow pattern in the skeleton image is 8-distance, scanning can be made in the local neighbourhood of each ridge pixel with a 3×3 window (see Figure 3.11), through which CN value is computed as follows:

$$CN(p) = \frac{1}{2}\sum_{i=1}^{8} |P_i - P_{i-1}| \quad where\ P_0 = P_8 \tag{3.3}$$

P_8	P_7	P_6
P_1	P	P_5
P_2	P_3	P_4

Figure 3.11: 3×3 window for Crossing Number scanning.

Half the sum of the difference between two adjacent pixels in the 8-distance neighbourhood is counted, for the reason that each pixel has to be used twice in (3.3). From the value of the CN, the ridge pixel can be classified as a ridge ending, bifurcation or an intermediate ridge point (see Figure 3.12).



CN	Type
0	Isolated point
1	Ridge termination point
2	Intermediate ridge point
3	Bifurcation point
4	Crossover point

Figure 3.12: Types of Crossing Number.

Another commonly used technique is based on mathematical morphology. Some special morphological operators are used[60, 61] to remove isolated islands, spurs, spurious bridges and holes and then to extract true minutiae with Hit or Miss Transformation.

$$A \otimes B = (A \ominus B_1) \cap (A^c \ominus B_2) \tag{3.4}$$

In (3.4) A is the binarized fingerprint image, while B is a structuring element pair $B = (B_1, B_2)$. The foreground pixels can be located by erosion with B_1 and the background pixels by erosion of the complement with B_2. As mentioned in [61], ridge endings have the property that there is only one foreground pixel in their 3×3 neighbour, whose element pair is composed of the elements (i)-(viii) and (ix)-(xvi) in Figure 3.13. And ridge bifurcations have the property that there are three nonadjacent foreground pixels in their 3×3 neighbour, whose element pair is composed of the elements (i)-(viii) and (ix)-(xvi) in Figure 3.14.

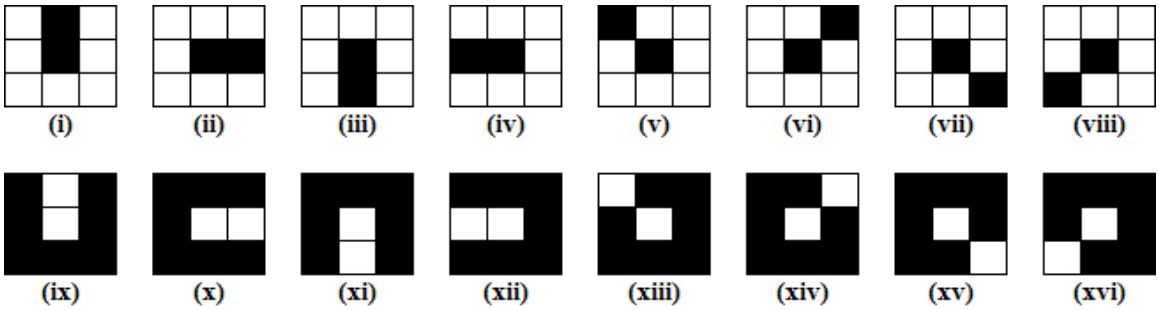


Figure 3.13: The structure element sequence for ridge endings.

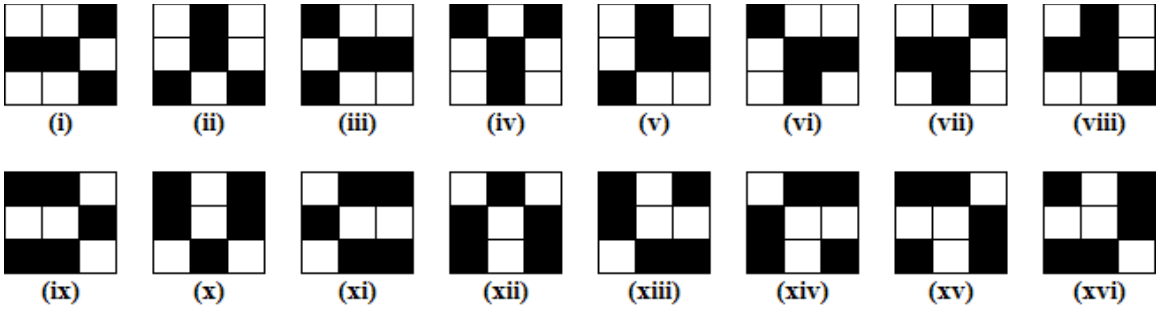


Figure 3.14: The structure element sequence for ridge bifurcations.

3.2.3.3 Extraction with Grey-Level Images

Some recent researches have shown that minutiae extraction directly from grey-level images without binarization and thinning can obtain better results in some database especially with low quality images. Some other considerations for this idea are that binarization and thinning bring degradation of image details and time consuming, furthermore a fair number of spurious minutiae will be generated during these processes.

Ridge lines are the most prominent feature in a grey-level fingerprint image, which are composed of a set of pixels with local maxima along one direction. Based on this idea, [8, 62] extract the minutiae directly from the grey-level image by following the ridge flow lines according to their local orientation field. From a pre-defined starting point with its definite direction, this algorithm calculates the ending point by moving given step from the current point along the local direction.

Then the median point with its median direction between the starting point and the ending point is computed as the next starting point. The process is iterated until all ridge lines are followed and can meanwhile be interrupted when a ridge line terminates or intersects another ridge line (minutiae point). Thence when all ridge lines are scanned, minutiae with its type, coordinates and direction are also obtained.

The above method was improved by using dynamical step[63] instead of given step. The step length is decided by the change of the ridge contrast. If the contrast (intensity variations) is small, a relative large step will be used, otherwise a small step will be used. Another optimized method[64] is proposed, which tracks not only the central ridge line but also its two adjacent valleys simultaneously. At each step, both local maxima and its two parallel local minima are located. Minutiae should be extracted where the relation of these three extreme values changes. Linear Symmetry filter[65, 66] was also used to extract minutiae directly from grey-level images for the reason that minutiae leads to the local discontinuities from the view of the symmetry filter. A local window is built to compute filter response. If the response is greater than pre-defined threshold, minutiae will be located.

3.3 Application and Optimization

3.3.1 Grey-Level Image Binarization

Among the approaches mentioned in 3.2.1, direct gray-scale extraction method depends on the quality of fingerprint image and cannot present the correct trend of ridges in poor quality image especially when image has much noise. Another direction-image-based extraction method utilizes the direction at each pixel to supply an amendment for the accumulations of the gray value in a window for the decision of background or foreground. This method can deal with most parts of one fingerprint image, whereas it gains not good results in such areas with not

enough or irregular direction information. To solve the above problems, an optimized noise image (see 2.3.2) is firstly used to mark all invalid noise area as background, and all further operations are processed only in other valid area. Moreover the frequency of directional filtering and the window size of adaptive-local-threshold for binarization also change according to the filling rate inside one window, i.e. ridge density, to reconcile the binarization result and computation speed. Through multiple directional filtering in such areas with uncertain direction information excluding background area, the binarization in these areas is remarkably improved by reconnecting broken ridges or separating two adjacent parallel ridges. The approach we propose performs binarization using not only grey-level and orientation image but also noise and frequency image. Furthermore the morphological dilation operators are also used to improve the binarization of non-fingerprint-area and fingerprint centre area where might produce some unexpected holes etc. The details of various procedures of the proposed algorithm are described as follows and in Figure 3.15.

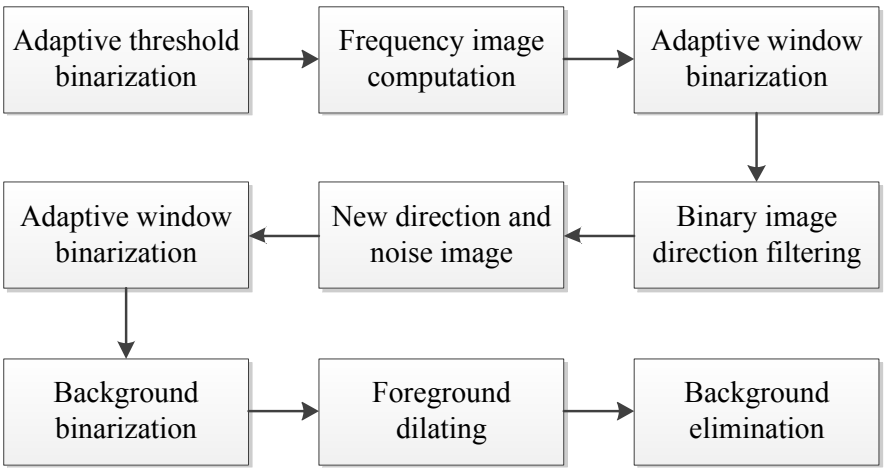


Figure 3.15: Stages of the proposed grey-level image binarization algorithm.

1) Define windows $(w_1 = 7, w_2 = 15)$ in image G_1, the mean value of all pixels in windows can be computed:

$$T_1 = \frac{\sum_{u=i-\frac{w_1}{2}}^{i+\frac{w_1}{2}} \sum_{v=j-\frac{w_1}{2}}^{j+\frac{w_1}{2}} G_1(u,v)}{w_1 \times w_1}$$

$$T_2 = \frac{\sum_{u=i-\frac{w_2}{2}}^{i+\frac{w_2}{2}} \sum_{v=j-\frac{w_2}{2}}^{j+\frac{w_2}{2}} G_1(u,v)}{w_2 \times w_2}$$

$$if \quad G_1(u,v) \geq \frac{(T_1 + T_2)}{2}$$

$$G_2(u,v) = 255$$

$$else$$

$$G_2(u,v) = 0$$

(3.5)

If the pixel $G_1(u,v)$ is a noise point in the orientation image, it will not be computed in this step.

2) Define windows $(w_1 = 3, w_2 = 36)$ in image G_2, and compute the ridge frequency image G_3 according to the proportion of window w_1, when it's not all 0 or 255 in the window w_1 (see 2.3.3).
3) Smooth image G_3 with window $(w_1 = 33)$.

$$G_4(u,v) = \frac{\sum_{u=i-\frac{w_1}{2}}^{i+\frac{w_1}{2}} \sum_{v=j-\frac{w_1}{2}}^{j+\frac{w_1}{2}} G_3(u,v)}{w_1 \times w_1}$$

(3.6)

4) Using the smoothed frequency image G_4 the pre-filtered fingerprint image can be binarized again in windows of different size $(w_1 = 7, w_2 = 11, w_3 = 15)$.

$$\begin{array}{ll} if & G_4(u,v) \geq thres_1 \quad w = w_1 \\ if & thres_1 \geq G_4(u,v) \geq thres_2 \quad w = w_2 \\ if & thres_2 \geq G_4(u,v) \quad w = w_3 \end{array}$$

(3.7)

In this window w, $G_5(u,v)$ is obtained as step 1.

5) Directional image filtering on image G_5. With the algorithm in 2.3.1 a binarized-directional-filtered image G_6 is computed.

6) From the binarization image G_6, the directional image G_7 and frequency image G_8 is recomputed and updated.
7) The binarization image G_6 is selectively filtered to image G_9 with new directional image in those areas where more noise exists.
8) With the new directional image and images G_4, G_6, the optimized binarization image G_{10} is computed as step 4.
9) Define windows $(w_1 = 3)$ in image G_{10}, and compute the sum of the grey values in each window and the binarization image G_{11} as follows (edge pixels are not included here and set to 255):

$$S(u,v) = \sum_{u=i-\frac{w_1}{2}}^{i+\frac{w_1}{2}} \sum_{v=j-\frac{w_1}{2}}^{j+\frac{w_1}{2}} G_{10}(u,v)$$

$$if \quad S(u,v) < w_1 \times w_1 \times 128$$

$$G_{11}(u,v) = 0$$

$$else$$

$$G_{11}(u,v) = 255 \tag{3.8}$$

10) At each foreground pixel in image G_{11} a 4-distance area is built and dilated along its horizontal and vertical directions until ending condition satisfies. When the central pixel and the pixel at one of its 4-distance position are background, dilation algorithm will set it to foreground and make a record at this pixel and locate it as the next starting point. When the ending condition satisfies i.e. the sum of the records reaches a pre-defined threshold, all of these pixels will be set back to background otherwise these pixels will be dilated to foreground (compare delta area in image G_{11} and dilated image G_{12} in Figure 3.16).
11) In the directional image G_7 pixels with uncertain direction have been specially marked. With this information the noise areas in dilated image G_{12} can be easily removed (set pixel value to background 255 in noise areas) and final binarization image G_{13} is generated.

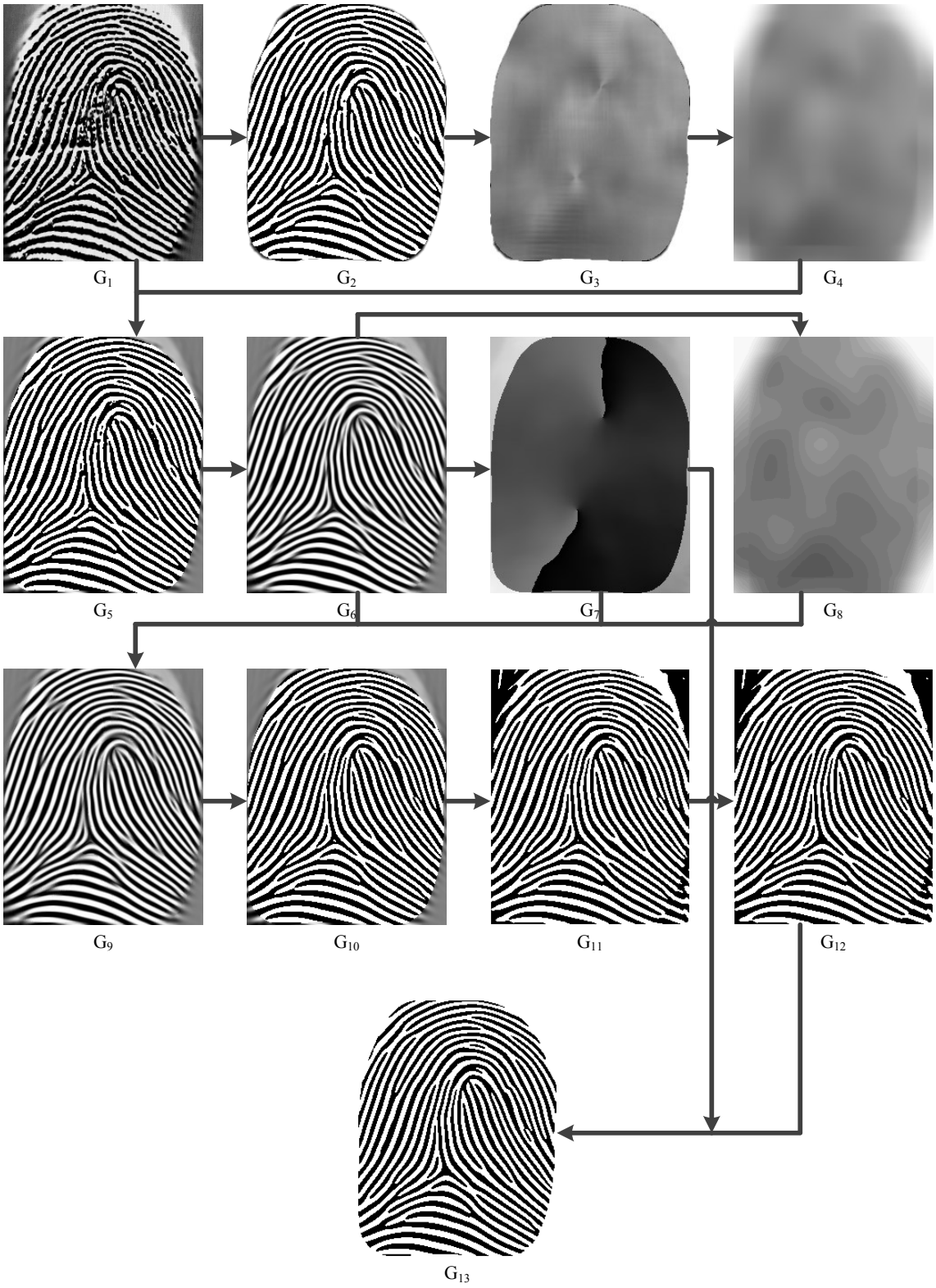


Figure 3.16: Results of various stages in grey-level image binarization.

3.3.2 Minutiae Extraction

As described in previous chapters minutiae extraction can make use of original grey-level image, contour traced binary image or skeletonized binary image. Among these approaches using a binarized and skeletonized image is the most popular one for minutiae extraction and thinning is the important step of minutiae extraction in this method.

As discussed in 3.2.2, the approach in [44] generates many biased skeletons, which has been improved in [48] by using some new patterns listed in Figure 3.17 b) instead of patterns in 3.17 a). Nevertheless the thinning patterns in Figure 3.17 a) are not suitable for the image thinning process, our experiment has shown that they are still helpful to amend the thinned image and make the skeleton corners smoother instead. Based on this idea, we have implemented the existing thinning patterns and given some improvements with our proposed thinning patterns subsequently, which are shown in Figure 3.17 c).

The thinning process can be concluded that when a pixel and its neighbour are examined and this central pixel is flagged as a boundary pixel, both the flag and the result image in this iteration will be updated for the next iteration process.

The proposed thinning algorithm is described as follows:

1) For each pixel and its neighbours decide whether it matches one of the thinning patterns in Figure 3.18 a). If it matches, make a flag.
2) Delete all flags and update the binary image at each flagged location as a background pixel.
3) Repeat step 1 and step 2 until no pixel can be deleted or the maximal repetition number reaches.
4) For each pixel and its neighbours decide whether it matches one of the thinning patterns in Figure 3.18 b). If it matches, update it as a background pixel.

Step 1 to step 3 aims to trim the ridges to one-pixel wide with the criteria that no end pixel is deleted; no connectedness is violated; no excessive erosion occurs. Step 4 tries to repair such convex corner pixels along a thinned ridge line in order to make this corner smoother.

The thinned skeleton image of fingerprint is then scanned and all possible minutiae such as ridge endings and ridge bifurcations are detected using the method of Cross Number described in earlier section.

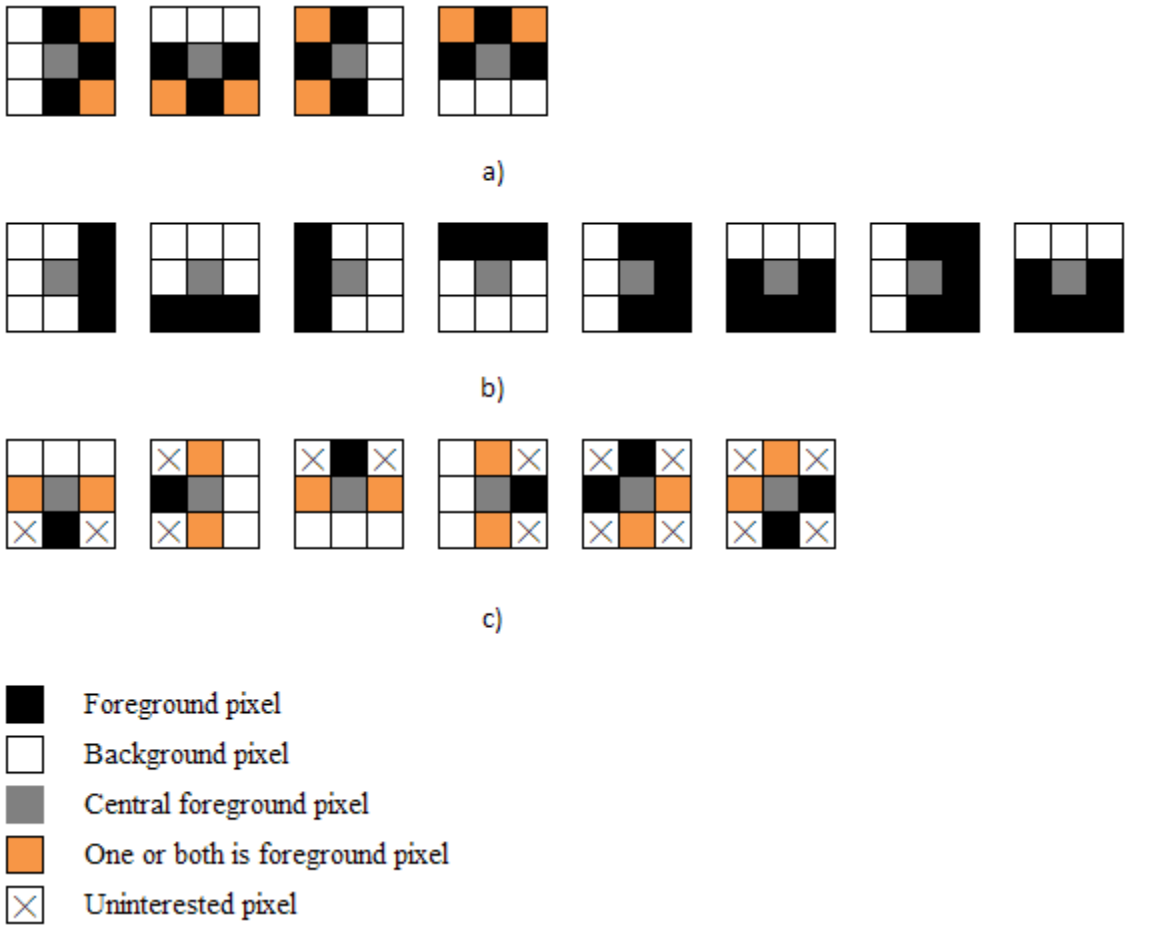
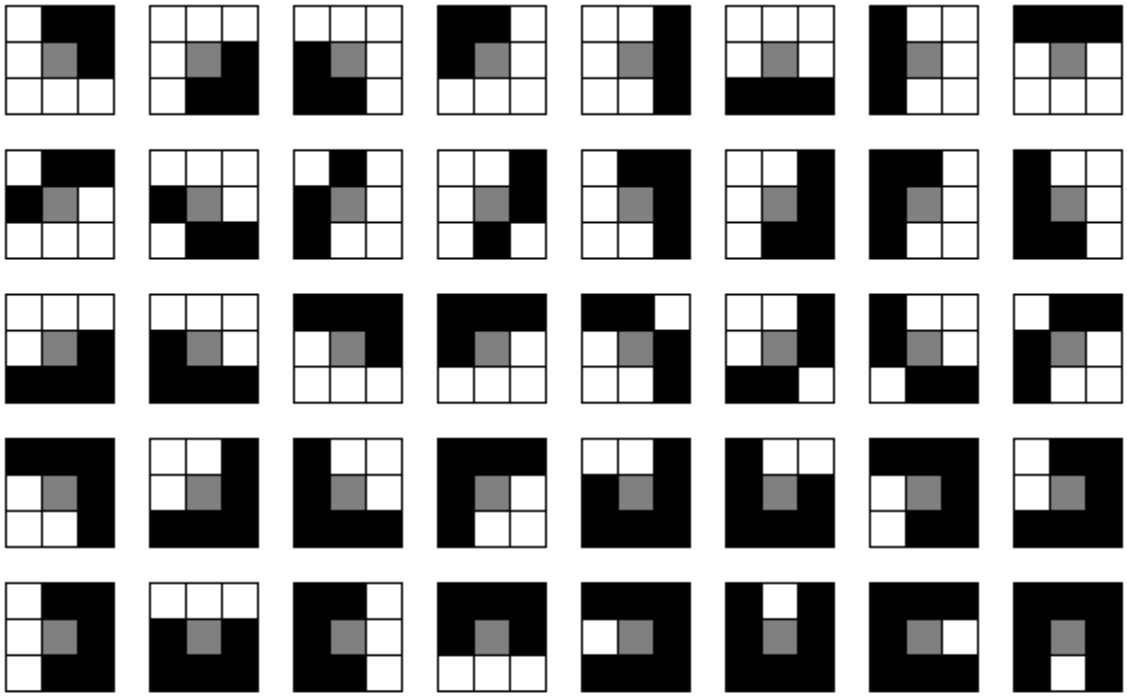
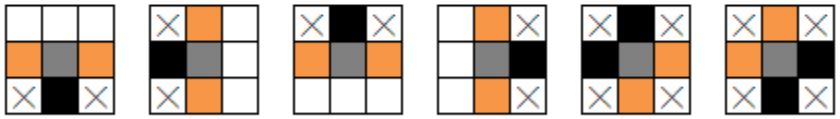


Figure 3.17: Thinning patterns improvements.



a) Basic thinning patterns



b) Special thinning patterns

Figure 3.18: The proposed thinning patterns.

3.3.3 Minutiae Post-processing

For those fingerprint images of poor quality, many spurious minutiae can be extracted due to noise. Consequently, a post-processing algorithm is designed to optimize the result of minutiae extraction. The proposed post-processing algorithm makes use of the extracted minutiae information such as coordinates (x_i, y_i), direction θ_i, and distance between two minutiae and score s_i of each minutiae (see Figure 3.19) as well as directional image G_1 and direction variance image G_2.

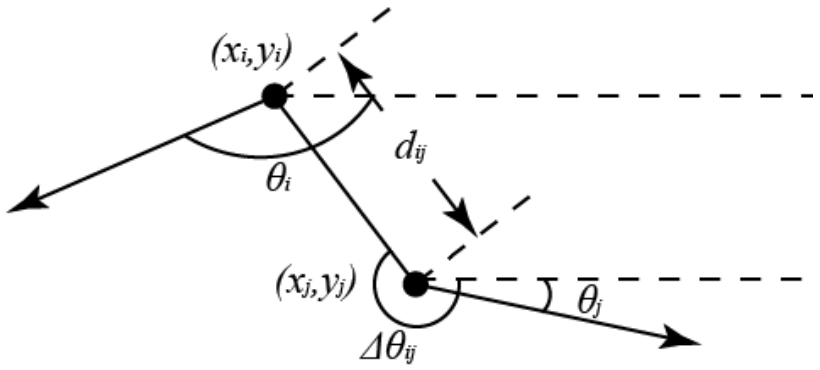


Figure 3.19: One minutia pair for post-processing.

The proposed minutiae post-processing rules are described as follows:

1) If two minutiae are close and have quite different direction, they are considered as from a broken ridge and should be deleted.

$$\begin{cases} d_{ij} < D_{thres1} \\ \left|\theta_i - \Delta\theta_{ij}\right| > \theta_{thres1} \\ \left|\theta_j - \Delta\theta_{ij}\right| > \theta_{thres1} \end{cases} \tag{3.9}$$

2) If the number of one minutia's neighbours within a definite distance exceeds the threshold T_1, it should be deleted.

$$T_{|d_{ij}<D_{thres2}|} > T_{thres1} \tag{3.10}$$

3) If one minutia has one or more neighbours with closer distance, both or all of them should be deleted.

$$d_{ij(i\neq j)} < D_{thres3} \tag{3.11}$$

4) If one minutia is too close to the image borders, it should be deleted.

$$d_{i,borders} < D_{thres4}$$

(3.12)

5) If one pixel is close enough to one minutia but it has no reliable direction in the directional image, this minutia is not reliable as well and should be deleted.

$$\begin{cases} d_{i,(x,y)} < D_{thres5} \\ G_1(x,y)\ is\ noise \end{cases}$$

(3.13)

6) If one minutia is too close to one singularity, it should be deleted.

$$d_{i,singularity} < D_{thres6}$$

(3.14)

7) If two close minutiae have similar direction and not ideal minutia scores, they should both be deleted.

$$\begin{cases} d_{ij} < D_{thres7} \\ \left|\theta_i - \theta_j\right| < \theta_{thres2} \\ s_i < S_{thres1} \\ s_j < S_{thres1} \end{cases}$$

(3.15)

8) If one minutia and its several neighbouring minutiae have worse scores even though they are further in distance, it should be deleted when such neighbours exceed the threshold T_2.

$$\begin{cases} T_{|d_{ij}<D_{thres8}|} < T_{thres2} \\ s_i < S_{thres2} \\ s_j < S_{thres2} \end{cases}$$

(3.16)

9) All other undeleted minutiae are kept according to the distance between the minutiae and the singular region or the coordinates in the original image.

3.4 Experimental Improvements

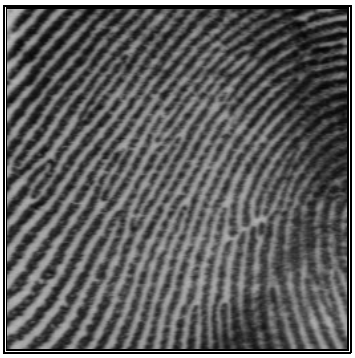
To achieve better minutiae extraction representation, we have improved the classic grey-level image binarization approach described in 3.3.1. Figure 3.20 shows the various results using different binarization methods aiming at filling holes, removing small gaps and other artifacts produced by the noise.

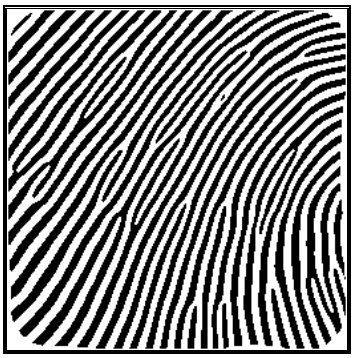
In evaluation of the minutiae extraction algorithm we make a 1:1 comparison between the proposed algorithm and the other existing algorithms in [8] (A), [67] (B), [33] (C), [68] (D), [23] (E). The dataset is also the same as what is used in [8]: 7 fingerprints (no. 3, 4, 5, 11, 12, 13 and 14 in Figure 3.21) taken from the NIST fingerprint database[69], 4 fingerprints (no. 1, 2, 9 and 10 in Figure 3.20) from an FBI sample set, and 3 fingerprints (no. 6, 7 and 8 in Figure 3.21) acquired through an opto-electronic device based on a prism.

The approach A uses ridge-line following technique directly on grey-scale fingerprint image. The approaches B, C, D and E use the binarization and thinning method similar to the proposed in this thesis. Figure 3.22 and Figure 3.23 show two fingerprint images, which are also used for comparison in [8], and their minutiae extraction process using our approach and the approaches A, B and E.

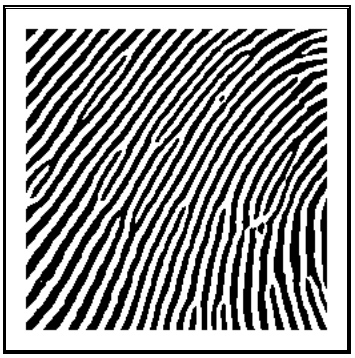
Table 3.1 gives the results of the number of undetected minutiae (dropped), non-existent minutiae (false) and type-exchanged minutiae (exchanged) from the algorithms listed above. Table 3.2, 3.3 and 3.4 give the results of the average error percentage from the classes - good, - poor and from the whole sample set. The average error percentage of total error is computed as follows[70]:

$$AEP = \frac{1}{N} \times \left(\sum_{i}^{N} \frac{drop(i) + false(i) + exchanged(i)}{total} \right) \times 100 \tag{3.17}$$

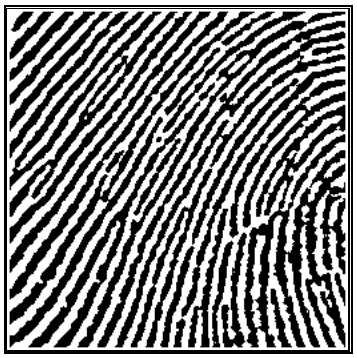




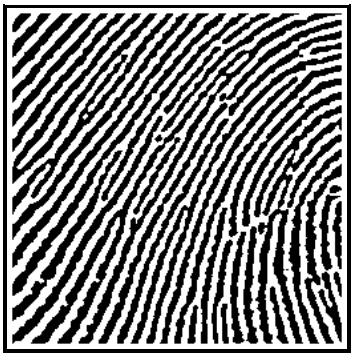
1 Proposed



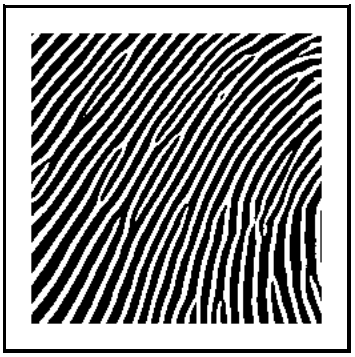
2 [23]



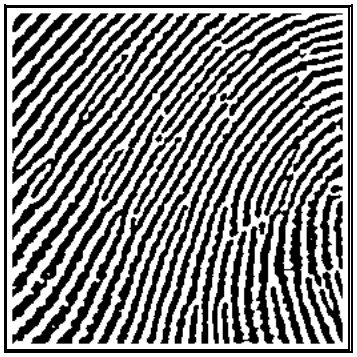
3 [68]



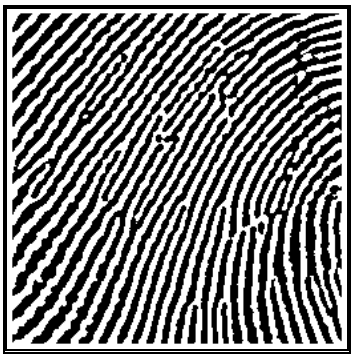
4 Local threshold



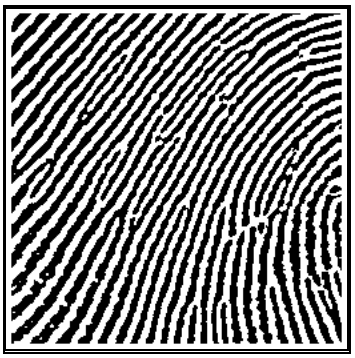
5 [19]



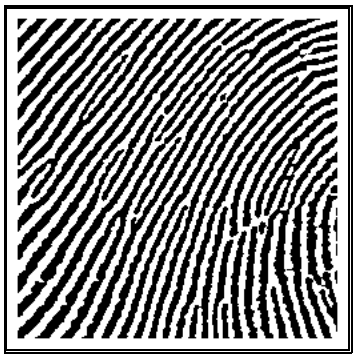
6 [71]



7 [33]



8 [67]



9 [37]

Figure 3.20: Comparison of binarization from a fingerprint of good quality with various methods.

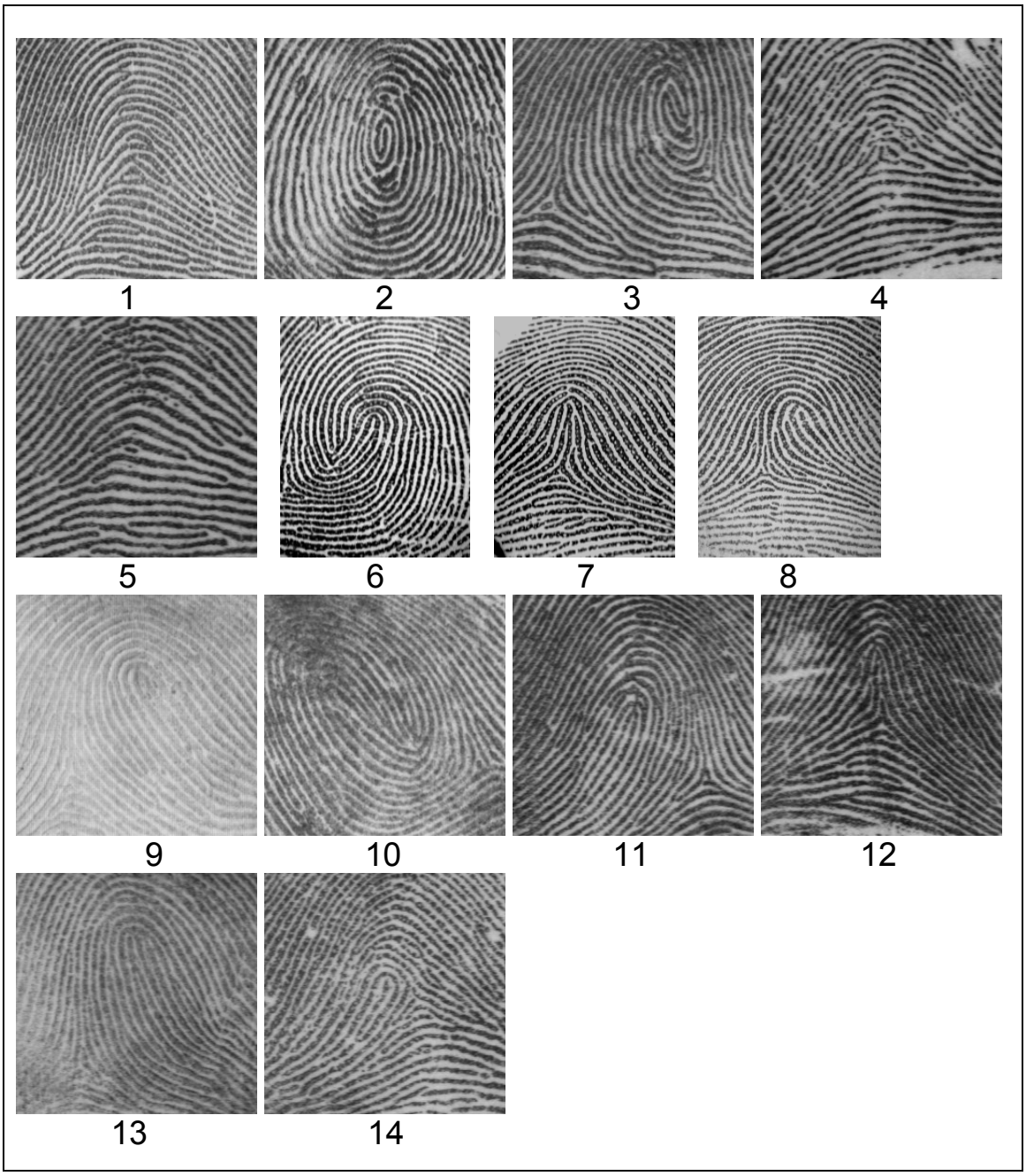


Figure 3.21: The sample set of fingerprints referred in [8] and used in our comparison. All the fingerprints (except no. 6, 7 and 8) have dimensions 256×256 with 256 grey levels. The fingerprints no. 6, 7 and 8 have dimensions 190×250 with 256 grey levels.

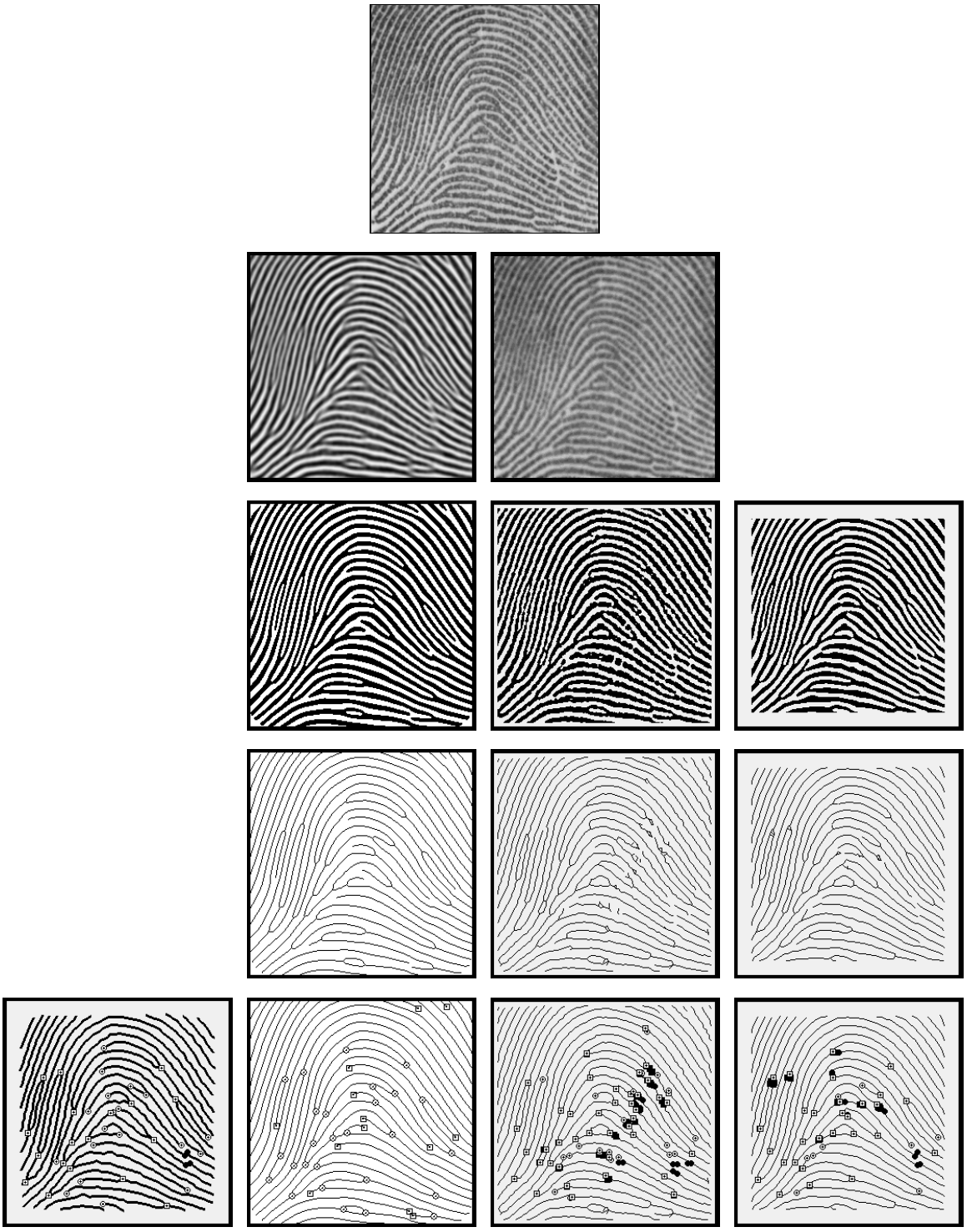


Figure 3.22: Automatic minutiae detection in fingerprint no.1 - class good using approach A, the proposed, B and E (from left to right). The first row is the input image successively with its smoothing, binarization, thinning and minutiae extraction results.

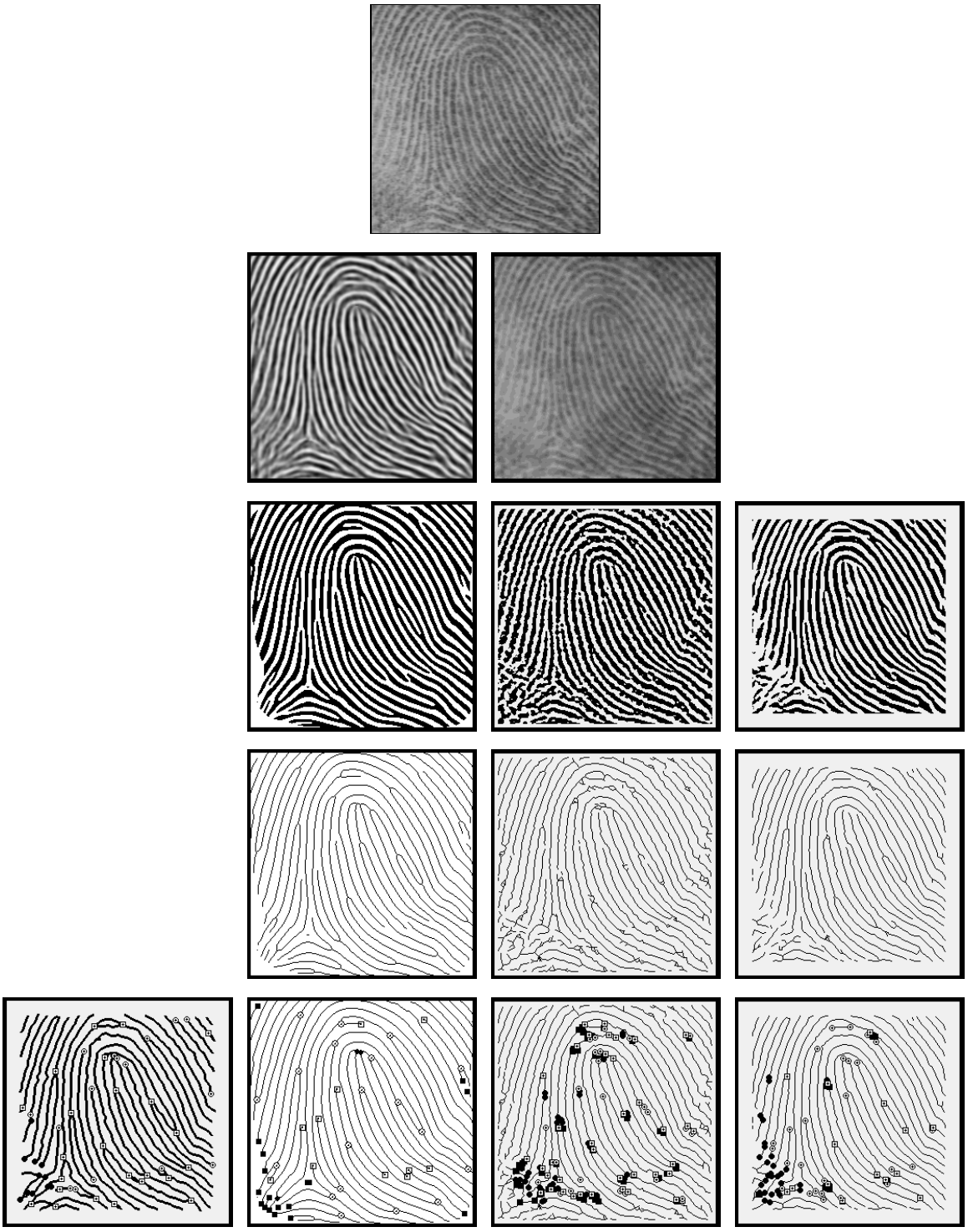


Figure 3.23: Automatic minutiae detection in fingerprint no.13 - class poor using approach A, the proposed, B and E (from left to right). The first row is the input image successively with its smoothing, binarization, thinning and minutiae extraction results.

		Proposed			A			B			C			D			E		
fingerprint	minutiae	d	f	x	d	f	x	d	f	x	d	f	x	d	f	x	d	f	x
1	33	0	0	5	0	2	7	0	25	2	1	102	7	0	53	6	0	5	4
2	29	0	4	3	3	1	4	0	20	2	2	24	1	1	34	2	2	4	0
3	28	0	1	4	1	2	4	0	24	0	1	35	1	1	28	2	0	4	4
4	37	2	2	0	3	0	4	0	15	3	4	32	0	1	42	3	2	2	2
5	22	0	2	5	0	0	3	0	38	1	0	80	2	1	18	0	0	8	2
6	23	0	0	3	0	0	4	0	3	2	0	25	3	0	19	2	0	2	1
7	31	0	1	6	2	1	2	3	13	2	2	27	2	0	40	3	2	0	2
8	31	1	0	4	1	0	3	0	2	0	1	10	3	0	5	4	0	0	3
9	21	0	10	3	1	10	1	1	115	1	0	180	1	0	153	1	0	24	2
10	22	0	4	4	1	0	4	0	53	1	0	104	3	0	100	1	0	8	4
11	32	0	2	3	3	5	4	1	22	4	2	22	3	3	73	4	1	5	2
12	33	2	3	2	3	8	2	0	23	3	4	45	1	1	79	3	0	10	5
13	20	0	3	2	0	0	4	0	48	2	0	57	3	0	81	2	0	7	5
14	37	0	0	6	0	5	6	1	43	5	3	67	5	1	57	4	0	11	2

Table 3.1: Minutiae extraction comparison

The first column is the fingerprint index. Fingerprints 1...8 belong to the class good; fingerprints 9...14 belong to the class poor. The second column is the minutiae number detected manually. The d, f and x denote the number of dropped minutiae, false minutiae and exchanged minutiae, respectively [8].

good						
	Proposed	A	B	C	D	E
dropped minutiae	1.48%	4.27%	1.28%	4.70%	1.71%	2.56%
false minutiae	4.93%	2.56%	59.83%	143.16%	102.14%	10.68%
exchanged minutiae	14.78%	13.25%	5.13%	8.12%	9.40%	7.69%
total error	21.18%	20.09%	66.24%	155.98%	113.25%	20.94%

Table 3.2: Average error percentage relative to the fingerprint of class good.

poor						
	Proposed	A	B	C	D	E
dropped minutiae	1.21%	4.85%	1.82%	5.45%	3.03%	0.61%
false minutiae	13.33%	16.97%	184.24%	287.88%	329.09%	39.39%
exchanged minutiae	8.48	12.73%	9.70%	9.70%	9.09%	12.12%
total error	23.03%	34.55%	195.76%	303.03%	341.21%	52.12%

Table 3.3: Average error percentage relative to the fingerprint of class poor.

whole set						
	Proposed	A	B	C	D	E
dropped minutiae	1.36%	4.51%	1.50%	5.01%	2.26%	1.75%
false minutiae	8.70%	8.52%	111.28%	203.01%	195.99%	22.56%
exchanged minutiae	11.96%	13.03%	7.02%	8.77%	9.27%	9.52%
total error	22.01%	26.07%	119.80%	216.79%	207.52%	33.83%

Table 3.4: Average error percentage relative to the fingerprint of the whole sample set.

The results from Table 3.1 to Table 3.4 show that the proposed method is comparable to other mentioned methods. The AEPs of the dropped minutiae and false minutiae are much remarkable for the reason that they play important roles in fingerprint matching and have direct effect on the matching result. Considering these two factors our method is the best one in the above list. On the other hand we also notice that the AEP of exchanged minutiae is not quite satisfying because of some excessive directional filtering by pre-processing. Moreover in the computation of noise image, the four corners of the fingerprint image are regarded as noise region, which causes the loss of minutiae in this area.

3.5 Summary

Our fingerprint minutiae extraction algorithm is introduced in this chapter. When the direction field is computed and the grey-level image is directionally filtered, this binarization-based algorithm generates a binarized image from the input image and uses our optimized thinning patterns to obtain a skeleton image for minutiae extraction. After all the possible minutiae are extracted, false minutiae are excluded using the proposed deletion criteria. A comparison between our algorithm and some other existing algorithms shows that this algorithm is comparable and acceptable.

Chapter 4

Fingerprint Matching

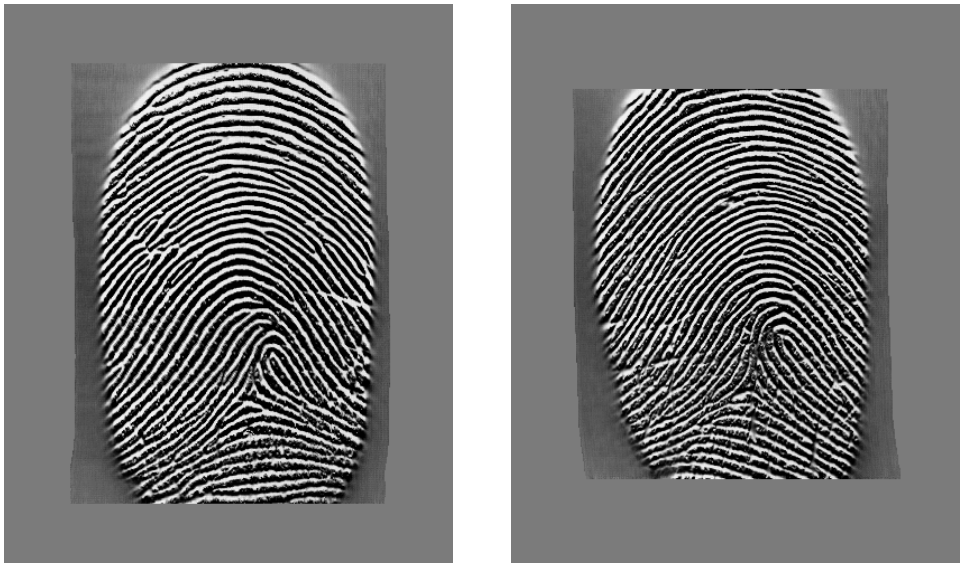
4.1 Introduction

In most Automated Fingerprint Identification Systems (AFIS), fingerprint matching is the crucial step, which compares two fingerprints and gives the similarity between them. According to this similarity one matching score is computed to decide whether these two fingerprints are from the same finger.

Some fingerprint matching algorithms use only the greyscale fingerprint images for the reason that without pre-processing the greyscale fingerprint images retain reliable original feature information. However more other algorithms generate templates from binarized images and matching is processed with two prepared templates. A template contains the info of the coordinates, angles and qualities of the minutiae and ridges in a fingerprint image. It also includes some auxiliary info such as core type, ridge frequency and block directional image etc.

In practice fingerprint matching is quite troublesome due to some unexpected factors. (1) Nonlinear distortion by fingerprint image acquisitions is directionally random and difficult to reconstruct mathematically. By acquisition the three-dimensional elastic surface of a finger is pressed onto a flat sensor interface.

a) FVC2006 DB3_A: 5_4.bmp, 81_9.bmp



b) FVC2006 DB3_A: 122_9.bmp, 140_7.bmp

Figure 4.1: Image in a) and b) are from two different fingers in FVC2006 DB3_A respectively, which are falsely matched in our preceding matching algorithm.

This 3D-2D conversion introduces nonlinear distortion, especially when the power is not orthogonally applied to the sensor[72]. (2) The translation and rotation in successive enrolments from the same finger give these images different appearance. (3) There might be only a partial overlap between the two

fingerprint images because of the translation and rotation. In our system a kind of small solid-state senor is used and this problem becomes much evident. (4) Finger conditions such as dryness, wetness, and injury often result in false contact and minutiae info. (5) The imperfect fingerprint algorithms also generate false minutiae, omit the true minutiae, and bring some deviation in coordinates' location computation. (6) According to some matching algorithms, a few of fingerprint images from different fingers are still difficult to distinguish due to their high similarity (Figure 4.1).

In recent years, the challenge of fingerprint matching algorithms is not the optimization in matching good quality fingerprint images, but the robustness in matching poor quality fingerprint images[73]. On the one hand fingerprint acquisition devices are not intelligent enough to dispose of fingers with different conditions such as dryness, wetness and especially distortion, on the other such original poor fingerprint images with cuts, bruises and gaps on ridges or parallel ridge intercepts are still the main sources of false match and no-match errors.

Fingerprint matching algorithms can be mainly classified into the following types [73]: Correlation-based matching – the fingerprint images to be matched are superimposed and the corresponding pixels are computed to retrieve the variation of the translations and rotations and compared with this info. Minutiae-based matching – mainly based on two kinds of minutiae – ridge ending and ridge bifurcation. Through analysing the similarity of the coordinates and rotation angles of these matched minutiae pairings, a global matching is obtained. Non-Minutiae feature-based matching makes use of some other features (e.g. local orientation and frequency, ridge shape, texture information) to differentiate fingerprint images. Based on not enough minutiae info, minutiae-based matching algorithm cannot mostly make reliable decision. This kind of approach helps especially in these databases which have small image size and few minutiae. Besides these methods there are some other matching techniques such as graph-based, genetic-based, hybrid feature-based and etc.

The rest of this chapter is organized as follows. The next section introduces the main existing methods of fingerprint matching mentioned above. And then our proposed matching algorithm will be described in detail. Performance analysis and experiment results are discussed subsequently. The databases that are used in this chapter are mostly from FVC2004_DB3 and FVC2006_DB3, because the main research in this thesis is focused on fingerprint images from ATMEL sensor.

4.2 State of the Art

Correlation-based approach makes use of the invariance of the rotation and translation of a fingerprint image. This approach uses the classical pre-processing techniques to extract the minutiae, ridges and the orientation field of the input fingerprint image[74]. If $I^{(\Delta x,\Delta y,\theta)}$ is defined as a rotated image from original image I with the rotation angle θ and the translation $(\Delta x, \Delta y)$, the similarity between these two images T and I can be defined [73] as

$$S(T,I) = \max_{\Delta x,\Delta y,\beta} CC(T, I^{(\Delta x,\Delta y,\beta)}) \tag{4.1}$$

Where $CC\left(T, I^{(\Delta x,\Delta y,\beta)}\right) = T^T I$ is the cross-correlation between T and I. Through computing the maximization of (4.1) the optimal rotation angle θ and the translation $(\Delta x, \Delta y)$ can be obtained. Since each acquisition has its particular condition as mentioned in last section, these factors such as finger pressure, image quality can lead most of the matching to false result in practice. In [75] instead of global correlation local or block-wise correlation is used to improve a non-linear distortion for the reason that such elastic distortion does not alter fingerprint pattern in local area significantly, however its accumulation differs the distributions of ridges between two fingerprint images. In [76, 77] some more sophisticated correlation measures such as normalized cross-correlation and

differential correlation are applied to optimize the results in view of the influence from the brightness, contrast and ridge thickness of a input fingerprint image. Furthermore to reduce the computational complexity local correlation and global correlation are also applied in Fourier domain which is described in [78, 79].

Minutiae-based approach plays the most important role in fingerprint matching field and is most widely used for fingerprint matching especially as a proof of identity in the courts of law. Similar to the correlation-based matching in this approach minutiae are certainly extracted from fingerprint images and stored in templates. Each minutia in a template consists of at least its coordinates and direction, and in addition maybe also its type, quality and neighbour minutiae info, which can be used to weight the matching score from the coordinates and direction of the minutiae pairs themselves.

Hough transform-based approach is one of the most popular minutiae-based methods. It is firstly proposed in [80, 81] to convert point pattern matching to the problem of detecting peaks in the Hough space of transformation parameters 73. The parameters - transformations of angles and coordinates of minutiae - are discretized and accumulated. According to Hough transform theory the most consistent transformation of these parameters gains the most overlap in their Hough spaces each and thus the optimal parameters are obtained. In [82] Hough transform-based minutiae matching approach is proposed with one additional parameter – scale besides translation and rotation. Another alternative approach in [83] determines the optimal parameter with the maximum Matching Pair Support, which is computed using the difference of the angle and scale of the possible minutiae pairs from the two templates each.

Another minutiae-based approach makes use of not only common minutiae but also some special pre-aligned characters associated with some minutia. Singularities such as core, delta or other core types can be stored in template to identify two different fingerprints. In [16, 31] the orientation of the core or around

the core and the orientations of the singularities are individually used as a measurement to differentiate two different fingerprints. In [84] the pre-aligned ridges, where the minutiae reside, are used to assist minutiae matching. By iterative matching these ridges between two templates, the best rotation angle and translation coordinates can be obtained. However this kind of approach has relative high computational complexity, furthermore the biggest problem of it is that how reliable are those singularities and ridges since there are still many false minutiae in a to-be-matched template or even in the reference template in database.

One more effective minutiae-based approach is local minutiae matching, which consists of a central minutia and its several neighbouring minutiae within a predefined distance. Since the relative angle and the distance between the central minutia and its one neighbour is absolute and independent of the image rotation and the translation and even distortion to some extent, this approach offers great assistance to the global minutiae matching mentioned above when a minutia pair cannot be confidently determined. In [55] the neighbours of the central minutiae are filtered and its N spatially nearest minutiae are reserved in distance order. When N is small, this approach is very efficient. In [85] this approach is developed that the N spatially nearest minutiae are sorted not by their distance to the central minutiae but their angles due to the reason that if two of its neighbours have similar distance to the central minutiae, the sequence of the final neighbours could be false. The approach mentioned in [86] selects the nearest neighbour minutiae within a fixed radius. In addition, local matching is also implemented using minutiae triangles [87-89]. The triangle features such as the length of each side, the ridge count between each pair of vertices, type, direction or even the average deviation of the local orientations in the neighbourhood of each triangle vertex are used for measuring the similarity of the triangles between two templates. Another local matching approach proposed in [90] gathers sampling points around each minutia along a set of circles with different radii. The orientation information at these points is recorded and the

average angular difference between corresponding orientations is used to weigh the similarity.

Non-Minutiae feature-based approaches include the utilization of the features such as texture information[91], spatial relationship of the ridge lines [86, 92], sweat pores and etc. For those small size sensors it is difficult to generate enough minutiae, moreover in low quality fingerprint images most of the extracted minutiae are not reliable enough. This kind of approaches has just an edge under such situations. In [91] the fingerprint area of interest is set around the core point and divided into several neighbouring sectors. The local texture information in each sector is decomposed into separate channels with a Gabor filter bank. Through gaining the local texture information and the global relationship among local sectors each fingerprint is encoded as a vector called Finger Code. This approach acts as assistance to minutiae-based matching. In [86, 92] the spatial relationship of ridges is used to verify how exact a minutiae pair is. In this approach the relative ridges are also matched to compensate the final matching score generated from a minutiae pair. If the relative ridges have similar length and curvature, this compensation is positive and vice versa. Another distinctive approach uses the reliable detection of sweat pores. Since this approach requires high-resolution scanners, it is not the range of our recent research and not discussed in this thesis.

4.3 Application and Optimization

To develop an efficient fingerprint verification and identification approach depends on its real application environment. We aim to discover one approach suitable for implementation in embedded system with limitations of computational capability and also available memory resources. As described in last section, various fingerprint matching algorithms can be classified into three categories. In correlation-based fingerprint matching, the reference image and the query image are correlated to estimate the degree of similarity between them. If the rotation

and displacement between these two images are unknown, the correlation must be computed over all possible rotations and displacement, which is very computationally expensive. Moreover the non-linear distortion and noise decrease the global correlation degree between these two images extraordinarily. However the ridge feature-based methods suffer from their low discrimination capability, which are generally suitable for such applications with small acquisition area and only 4-5 minutiae available. In minutiae-based matching, the similarity between two templates depends on the matching of each minutia pair and therefore be proportional to the number of the minutia pairs. Although minutiae contain most of the fingerprint discriminatory information, the missing or spurious minutiae in minutiae extraction because of fingerprint image acquisition and pre-processing deteriorate the eventual matching result. [73] The complexity of this method can be different. Some earlier researches based on only the location and orientation of the minutiae are comparatively faster but lack of robustness [82, 83]. Later researches combine the minutiae features and some other features such as the orientation and frequency of ridges, the type of singularities to improve this kind of method and obtain better robustness at cost of more computational complexity.

As a trade-off between the computational complexity and discrimination capability in view of our hardware environments such as fingerprint sensor and processor speed, we have developed one minutiae-based fingerprint matching approach. In this approach several improvements have been made to overcome the shortcomings mentioned above. (1) Feature vector pairs are used instead of minutia pairs to find the best translation and rotation parameters since each feature vector pair contains two minutia pairs and provides a factor of n^2 with respect to n of one minutia pair, which make these parameters more precise. (2) Fingerprint identification is a $1{:}\,N$ matching process, which consists of most matching between two templates from different fingers. Several accelerating algorithms are designed to discriminate two different fingers and end the matching process quickly through calculating the scores of all raw matched

feature vectors, comparing the different of the directional image blocks which contain the valid feature vectors and counting the proportion of the corresponding directional blocks in directional images. (3) Additionally two more similarity arbitration algorithms are designed to re-evaluate the matching score if matching fails with the standard algorithm. One hypothesis is that if two templates are from the same finger, one block directional image and another block direction image after rotating and translating according to some parameters should obtain the best overlap. To find these parameters, each reliable minutia from one template will be compared with each minutia from another. If they are similar enough, i.e. with similar curvature and limited difference of minutia angle and location, the difference between them will be considered as the parameters to transform the corresponding block directional image. Finally all possible parameter sets are recorded and one best set is selected as the translation and rotation parameters to execute the standard algorithm again. Another hypothesis is that since singularities are usually characterized by high curvature [73], the minutia point with higher curvature can represent the distinctive shapes of the original fingerprint image better. Therefore all minutiae are rearranged and several minutiae with higher curvature are selected as candidates, which are used one by one to find the best matched minutia on the other template. If a matched minutia can be found, the similarity of this minutia pair is recorded. The translation and rotation parameters of the minutia pair with the best similarity will be used to execute a simplified standard matching algorithm.

The proposed minutiae-based multiple-matching algorithm can be summarized into two main steps: (1) template location and (2) template multiple-matching. In the template location step the translation and rotation between the reference template and the template to be matched are computed and the most reliable feature vector pairs are obtained as the input of the next step. In the template multiple-matching step the similarity between two templates is arbitrated with different matching algorithms for the reason that some single matching algorithm

cannot discriminate the two templates as expected. The key steps of our algorithm are described as follows.

A. The Translation and Rotation Location and the Feature Vector Pairs Investigation

i. To find the optimal translation and rotation parameters, each minutia is connected with another ($\vec{P}_{ij}$, see Figure 4.2) within each template. The feature vector of this line segment is defined as $\vec{P}_{ij} = \{\theta_{ij}, \delta_i, \delta_j, l_{ij}\}$ where θ_{ij} denotes the angle from this line segment to x-axis anticlockwise; and (δ_i, δ_j) indicates the direction from the vector $\vec{P}_{ij}$ to minutia i and the direction from the vector $\vec{P}_{ij}$ to minutia j; l_{ij} presents the length of the vector $\vec{P}_{ij}$. These parameters are independent of the translation and rotation between two templates. If one minutia in a vector is not reliable enough, which means that its minutia score is relative low, or if the length of this vector is outside our predefined threshold, the vector will be not included. And then all qualified vectors are rearranged and sorted by their length and updated for later use.

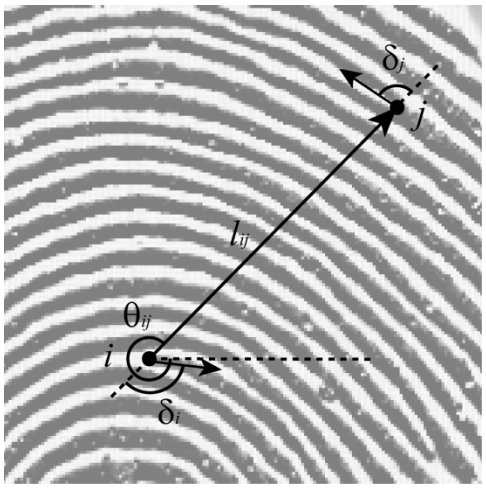


Figure 4.2: Feature vector between two minutiae.

ii. The unreliable minutiae are filtered using block direction image and the vectors from the previous step are updated. To compare the respective direction image in block requires such a transformation of one block direction image F as to reach a better overlap between the two images F and G. The correlation between the core points from the two templates is used to retrieve the approximate transformation parameters. So this step can be ignored for those templates without core. In general there may be three possibilities of core condition:

$$
\begin{cases}
a)\ N_1 = 1, N_2 = 1 \\
b)\ N_1 = 1, N_2 = 2\ or\ \ N_1 = 2, N_2 = 1 \\
c)\ N_1 = 2, N_2 = 2 \\
\\
Rotation\ and\ Displacement: \\
if\ \ N_1 = N_2 = 1 \\
\quad \begin{cases} \alpha = dir_2 - dir_1 \\ cx = x_2 - x_1 \\ cy = y_2 - y_1 \end{cases} \\
\\
else\ if\ \ N_1 = N_2 = 2 \\
\quad \begin{cases} \alpha = dir_2(\overline{core12}) - dir_1(\overline{core12}) \\ cx = \dfrac{x_{2\ core1} + x_{2\ core2}}{2} - \dfrac{x_{1\ core1} + x_{1\ core2}}{2} \\ cy = \dfrac{y_{2\ core1} + y_{2\ core2}}{2} - \dfrac{y_{1\ core1} + y_{1\ core2}}{2} \end{cases} \\
\\
else \\
\quad if\ \ N_1 = 1 \\
\quad\quad \begin{cases} \alpha = dir_2(matched) - dir_1 \\ cx = x_2(matched) - x_1 \\ cy = y_2(matched) - y_1 \end{cases} \\
\\
\quad if\ \ N_2 = 1 \\
\quad\quad \begin{cases} \alpha = dir_2 - dir_1(matched) \\ cx = x_2 - x_1(matched) \\ cy = y_2 - y_1(matched) \end{cases}
\end{cases}
\tag{4.2}
$$

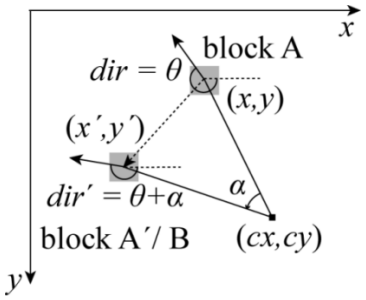
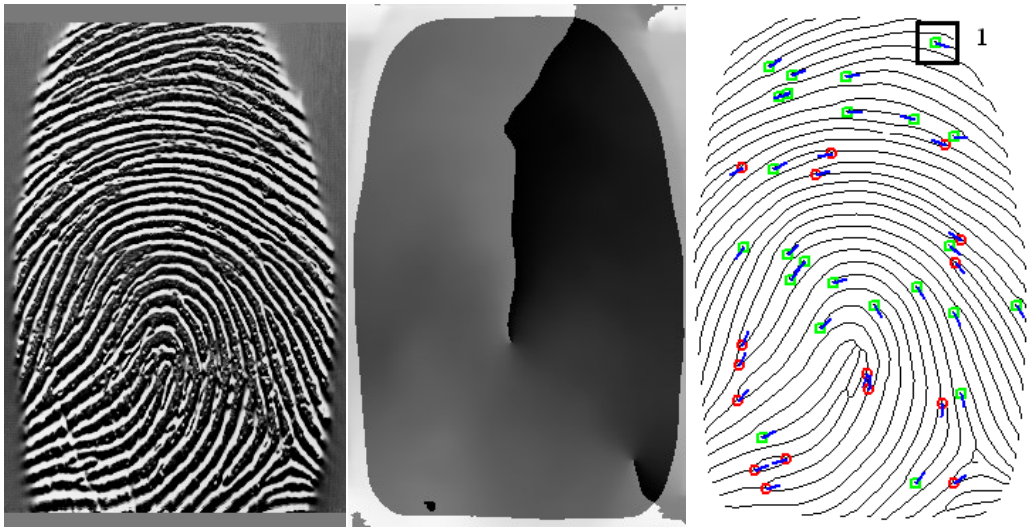
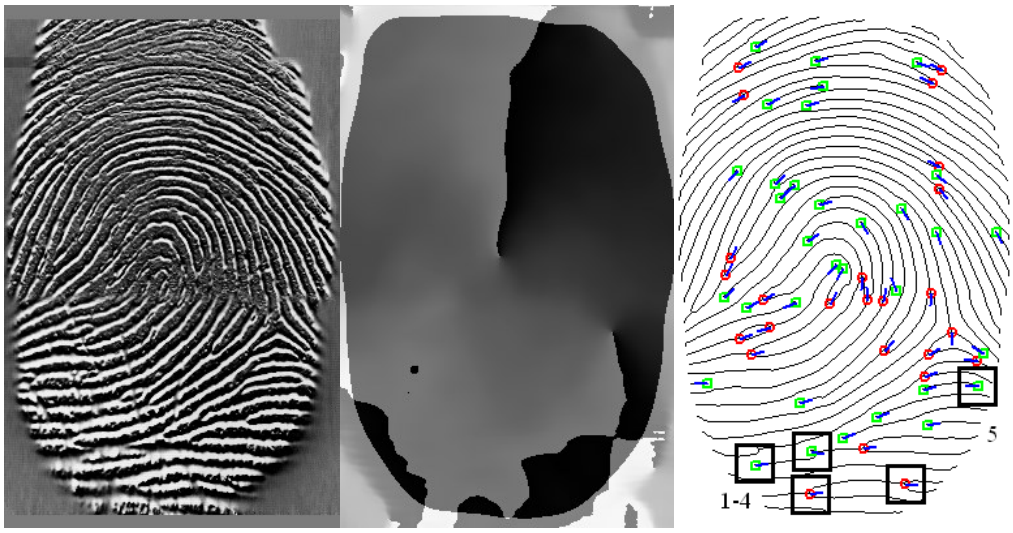


Figure 4.3: Minutiae filtering using block direction image.



a) FVC2006 DB3_A: 115_12.bmp



b) FVC2006 DB3_A: 115_11.bmp

Figure 4.4: Minutiae filtering using block direction image.

In (4.2) the core numbers N_i can be different. The aim is to find which core is matched in one template with which core in another. Subsequently block direction image F is transformed to block direction image F' (see Figure 4.3) and the corresponding template too. In images F' and G the blocks with indefinite direction, i.e. noise, are temporarily marked as invalid. Any minutia, which is located in such a noise area i.e. there are no valid direction blocks around with threshold radius r or the two relative central direction blocks have too much difference, will be eliminated.

In Figure 4.4 shows the result of the method - original fingerprint image, continuous direction image and minutiae with binarized image. Block direction image uses the direction at the centre of each block from continuous direction image. Minutia 1 in a) and minutiae 1-4 in b) are deleted for the lack of the overlap of direction blocks and minutia 5 in b) is deleted for too much difference of relative direction blocks between two block direction images.

iii. In this step all qualified feature vectors $\vec{P}_{ij}$ in both templates are compared in all possible directions and every comparison score will be accumulated. Since (δ_i, δ_j) in $\vec{P}_{ij}$ is independent of the direction of a template, the feature vectors with similar (δ_i, δ_j) can be directly matched without rotation previously. In view of the influence of distortion the feature vectors from the second template are chosen not only at the precise (δ_i, δ_j) but also its upper and lower deviations to obtain better focus. The score of each matched vector pair will be evaluated under a 3-D Guess distribution according to the difference of the features $(\delta_i, \delta_j, l_{ij})$. Then the score is further optimized with some features from the related minutiae themselves. The procedure of this step is described as follows.

$$\begin{cases} for\ each\ i \in (0,2\pi) \\ \quad t_1 = i - T_1, t_2 = i + T_1 \\ \quad for\ each\ \delta_{ij}\ in\ template\ 1 \\ \quad\quad for\ each\ j \in (t_1, t_2) \\ \quad\quad\quad for\ each\ \delta'_{ij}\ in\ template\ 2 \\ \quad\quad\quad\quad if\ |\delta_i - \delta'_i| > T_2 \quad continue \\ \quad\quad\quad\quad if\ \left|\delta_j - \delta'_j\right| > T_2 \quad continue \\ \quad\quad\quad\quad if\ \left|l_{ij} - l'_{ij}\right| > T_3 \quad continue \\ \quad\quad\quad\quad score = G\left(|\delta_i - \delta'_i|, \left|\delta_j - \delta'_j\right|, \left|l_{ij} - l'_{ij}\right|\right) \\ \quad\quad\quad\quad score\ optimization \dots \\ \quad\quad\quad\quad angle[|\delta_i - \delta'_i|] += score \\ \quad\quad\quad\quad angle\left[\left|\delta_j - \delta'_j\right|\right] += score \\ \quad\quad\quad\quad pairlist\ update \dots \\ \quad\quad\quad\quad if \quad score > score_{max} \\ \quad\quad\quad\quad\quad score_{max}\ update \dots \\ \quad\quad\quad \sum score_{max}\ update \dots \end{cases}$$

(4.3)

In (4.3) the constants $T_1 - T_3$ are predefined thresholds and function G submits to Gauss distribution. Array $angle[\ ]$ records the statistic arrangement of rotation between two templates. Score optimization is here applied with five algorithms.

Algorithm 1:

$$C_{xy} = \frac{\sum_{i,j=(x,y)-w}^{(x,y)+w} \left| dir_0 - dir_{ij},\ if\ (dir_{ij} \notin noise) \right|}{N}$$

(4.4)

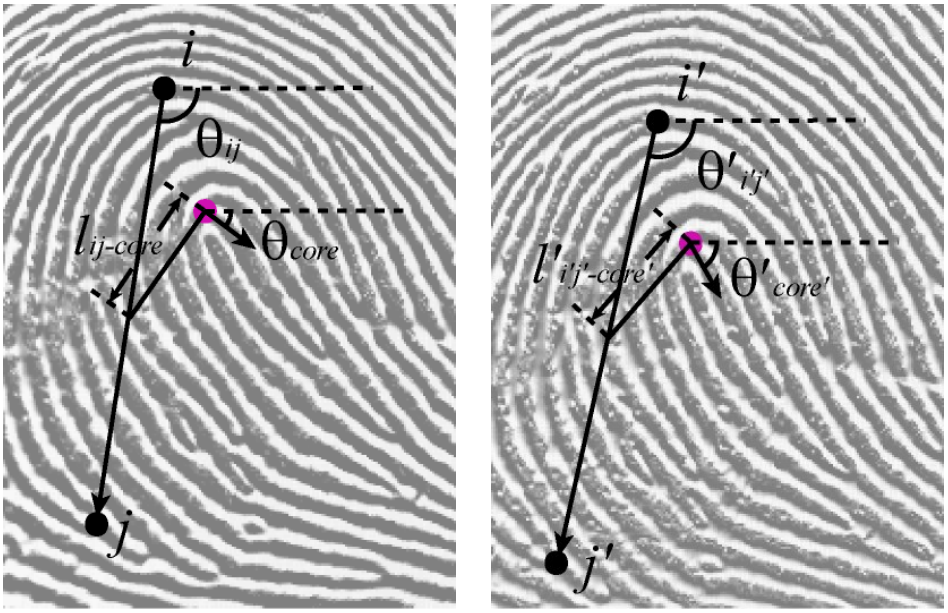
Define C_{xy} in (4.4) as the curvature of one minutia at the position (x, y). The variants dir_0 and dir_{ij} indicate the direction at position (x, y) and its neighbours. The denominator N denotes the number of pixels with qualified direction. If C_{xy} and C'_{xy} from the corresponding feature vectors are not the same. The score is scaled according to the difference. Furthermore, if these two minutiae are reliable enough, i.e., the minutia

scores are greater than a threshold, their minutia types are also considered as a factor to scale the score.

Algorithm 2:

As shown in Figure 4.5, define $l_{ij-core}$ as the distance between the midpoint of feature vector $\vec{P}_{ij}$ and the core point, and θ_{ij} also in Figure 4.2 as the angle of $\vec{P}_{ij}$, and θ_{ij}as the direction of core point. If the feature vectors $\vec{P}_{ij}$and $\overrightarrow{P'}_{ij}$are a correct pair, $\Delta l_{ij-core}$ and $\Delta(\theta_{ij} - \theta_{core})$ should be within a threshold as shown in (4.5). If (4.5) is not supplied, the score of this minutiae pair is scaled.

$$\begin{cases} \left| l_{ij-core} - l'_{i'j'-core'} \right| < T_4 \\ \left| \left| \theta_{ij} - \theta_{core} \right| - \left| \theta'_{i'j'} - \theta'_{core'} \right| \right| < T_5 \end{cases} \tag{4.5}$$



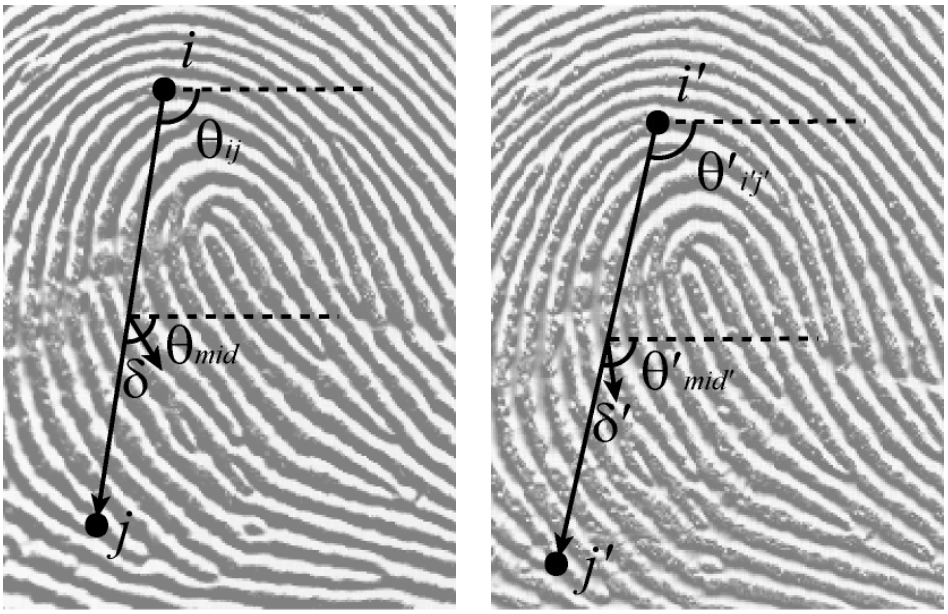
FVC2006 DB3_A: 111_8.bmp, 111_12.bmp

Figure 4.5: Minutiae similarity decided with singularity.

Algorithm 3:

As shown in Figure 4.6, define θ_{mid} as the direction of the midpoint of the feature vector $\vec{P}_{ij}$. If (4.6) is not supplied, the score of this minutiae pair is scaled.

$$\begin{cases} \delta = \left|\theta_{ij} - \theta_{mid}\right| \\ \delta' = \left|\theta'_{i'j'} - \theta'_{mid'}\right| \\ |\delta - \delta'| < T_6 \end{cases} \tag{4.6}$$



FVC2006 DB3_A: 111_8.bmp, 111_12.bmp

Figure 4.6: Minutiae similarity decided with midpoint.

Algorithm 4:

In this step frequency block image F is used to measure the difference between the two feature vectors. If the minutiae from the corresponding feature vectors don't have similar frequency in their frequency blocks (see (4.7)), the score of this minutiae pair is also scaled.

$$\begin{cases} |f_i - f_{i'}| < T_7 \\ |f_j - f_{j'}| < T_7 \end{cases} \tag{4.7}$$

Algorithm 5:

Figure 4.7 and Figure 4.8 show two different descriptors, which symbolize the relation between a central minutia and its neighbours. Some parameters are extracted and matched. The similarity is then converted to standard similarity score. If either of these two descriptors is not well matched, the score of this minutiae pair is to some extent scaled.

The descriptor in Figure 4.7 denotes that the direction blocks around the minutia should also be consistent between two to-be-matched templates. These blocks are divided into several different groups according to the distance to the central minutia. The farther the direction blocks are to the central minutia, the more detailed they will be segmented. The differences of the minutia direction and such 16×16 direction blocks at each circle with radius from r_i to r_k are recorded and matched. In order to minimize these differences, the corresponding direction blocks in these two templates should be independent of rotation. Therefore the first direction block is located along the direction of the minutia and at the circle with the first radius r_i. If some direction block is marked as noise previously, it will also be recorded but with invalidity.

$$\begin{cases} for\ each\ r \in (i,k) \\ \quad for\ each\ n \in (1, N_r) \\ \quad\quad orient_m = opt_1(\theta_0, \theta_{rn}) \\ score_{descriptor\_orient} = \sum_{n=0}^{N} opt_2\big(orient_1(n), orient_2(n)\big) \end{cases} \tag{4.8}$$

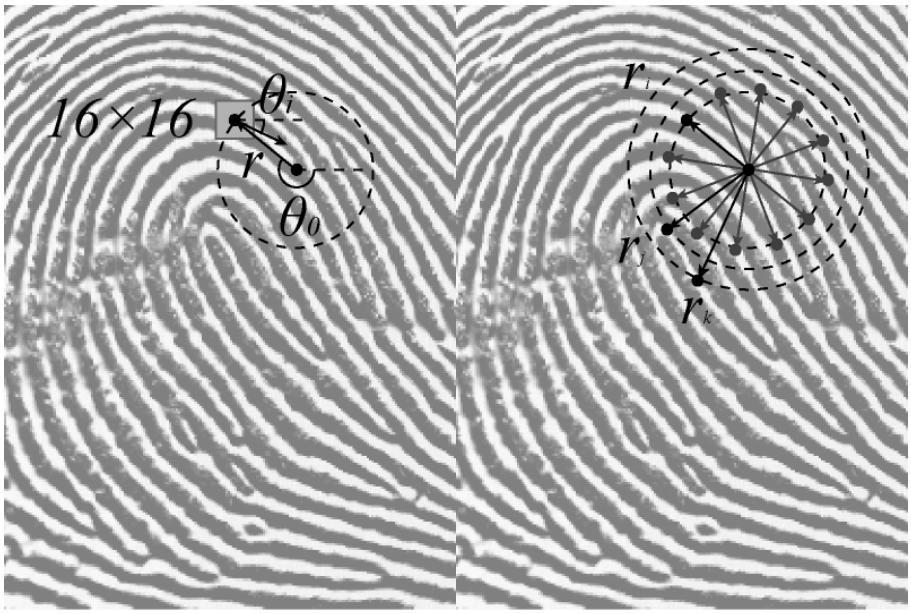
In (4.8) (i,k) gives the possible distances between central minutia and its neighbour direction blocks. Function opt_1 normalizes each difference of

the minutia direction and the block direction. Function opt_2 converts the orientation difference to a measurable score. And the variable N denotes the number of all neighbour direction blocks.

The descriptor in Figure 4.8 denotes relation between the central minutia and its neighbour minutiae within radius r. The similar approach called local structure matching can also be referred in [55]. A constant radius R is defined in advance. Two angle parameters will be extracted from the central minutia m_0 and its neighbour m_i, which has the distance within R to the central minutia. One parameter is the alpha, which presents the angle difference of minutia m_0 and vector $\overrightarrow{m_0 m_i}$. Another parameter is the delta, which shows the angle difference of minutia m_0 and minutia m_i. The comparison of the alpha and delta for a minutia pair is shown in (4.9).

$$
\begin{cases}
\alpha_i = |\varphi_i - \theta_0| \\
\delta_i = |\theta_0 - \theta_i| \\
for\ each\ i,j \epsilon (0, maxCount_{ij}) \\
\quad pair_i[i] = 0 \\
\quad pair_j[j] = 0 \\
if\ \left(|\alpha_i - \alpha_j| < T_8\ and\ |\delta_i - \delta_j| < T_8\ and\ |r_i - r_j| < T_9\right) \\
\quad pair_i[i] = 1 \\
\quad pair_j[j] = 1 \\
for\ each\ i,j \epsilon (0, maxCount_{ij}) \\
\quad if(pair_i[i] == 1)\ \ nNum_i++ \\
\quad if(pair_j[j] == 1)\ \ nNum_j++ \\
score_{descriptor\_minutiae} = opt_3\left(\dfrac{(nNum_i + 1) \times (nNum_j + 1)}{(maxCount_i + 1) \times (maxCount_j + 1)}\right)
\end{cases}
\tag{4.9}
$$

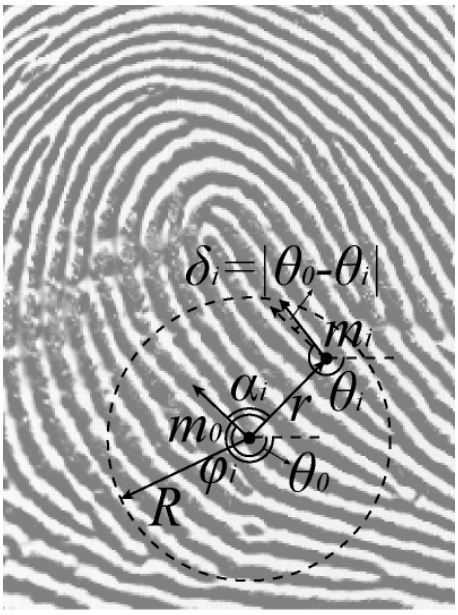
Evidently each minutia pair $\overrightarrow{m_0 m_i}$ in one template is compared with each pair from another. Moreover, the condition of the comparison consists of the difference not only of angles alpha and delta but also of the Euclidean distance of the minutia pair. In this thesis the radius r in Figure 4.7 is set to 27, 45, 63 and 81 and the radius R in Figure 4.8 is set to 80.



a) b)

FVC2006 DB3_A: 111_8.bmp

Figure 4.7: Minutiae local orientation descriptor.



FVC2006 DB3_A: 111_8.bmp

Figure 4.8: Minutiae local neighbour descriptor.

iv. As discussed in step (iii) all feature vectors $\vec{P}_{ij}$ from two templates are cross-matched according to their intrinsic parameters $\{\theta_{ij}, \delta_i, \delta_j, l_{ij}\}$ at all possible rotations. The matching scores of two feature vectors are accumulated in array $angle[\ ]$. In this step the final translation and rotation parameters will be extracted based on (i) - (iii).

To gain precise rotation parameter, array $angle[\ ]$ is firstly locally focused, i.e. each matching score here is amplified in a limited field (see (4.10) (a)). Hereafter the angle which has the maximal matching score is set as the focus and its neighbouring angles, which have only limited angle difference from the focus, use such difference as weighted factors for computing the final rotation angle (see (4.10) (b) and (c)).

$$Angle'[i] = \sum_{j=i-w}^{i+w} Angle[j] \qquad \text{(a)}$$

$$n_{focus} = the\ index\ of\ Max\big(Angle'[i], i \in (0,2\pi)\big) \qquad \text{(b)}$$

$$angle_{rot} = opt\left(\frac{\sum_{i=n_{focus}-w}^{n_{focus}+w}(Angle'[i] \times i) \quad (if\ Angle'[i] > Angle'_{Max}/2)}{\sum_{i=n_{focus}-w}^{n_{focus}+w} Angle'[i]}\right) \ (c)$$

(4.10)

v. With the rotation angle the template to be matched is rotated to the optimal direction to the reference template. Before computing the translation, one preparation step is essential. Each matched pair from pair list in (4.3) should be checked whether their slopes are similar. If the two slopes from a pair are quite different $\left(\Delta\varphi > \frac{\pi}{12}\right)$, this pair will be removed from the pair list. And the translation parameters can then be computed as (4.10).

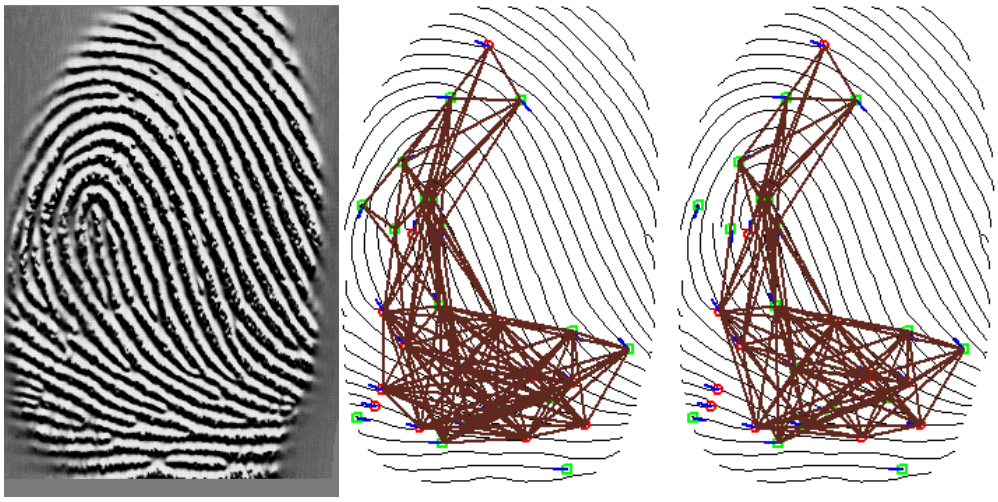
vi. Some post-processing are implemented in this step to further optimize the feature vector pairs in view of computing more reliable matching score later. From this step, the template to be matched and its related direction blocks have been rotated and translated to the best position to the reference template.

The direction blocks are firstly used as the reference of the reliability of a matched feature vector pair. The percentage of the non-noise blocks of the two templates is recorded as p_c and the average similarity of all the corresponding blocks, whose similarities are beyond the threshold $\left(\Delta dir > \frac{\pi}{24}\right)$, is recorded as p_b. If either of p_c and p_b is below pre-defined threshold T_{10} and T_{10}, it can be directly decided that the matching of these two templates is rejected.

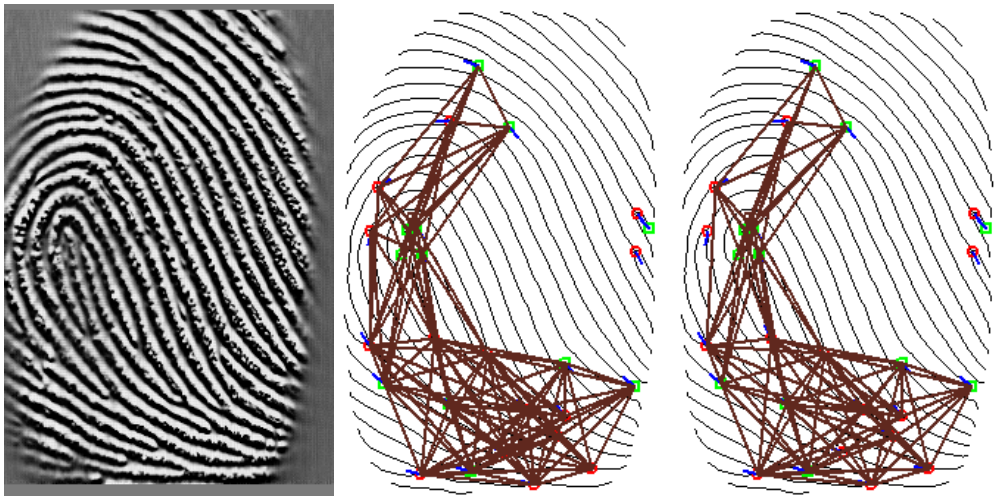
When p_c and p_b are satisfied the condition, the minutiae of both templates will be checked again according to their corresponding block direction image. If one minutia lies in a noise block, it should be removed and the related feature vector too.

Another optimization is a compromise to the finger distortion. We define the feature vector $\vec{P}_{ij} = (x_1, y_1, x_2, y_2)$ and the feature vector $\vec{P}'_{ij} = (x'_1, y'_1, x'_2, y'_2)$, which are a matched pair from two templates respectively. If either of the conditions in (4.11) is not supplied, this pair will be removed because of too much distortion. The result is shown in Figure 4.9.

$$\begin{cases} \Delta x_1 = |x_1 - x'_1| < T_{12} \\ \Delta y_1 = |y_1 - y'_1| < T_{12} \\ \Delta x_2 = |x_2 - x'_2| < T_{12} \\ \Delta y_2 = |y_2 - y'_2| < T_{12} \end{cases} \tag{4.11}$$



FVC2006 DB3_A: 42_8.bmp



FVC2006 DB3_A: 42_9.bmp

a) b) c)

Figure 4.9: Comparisons of feature vectors before and after optimization for Finger Distortion: a) Original grey fingerprint image; b) Feature vectors before optimization; c) Feature vectors after optimization.

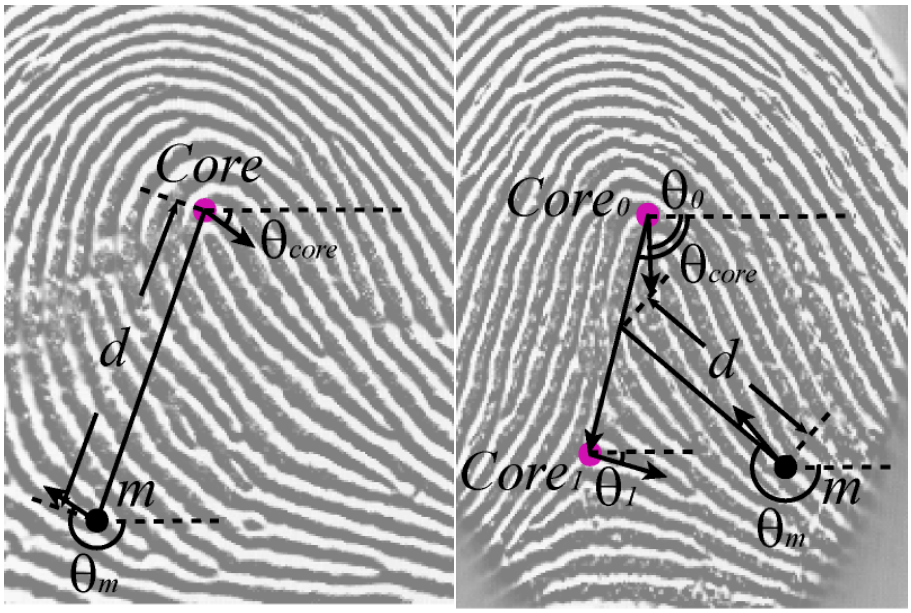
Furthermore there might be a special situation that more than one feature vector in one template are matched with the same feature vector in another template. Generally only one pair among these matched pairs is correct, and the others are caused by associated false minutiae or too dense minutiae. In this event the matching scores of all pairs with one

same component are compared and the pair with the best score is kept and others are removed from the pair list.

B. Matching Score Computation with Multi-matching Algorithms

After the template to be matched is transformed to the optimal position as described in last section, two templates can be further matched with various approaches. In this section the matching score $score\_max$ obtained above will be temporarily stored for reference, and new matching score will then be computed and evaluated in the following steps.

i. A fast algorithm is firstly proposed in order to finish matching between two templates with high similarity. Several minutiae (we define $N = 7$) from the reference template are chosen randomly, which should have enough distance $(d > 30)$ from each other. Figure 4.10 shows the correlation between a minutia m and core points of different number. The selected minutiae in the reference template have their own parameters $(d, |\theta_m - \theta_{core}|)$. If the template has two singularities, θ_{core} means the slope of vector from one core to another. Each parameter will be set as a reference and used to find the best matched minutia in another template. If such qualified minutia exists in the template to be matched, this template will be transformed again according to the difference between this pair and the matching result is marked as a candidate $score_i$. The best of the candidates from $score_i$ $\left(i \in (0, N)\right)$ is chosen as the final score $score_f$ of this algorithm. If $score_f$ and $score\_max$ are both beyond their thresholds, the two templates are regarded as from the same finger.



FVC2006 DB3_A: 111_8.bmp, 17_8.bmp

Figure 4.10: Correlations between a selected minutia and one singularity or two singularities.

ii. If the algorithm in (i) cannot offer one positive decision, local matching [55, 93] is applied to obtain a negative one with celerity. This method makes use of the relations, also shown in Figure 4.8, between one minutia and its neighbour minutiae within definite spatial distance. The starting threshold of this distance is $d_{max1} = 100$. If no enough minutiae $(n_{neighbors} < 8)$ can be obtained in this area, the distance will be enlarged up to $d_{maxN} = 200$. When all minutiae pairs from the related feature vectors are scanned and their neighbours are recorded, a local matching will be executed between these minutiae with enough neighbours in respective templates. Different from [93], we just count the number of matched minutia pairs instead of accumulating the similarities. If the count cannot reach our threshold, the two templates are regarded as from different fingers.

iii. If the decision cannot be made in (i) and (ii) that the two templates are from the same finger or different fingers, another reference score $score_d$ will be generated in this step using the descriptors described in section - *A*

(iii) algorithm 5. The scores $score_{descriptor\_orient}$ and $score_{descriptor\_minutiae}$ will be computed for each minutia pair and accumulated in (4.12).

$$score_d = \sum_{i=0}^{N_{\substack{number\ of \\ minutiae\ pairs}}} \frac{score_{descriptor\_orient}(i) \times score_{descriptor\_minutiae}(i)}{100} \tag{4.12}$$

Combined with $score_d$ and block scores p_b and p_c the score $score\_max$ will be revised. Finally based on a set of parameters $(score_{max}, score_f, p_b, p_c, N, f_{ridge})$ a reliable matching score $score_{main}$ will be generated.

iv. After $score_{main}$ is obtained, our main matching algorithm is completed. If it is beyond a moderate threshold $(T = 50)$, a final matching score is made. However in order to improve the FRR in statistic result, some optimizations are developed in this step to process the matching with score below this threshold.

Algorithm 1:

The rotation and translation parameters will be recomputed with one approach described in (4.13). On condition that two minutiae from the reference and the to-be-matched template have similar curvature, every minutia in reference template is compared with every minutia in another.

The minutia m_{ref} in the reference template is then rotated according to the difference of the direction of a minutia pair. The new coordinate of this minutia m'_{ref} is compared with the coordinate of another m_{search} and the translation $(\Delta x, \Delta y, \Delta dir)$ is obtained. If this translation is not beyond the threshold, the block direction image of the reference image is rotated according to this rotation and translation and matched with another block

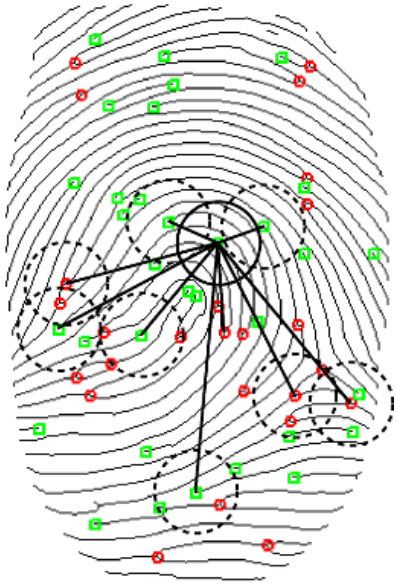
direction image. If the matching of block direction images returns a reliable score, these transformation parameters and their corresponding scores of block image matching are recorded in arrays $coordinates$ and $blockscore$. According to the similarities of the minutiae from array $coordinates$, the array $blockscore$ will be accumulated and the optimal rotation and translation parameters are extracted.

$$
\begin{cases}
for\ each\ i = 1 \cdots N \\
\quad for\ each\ m_{ref} \in (reference\ template) \\
\quad\quad for\ each\ m_{search} \in (to-be-matched\ template) \\
\quad\quad\quad if\ \left|curv_{ref} - curv_{search}\right| < T_{13} \\
\quad\quad\quad\quad m'_{ref} = rot_1\left(m_{ref}, \left|dir_{ref} - dir_{search}\right|\right) \\
\quad\quad\quad\quad if\ \left|x'_{ref} - x_{search}\right| < T_{14}\ and\ \left|y'_{ref} - y_{search}\right| < T_{14} \\
\quad\quad\quad\quad\quad BlockImgRef' = rot_2(BlockImgRef, \Delta x, \Delta y, \Delta dir) \\
\quad\quad\quad\quad\quad BlockImgScore = match(BlockImgRef', BlockImgSearch) \\
\quad\quad\quad\quad\quad if\ BlockImgScore > T_{15} \\
\quad\quad\quad\quad\quad\quad coordinates[i] = (\Delta x, \Delta y, \Delta dir) \\
\quad\quad\quad\quad\quad\quad blockscore[i] = BlockImgScore \\
\quad\quad\quad\quad\quad\quad i = i + 1 \\
for\ each\ i = 1 \cdots num_i \\
\quad for\ each\ j = i + 1 \cdots num_i \\
\quad\quad if\ \Delta(coordinates[i], coordinates[i]) < \left(T_{16}(\Delta x, \Delta y), T_{17}(\Delta dir)\right) \\
\quad\quad\quad blockscore[i] \mathrel{+}= blockscore[j] \\
\quad\quad\quad blockscore[j] \mathrel{+}= blockscore[i] \\
for\ each\ i = 1 \cdots num_i \\
\quad if\ blockscore[i] == Max(blockscore) \\
\quad\quad (\Delta x, \Delta y, \Delta dir)_{optimal} = coordinates[i]
\end{cases}
\tag{4.13}
$$

With these parameters the reference block image and template are transformed. The new reference template and the to-be-matched template are scanned and the minutiae located in the noise block in the corresponding block direction images will be removed. Finally these two new obtained templates are matched with the algorithms mentioned in section *A* and *B* again.

Algorithm 2:

The rotation and translation parameters will be recomputed with another approach described in Figure 4.11.



FVC2006 DB3_A: 111_8.bmp

Figure 4.11: Minutiae selection for transformation parameters using spatial distance, minutia score and curvature.

All minutiae, which have good minutia scores, in the reference template will be rearranged in descending order according to the curvature of each minutia. From the first minutia of this rearranged template, several minutiae (we define $N = 10$) will be selected, which should have a spatial distance more than a threshold (we define $D_{thres} = 30$) from each other. Since these selected minutiae have good minutia scores and relative long distance, they can be considered as reliable and independent from each other. Then every minutia will be compared with all minutiae in another template. If one minutia from the selected has a pair in another template with similar curvature and limited difference of minutia direction, the reference template will be transformed with the difference of these two

minutiae, and a candidate matching score is obtained. Then the transformation with the best candidate matching score is used as the new parameters for repetitive matching. In addition block direction image should be matched if this score is below pre-defined threshold. When this score and the block direction image matching score are both unreliable, the matching of these two templates will be rejected. With this best translation and rotation parameters the reference template is transformed and the both templates are filtered to remove those minutiae located in noise area.

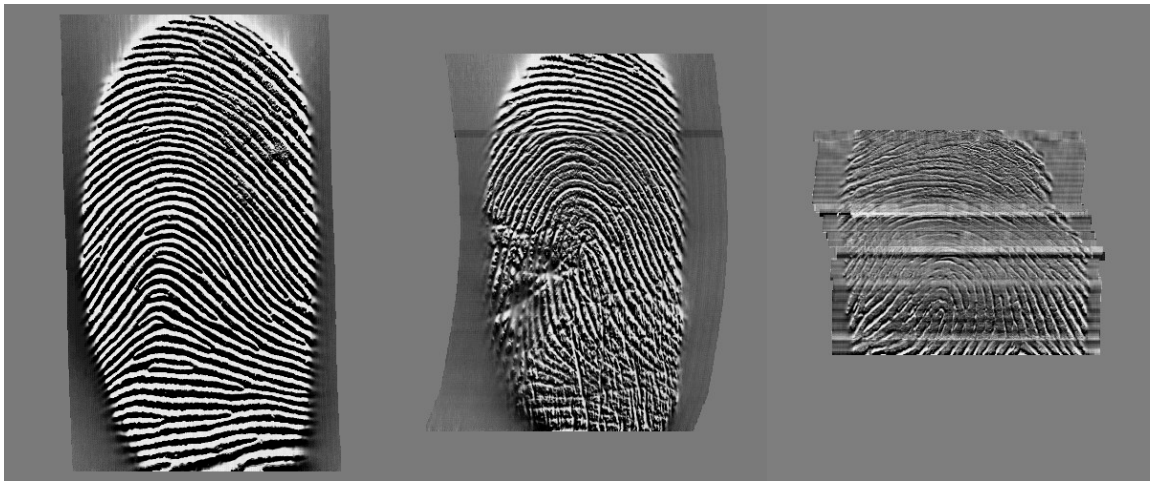
As described in Algorithm 1, these two new obtained templates are matched with the algorithms mentioned in section *A* and *B* again but with more flexible thresholds.

4.4 Experimental Results

Since the proposed algorithm aims to improve the identification system based on the thermal sweeping sensor by ATMEL, in our experiments, the FVC2004 DB3_A and FVC2006 DB3_A have been selected as the testing database for statistics and algorithm comparison. The introduction of FVC can be referred in Appendix A. Figure 4.12 indicates some examples from these two databases.

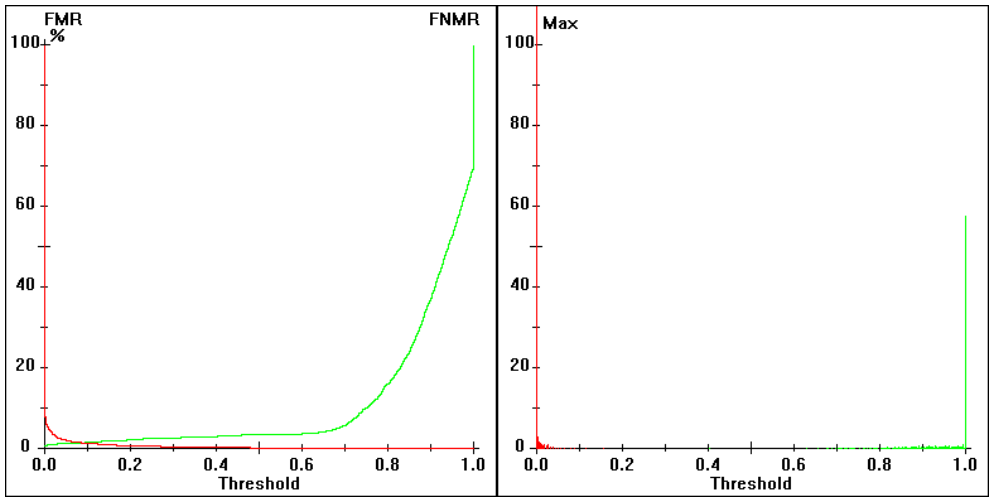
The database FVC2004 DB3_A consists of 800 fingerprint images from 100 fingers with 8 fingerprints each. The number of the genuine match and the imposter match is 2800 and 4950. For imposter match, the first impression of each finger is compared with the first impression of other fingers. The size of the image is 300×480 pixels with a resolution of 500 dpi. The performance of the proposed algorithm on this database is shown in Figure 4.13.

a) FVC2004 DB3_A: 8_8.tif, 100_7.tif, 58_7.tif

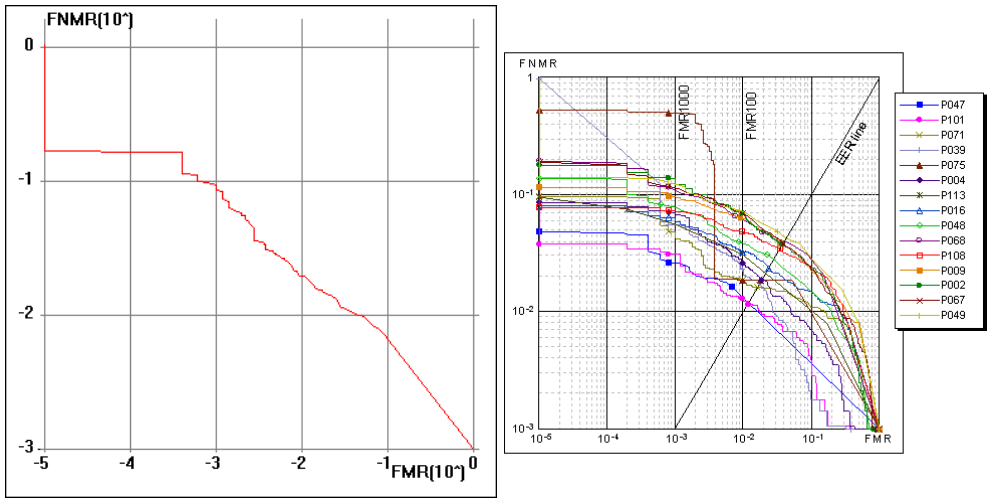


b) FVC2006 DB3_A: 62_12.bmp, 44_7.bmp, 115_1.bmp

Figure 4.12: Fingerprint examples from FVC databases.



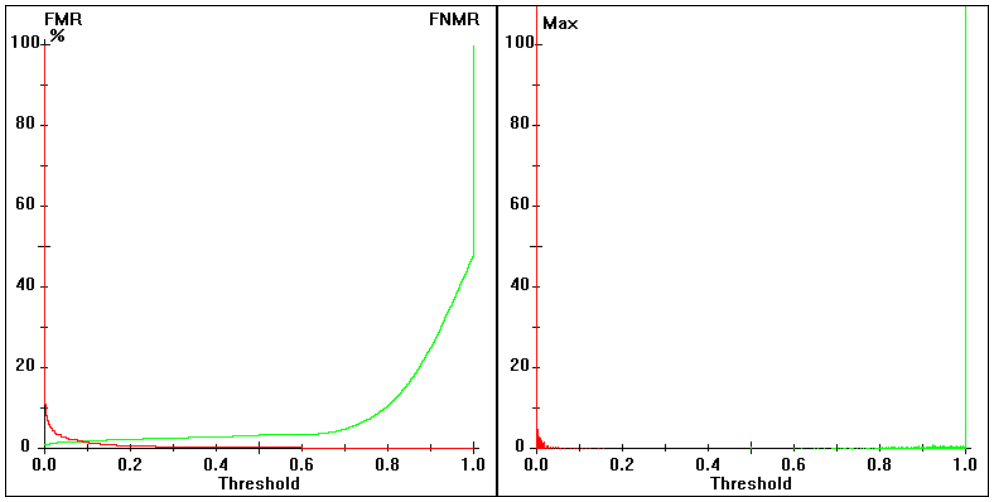
a) FMR(t) and FNMR(t) b) Score distributions



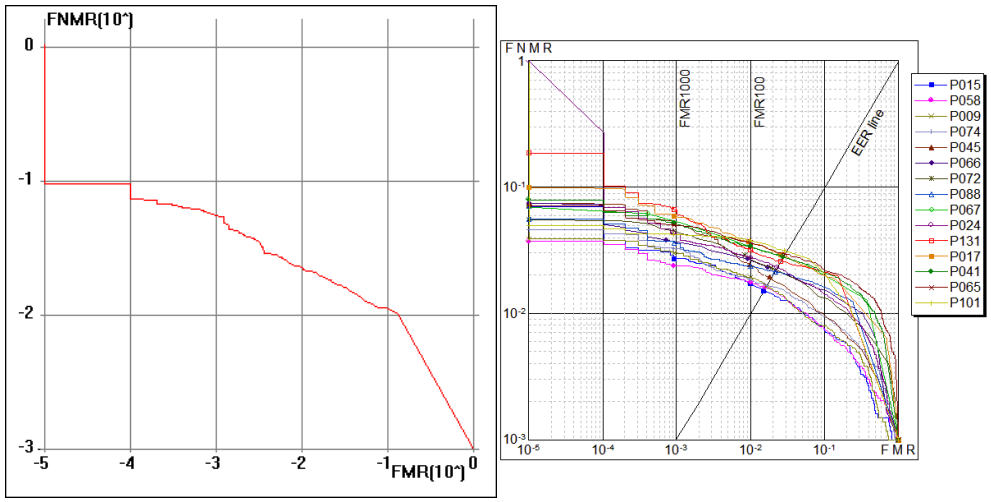
c) ROC curve d) ROC FVC2004

Figure 4.13: Experimental results of our algorithm on FVC2004 DB3_A. Part d) shows the ROC results from the participants in FVC2004.

The database FVC2006 DB3_A consists of 1680 fingerprint images from 140 fingers with 12 fingerprints each. The number of the genuine match and the imposter match is 9240 and 9730. The size of the image is 400×500 pixels with a resolution of 500 dpi. The performance of the proposed algorithm on this database is shown in Figure 4.14.



a) FMR(t) and FNMR(t) b) Score distributions



c) ROC curve d) ROC FVC2006

Figure 4.14: Experimental results of our algorithm on FVC2006 DB3_A.

In the Figure 4.13 and Figure 4.14, the last parts contain the ROC results of the first 15 participants. The detailed comparisons with the best 5 participants' algorithms in FVC2004 and FVC2006 can also be referred in Table 4.1 and Table 4.2. The hardware environment for our test is PC INTEL PENTIUM 4 – 2.8 GHz – 1.00 GB RAM, which is shown in Table 4.4 and also compared with the hardware environment of FVC2004 and FVC2006. A high-level description of the corresponding algorithms in FVC2004 is listed in Table 4.3.

	Proposed	P047	P101	P071	P039	P075	P108
EER	1,07%	1,18%	1,20%	1,64%	1,78%	1,85%	2,92%
FMR 100	1,18%	1,68%	1,32%	1,86%	2,32%	1,89%	4,43%
FMR 1000	5,14%	2,68%	3,11%	4,96%	5,64%	50,79%	9,11%
Zero FMR	21,00%	4,89%	3,79%	9,89%	99,61%	53,96%	11,04%
Average Enroll Time (s)	0,106	2,310	0,070	0,400	0,820	0,330	0,240
Average Match Time (s)	0,008	2,300	0,850	0,810	1,090	0,680	0,250
Average Model Size (byte)	768	1.5K	18.6K	18.8K	3.0K	0.6K	1.6K
Max Model Size (byte)	1160	2.7K	18.6K	28.1K	3.5K	1.2K	1.6K

Table 4.1: Performance comparison with FVC2004 DB3_A

	Proposed	P015	P058	P009	P074	P045	P133
EER	1,88%	1,53%	1,61%	1,65%	1,68%	1,89%	1,63%
FMR 100	2,61%	1,75%	1,79%	1,94%	1,85%	2,47%	1,74%
FMR 1000	5,55%	2,76%	2,41%	3,03%	3,27%	5,26%	3,01%
Zero FMR	7,42%	7,18%	3,76%	3,93%	4,72%	7,59%	4,33%
Average Enroll Time (s)	0,091	1,682	0,307	0,583	0,235	0,194	0,054
Average Match Time (s)	0,005	1,678	0,307	0,641	0,242	0,223	0,056
Average Model Size (byte)	646	5.88K	4.34K	1.81K	4.73K	8.32K	1,71K
Max Model Size (byte)	1113	10.67K	12.31K	3.06K	10.12K	12.79K	1,71K

Table 4.2: Performance comparison with FVC2006 DB3_A

Preprocessing			Alignment		Features								Comparison				
Participant	*Segmentation*	*Enhancement*	*Before Matching, During Matching*	*Displacement, Rotation, Scale, Non-linear*	*Minutiae*	*Singular points*	*Ridges*	*Ridge counts*	*Orientation field*	*Local ridge frequency*	*Texture Measures*	*Raw/Enhanced image parts*	*Minutiae (global)*	*Minutiae (local)*	*Ridge pattern (geometry)*	*Ridge pattern (texture)*	*Correlation*
P002	√	√	D	NL	√				√				√	√			
P009	√	√	BD	DRS	√	√	√	√	√	√			√				
P016		√	-	-	√		√						√				√
P026			-	DR	√			√	√				√				
P027	√	√	D	DRS							√					√	√
P039	√	√	D	N	√				√	√			√				
P041	√	√	D	DR	√		√								√		
P047	√		D	DRSN	√	√	√	√	√	√	√	√	√	√		√	
P049	√	√	D	DR	√	√	√		√	√			√	√			
P050	√	√	B	DR	√				√				√				
P051	√	√	D	DR	√	√		√	√				√	√			√
P067	√	√	D	DRN	√				√				√				
P068	√	√	D	DR	√		√						√				√
P071	(√)	√	D	DR(N)	√	√	√	√	√			(√)		√	√		(√)
P072	√	√	D	DR	√	√			√				√				
P075	√	√	B	DR	√									√			
P078	√	√	D	DRS	√								√				
P087	√	√	D	DR	√				√	√	√		√	√	√		
P097	√		D	DR	√	√			√	√			√	√			
P099	√	√	D	DRN	√								√				
P101	(√)	√	BD	DRS		√	√	√	√	√		√			√		√
P103	√	√	-	-	√				√					√			
P104	√	√	D	DR	√									√			
P105	√	√	B	DR	√	√			√					√	√		
P106			-	-	√		√							√			
P107		√	D	DRS	√	√							√	√			
P108	√	√	D	DR	√	√			√					√			
P111	√	√	D	DR	√				√				√				
P113	√	√	D	N	√	√	√		√	√		√	√	√			√
Proposed	√	√	D	DRS	√	√	√	√	√	√			√	√	√		

Table 4.3: Comparison of the high-level description of the algorithms between the 29 participants in FVC2004 and the proposed algorithm[94]

	Intel Pentium 4 521	Intel Pentium 4 640
Bus Speed	800 MHz	800 MHz
L2 Cache Size	1 MB	2MB
CPU Speed	2.80 GHz	3.20 GHz

Table 4.4: Hard difference between FVC (right) and us (left)

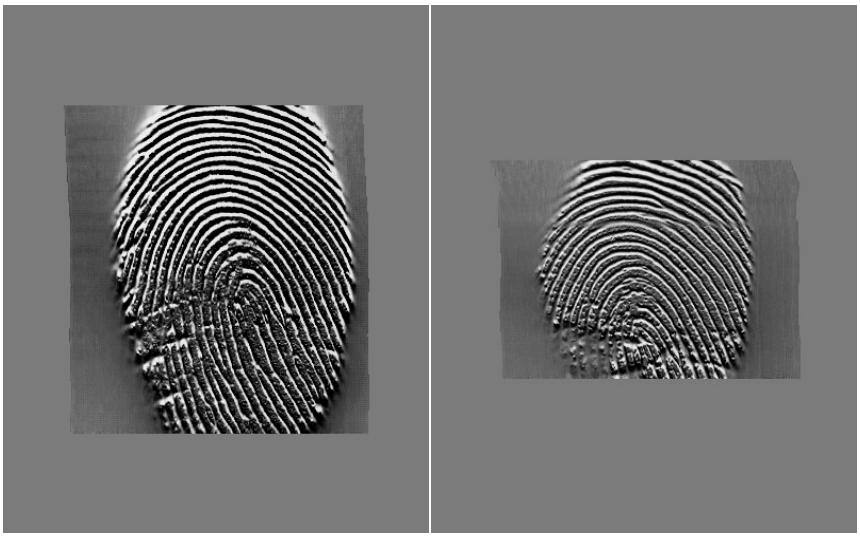
The FVC has two different categories: Open category and Light category. The difference between them is that the open category has no limits on memory requirements and template size, so that most all of the participants can obtain better results. For example, when P101 and P071 in FVC2004 limit their average template sizes from 18.6K and 18.8K to 1.1K and 1.3K respectively, the EERs drop down from 1.2% and 1.64% to 3.57% and 4.69%. And the corresponding Zero FMRs are also two to three times worse (3.79% ➔ 8.14% and 9.89% ➔ 21.11%). Overall the top 10 participants from the open category of FVC2004 gain the average performance drop more than 40 percent on EER and 35 percent on Zero FMR [94] when template size and memory requirement are restricted. To implement our algorithm in embedded hardware environment, we firstly confirm that the memory usage of our algorithm meets the memory requirement of the light category – 4MBytes (In actuality: Max Enroll Memory < 768KByte, Max Match Memory < 256KByte). Based on this condition, our algorithm is compared with the results from the open category of some participants and only the best result from the light category.

As for FVC2004 in Table 4.1, the performance of our algorithm is a little better than P047 and P108, which are the best in the open and light category respectively, except the higher Zero FMR, caused by one imposter comparison which results in a high score. Besides the performance, the efficiency, i.e. the Average Enroll and Match Time, is also remarkable. Template sizes less than 0.5KB are related to those algorithms based only upon minutiae. The size in the range of 1KB to 2KB indicates that not only minutiae points but also other

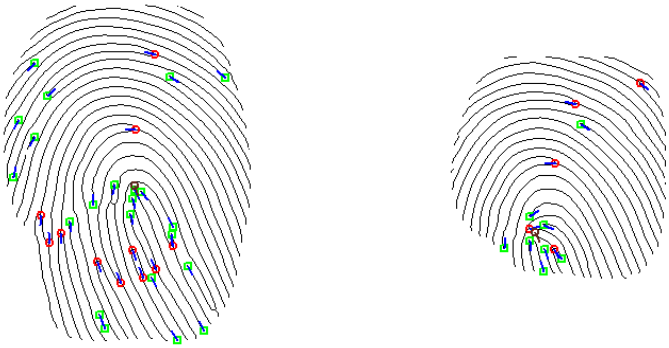
features are used in matching process. The other template formats with larger sizes contain the storage of some portions of the fingerprint image itself [94]. As shown in Table 4.3, P047, P101 and P071 store raw/enhanced image parts for later matching or use correlation-based matching algorithm, which causes either increase of template size or lower matching speed. P039 and P075 use only minutiae to match. However P075 is not much reliable by FMR 1000 in spite of small template size and P039 is just the opposite. The best one P108 in the light category uses also only minutiae to match, which has a relative low matching speed and is still not suitable for embedded application.

As for FVC2006 in Table 4.2, the performance of our algorithm is near to the best results on both open and light category. Therefore the efficiency and the template size should be further compared. Our Average Enroll Time is 0,091s, higher than P133 from the light category and lower than other participants from the open category. On the other hand our Average Match Time 0.005s is at least 10 times faster than all other algorithms. Moreover our template size is comparable with most of other participants from the light category.

In general the performance of our algorithm exhibits a good tradeoff between speed and accuracy. Taking the EER as the reference, we have used relative less information to describe the similarity between two templates than most participants in FVC2004 and FVC2006. Excluding the factor that we have scaled the input fingerprint images to a certain degree, our match algorithm is still much efficient for even larger images (Max Matching Time: 34ms in FVC2004_DB3_A, 77ms in FVC2006_DB3_A). This makes it possible for further research on hardware platform with more limited resources.



FVC2006 DB3_A: 18_3.bmp, 18_4.bmp



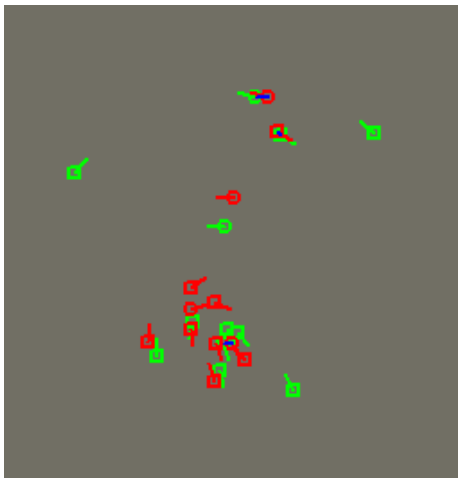


Figure 4.15: Matching failed due to insufficient overlap.

FVC2004 DB3_A: 19_5.bmp, 19_3.bmp

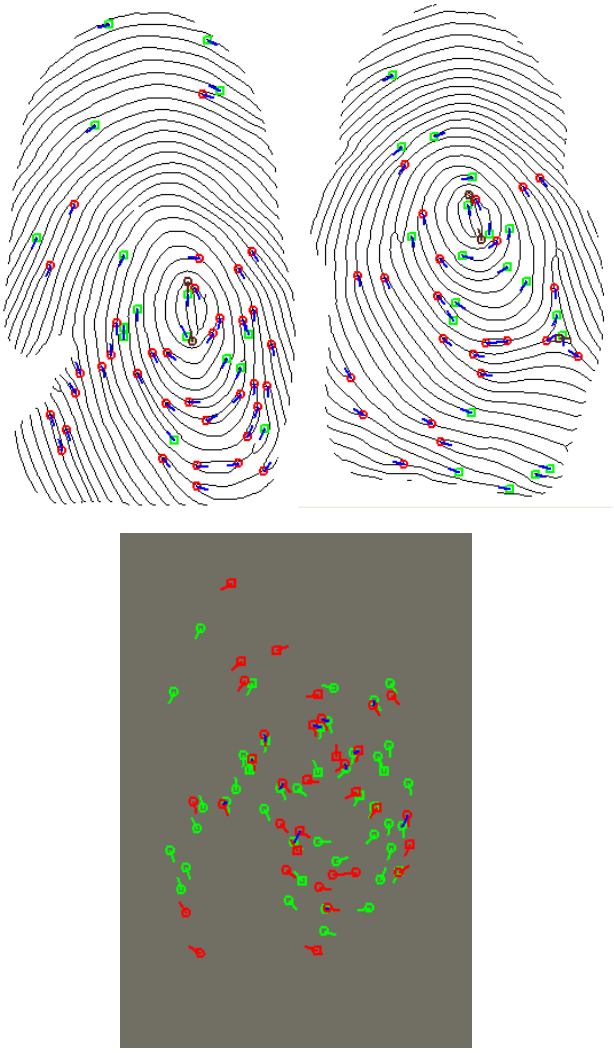


Figure 4.16: Matching failed due to global distortion.

In the databases that we have used, besides those which have very poor qualities, the fingerprint images shown in Figure 4.15 and Figure 4.16 are the two principal kinds which lead to the failure in matching. In Figure 4.15 the overlap area exists only above the core point and in this area of the second fingerprint image no enough minutiae can be extracted unfortunately. Therefore a positive decision cannot be made. The final minutiae's overlap image indicates that only three minutiae pairs are correlated. In addition in Figure 4.16 the overlap area is large enough and also many minutiae are extracted. The nonlinear distortion by fingerprint image acquisitions causes the matching failure. Altogether 11 minutiae pairs have been found; nevertheless the difference of minutia directions and the spatial distance decrease their contributions to the final matching score.

4.5 Summary

We have proposed a minutiae-based multiple-matching method for fingerprint matching. The fingerprint template is represented by feature vectors composed of every two minutiae points and their connecting line. The similarity of two templates can be simplified to find the translation and rotation parameters with which the two templates have the most feature vectors pairs coincided with each other. After the best transformation parameters are obtained, a multiple matching evaluation criterion is applied in the similarity decision. Finally for such matching with uncertainty two other approaches based on the characteristics of each minutia itself are proposed to amend the matching score.

The proposed algorithm has been evaluated with both FVC2004 and FVC2006 DB3 databases, which have relative large distortion and poor quality. The EER are 1.07% and 1.88% respectively. It works well for most of fingerprint images by experiment; however this algorithm still needs to be improved to deal with those images which have too much global distortion or insufficient minutiae.

Chapter 5

Fingerprint Embedded System

5.1 Hardware Architecture

5.1.1 Introduction

A complete Automated Fingerprint Identification Systems (AFIS) contains not only an effective fingerprint feature extraction and matching algorithm but also appropriate hardware environment where the algorithm can be executed. PC has very high processing speed, but is not competent for those offline applications such as access control system, fingerprint ID card etc. An embedded system, which has low-cost, relative high speed and runs independently, provides a better selection for the applications of fingerprint techniques.

During the development of an embedded system, to select one suitable embedded processor is a most crucial step. Different processor may have great difference in use. The Microcontroller Unit (MCU) has sufficient peripheries and low-cost but limited computing performance. Digital Signal Processor (DSP) dedicates to signal processing such as signal filtering, FFT etc. due to its special computing oriented design. Micro Processor Unit (MPU) is high-integrated

general processor expert in computing, which has no such peripheries like MCU. The System on Chip (SoC) utilizes hardware programming language to design more modules besides MCU or MPU through simulation and synthesis with special software and integrates into one chip, which is a real mono-chip computer to some extent.

The stand-alone AFIS that we design should not only provide fingerprint image acquisition and processing but also communicate with its additional subsystems such as relay board, alarm device or door-motor. In view of these practical applications, the SOC ARM9G20 from ARM, which will be introduced in later sections, has been selected for our main hardware development.

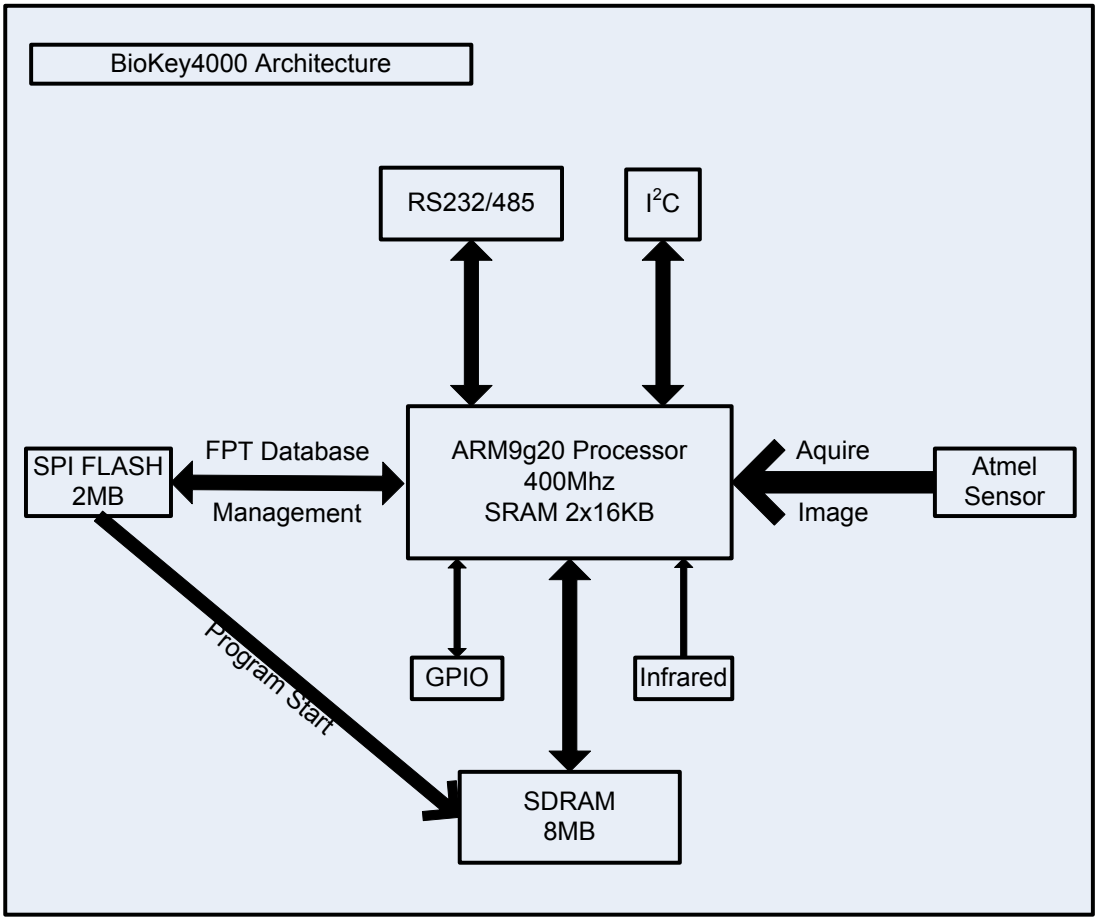


Figure 5.1: Hardware architecture based on ARM9G20.

As shown in Figure 5.1, the core module consists of ARM9G20 processor and several essential components such as sweep sensor, serial flash, SDRAM etc. The fingerprint identification and verification software will be compiled under PC platform and be stored in serial flash. After power up, the bootloader program in serial flash will firstly be executed, which copies the main program to SDRAM. Then the main program, running in SDRAM, communicates with ARM9G20 to process various commands such as fingerprint image acquisition from sensor, fingerprint template read from or write to database located in serial flash or different requests from I/O of processor.

5.1.2 Sensor Techniques

In AFIS fingerprint sensor acts as the data input device to acquire images through reading the ridge pattern on the finger surface and converting the analog signal to digital through an A/D converter. The sensors can be generally classified into several types based on their technologies, e.g. optical, solid-state, ultrasound, etc. A fingerprint encoding and matching algorithm with definite parameters can obtain recognition accuracy with great difference if various sensors are used. Optical sensors and ultrasound sensors can obtain images with good quality and high resolution, but the considerable packaging size or high cost limits their use in embedded system. Some solid-state sensors based on the thermal technology can overcome the above problems; furthermore unlike the optical sensor no latent fingerprint is left on the sensor. The sensor in our AFIS system belongs to the solid-state sensors based on thermal technology. In Figure 5.2 shows a sensor of this kind from ATMEL[95].

Some special pyro-electric materials are used in these sensors to generate current based on temperature differential[96]. When sweeping on the sensor, the ridges of the fingerprint generate a temperature differential against the valleys for the reason that the ridges and the valleys have different distance from the sensor

surface. The sensor continues to heat up and amplifies the temperature differential and converts it to an image to indicate the fingerprint. After fingerprint sweeping there is no more temperature differential on the sensor surface, the sensor returns back to normal temperature. A typical character of this sensor is that fingerprint image has to be reconstructed from all acquired slice images since the area of the temperature differential which can be recorded at one time is limited to the valid surface of the sensor.

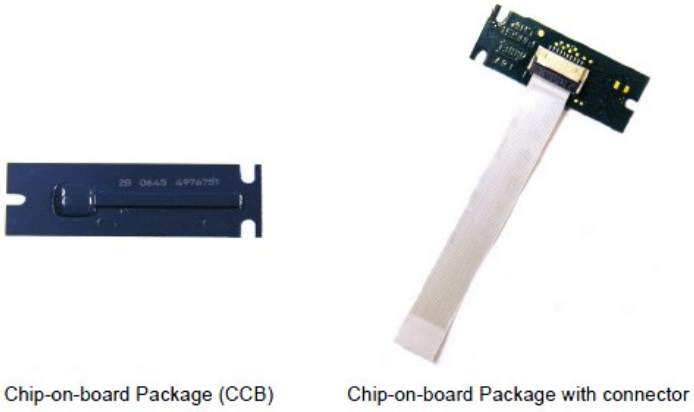


Figure 5.2: FingerChip packages of AT77C102B.

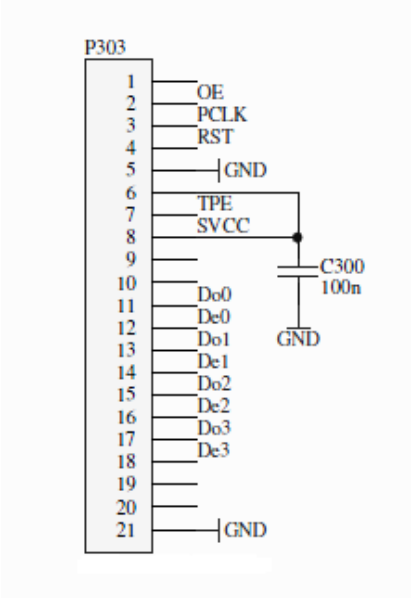


Figure 5.3: AT77C102B pin description.

Figure 5.3 shows the schematic diagram of fingerprint sensor AT77C102B. Sensor initialization should be firstly executed through toggling RST pin (high to low) and adding additional clock pulses to synchronize the pipe-line and skip the first dummy frame. However it is not necessary between each slice acquisition. The pin TPE will be set to turn on heating during the entire acquisition. While fingerprint sweeps, the 8-bit data is transferred through Do0-Do3 and De0-De3 in sequence. If a slice has acceptable variance, it will be stored for later image reconstruction. Otherwise it will be discarded. When the number of received slices reaches the slice limit, the acquisition is completed. The sensor returns to idle state through setting pins RST and OE. In the end all slices will be reconstructed to one fingerprint image through computing the global horizontal and vertical translations of two consecutive slices and combining the slices with these parameters. Figure 5.4 indicates the flow diagram of one fingerprint acquisition.

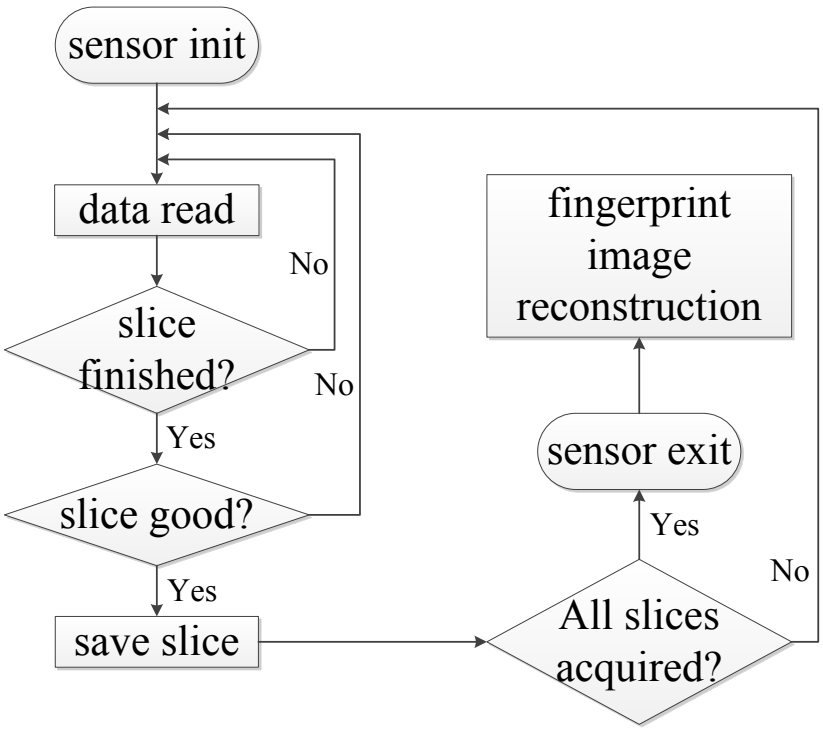


Figure 5.4: Flow diagram of fingerprint image acquisition.

5.1.3 ARM Processor

In consideration of both performance and cost, AT91SAM9G20, which derives from AT91SAM9260 with the same core ARM926EJ-S but faster ROM and RAM memories, is selected in our hardware application. Figure 5.5 shows the block diagram of AT91SAM9G20. As follows lists some features of this processor.

- 400 MHz frequency on the core and 133MHz on the bus;
- 32-KByte Data Cache and 32-Kbyte Instruction Cache;
- Memory Management Unit (MMU);
- Two 16-Kbyte internal SRAM;
- Debug Unit;
- Watchdog Timer
- Other peripherals such as 32-bit PIO controller, UART, Master/Slave SPI and 16-bit Timer/Counter etc.

Since a fingerprint image acquired from sensor has the resolution of 300×400, more than 2-MByte memory should be allocated for intermediate results of image processing and templates matching. Limited to the internal memory size, a 32-bit 8-MByte SDRAM from Micron is used for loading program and executing fingerprint image processing and a 2-MByte SPI serial flash for storing program and fingerprint template database.

As described in Figure 5.1, our system based on AT91SAM9G20 has the following routines. After system starts, a bootloader program in serial flash is firstly activated, which makes some important initializations and copies the main program from serial flash to SDRAM and then this main program runs in SDRAM. When a fingerprint sweeps on the sensor, the fingerprint data is transferred from sensor to SDRAM via processor. The fingerprint data is then processed in both SDRAM and internal SRAM. The notification or result is finally sent out through peripherals such as UART for Relay Board or PC, SPI for serial flash, and also PIO for some other different usage.

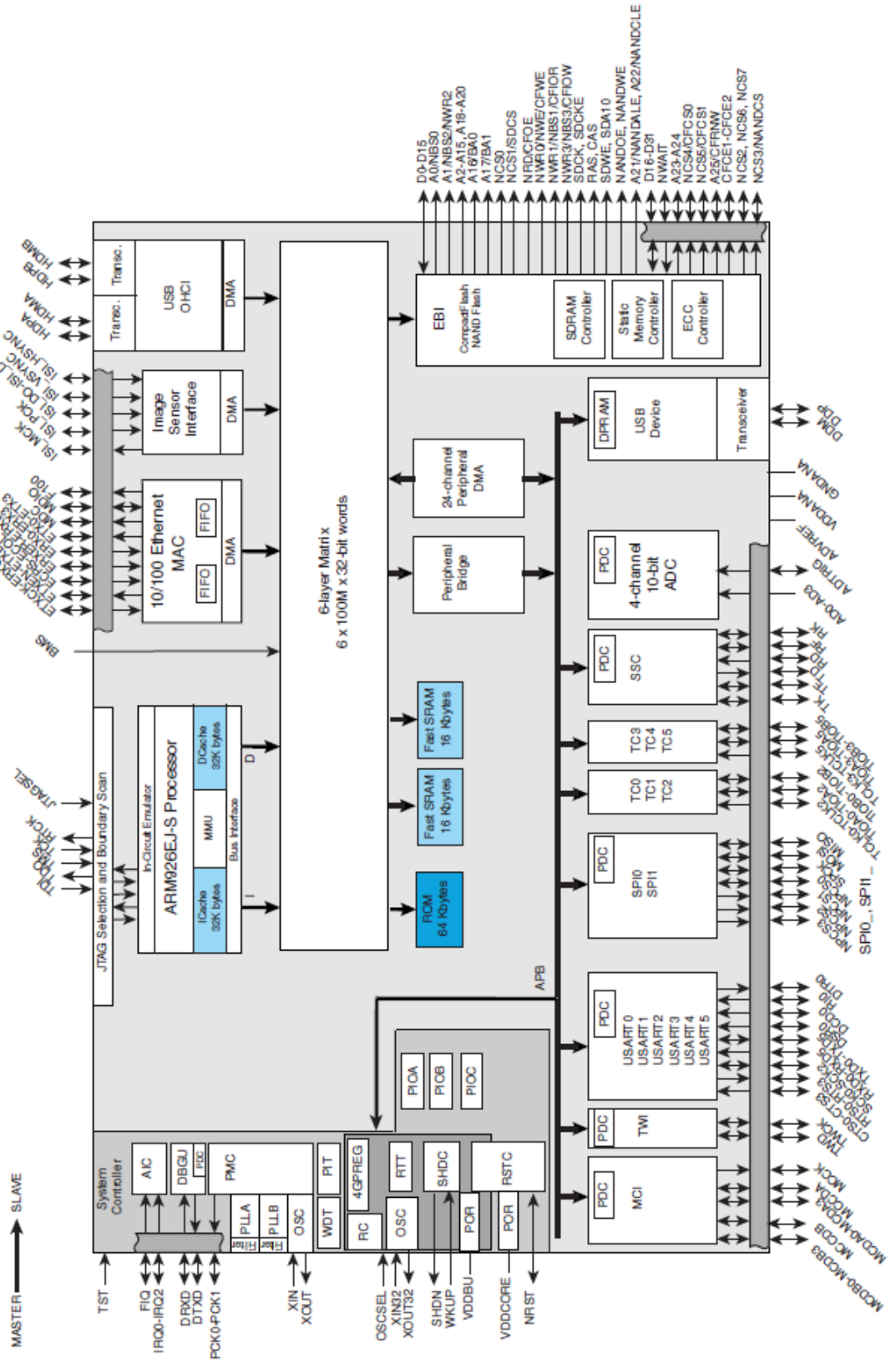


Figure 5.5: AT91SAM9G20 block diagram.

5.1.4 SPI and Flash Organization

The serial flash in our system is a 2-MBytes CMOS serial flash MX25L1605D from Micron. Its pins are fully compatible with the processor's serial peripheral interface (SPI). This interface includes two data lines and two control lines. The MOSI and MISO lines supply the data shift between the master (processor) and the slave (serial flash). The other two control lines are clock and chip select respectively. The master and the slave have a serial shift register each. The master starts one transmission through writing data to its serial shift register. The data is then transmitted to the slave via MOSI line and the slave transmits the data in its register back through MISO line simultaneously. Figure 5.6 shows the basic process of SPI data transmission. Figure 5.7 shows the connections of the SPI between the processor and the flash.

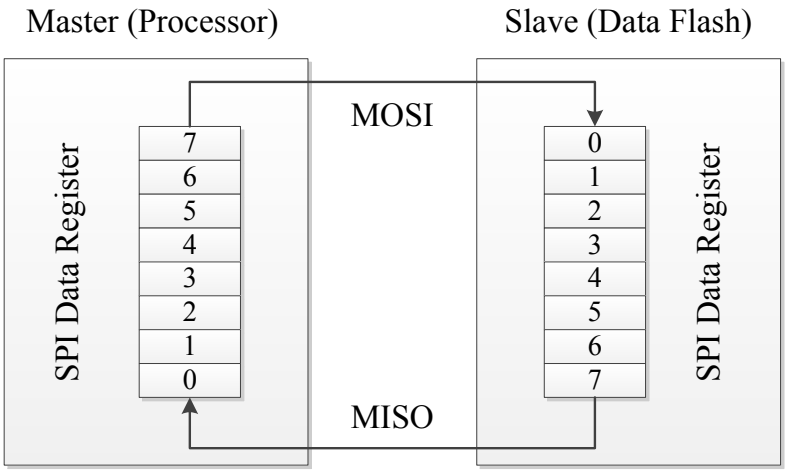


Figure 5.6: SPI data transmission.

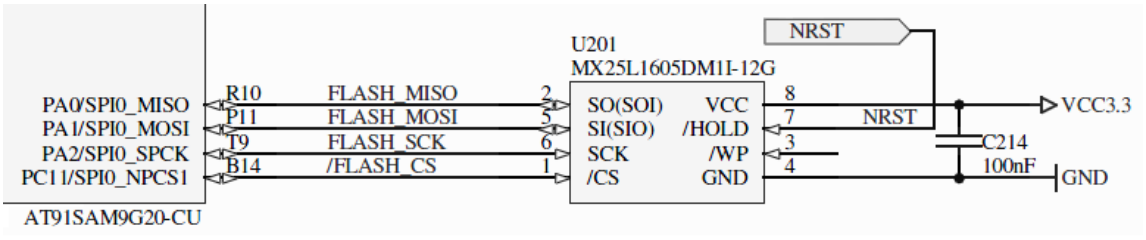


Figure 5.7: SPI connections between MX25L1605 and AT91SAM9G20.

The SPI timings and operation modes have been designated by SPI controller of the processor. What needs to be implemented is to initialize related registers such as control register, data receive and transfer register, interrupt enable and disable register and chip select register etc. Each complete SPI transmission consists of command and related data packet, which are stored in a 16-bit transmit data register successively. The data is transmitted from master to slave, processed in slave and then the new data will be sent back to a 16-bit receive data register of the master. If the command is a write operation, the received data can be ignored. However such transmission is not perfect for the reason that in data transferring by master mode the processor must check the transmit empty status or trigger the interrupt of this status frequently and refill the transmit buffer if needed to avoid the transmission delay, i.e. if the next 16-bit data cannot be sent to the transmit buffer in time, the transmission is delayed. One better approach is to use its Direct Memory Access (DMA). If DMA is enabled in SPI transmission, the corresponding DMA channel will receive a request from its SPI controller and send an answer back. Subsequently the pointer address of the command and related data packet will be copied to 32-bit DMA register for SPI transmission. In this case the data is transmitted continuously without the check of transmit empty status by the processor, which accelerates the SPI speed especially in high frequency. Figure 5.8 shows the process of SPI data transmission with DMA.

The flash organization is shown in Figure 5.9. The bootloader program locates at the starting address of the flash, which occupies 8 Kbytes memory. The next sector is allocated to configuration, which contains the diverse parameters with which the fingerprint algorithm can be properly executed. The AFIS software is stored in the next several sectors, whose size is dependent on the complexity of the fingerprint verification and identification algorithm and the additional usages. The reserved sector is for some special usage and the backup sector for a temporary backup of template copy operation. The residual memory is used to store fingerprint templates.

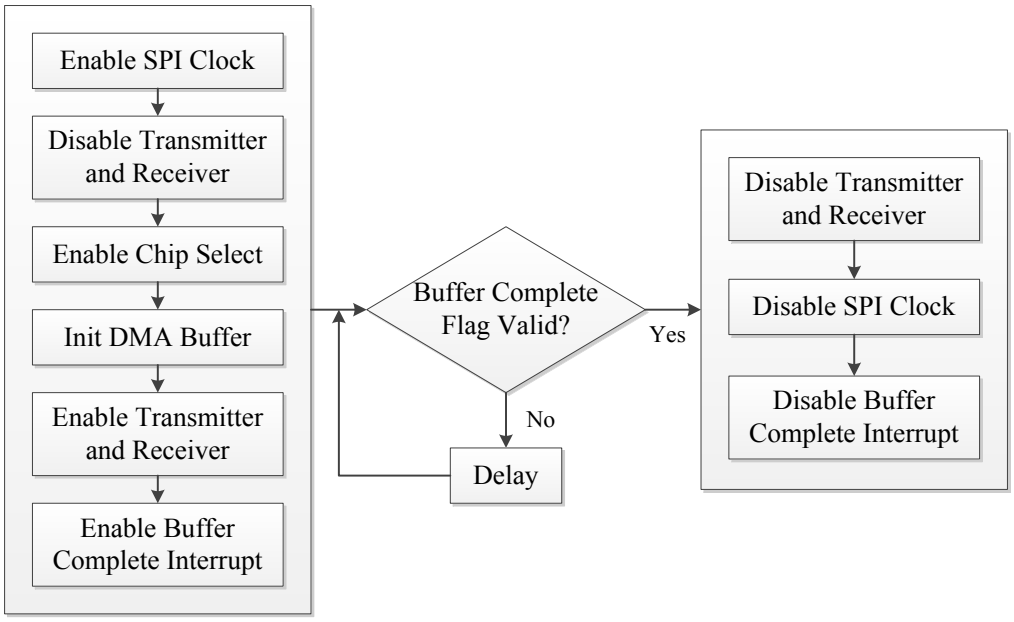


Figure 5.8: SPI data transmission with DMA.

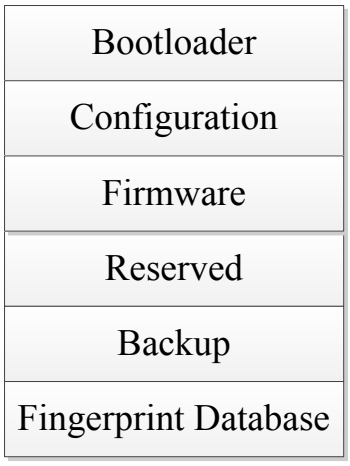


Figure 5.9: Serial flash organizations.

5.1.5 Serial Interface

The AT91SAM9G20 processor supplies five standard Universal Synchronous Receiver Transmitters – USART, with which the system can communicate with other devices e.g. PC or relay board. In view of various interfaces of the devices

and the transmission distance, the TTL level USART interface with insufficient signal intensity and anti-interference ability should be converted to RS-485 differential signal with a converter. Figure 5.10 shows a 3.3-V RS-485 transceiver of TI and its schematic diagram. Through setting or clearing the RTS0 pin to enable delivery or receive, data can be sent with TXD0 or received with RXD0 in half-duplex mode.

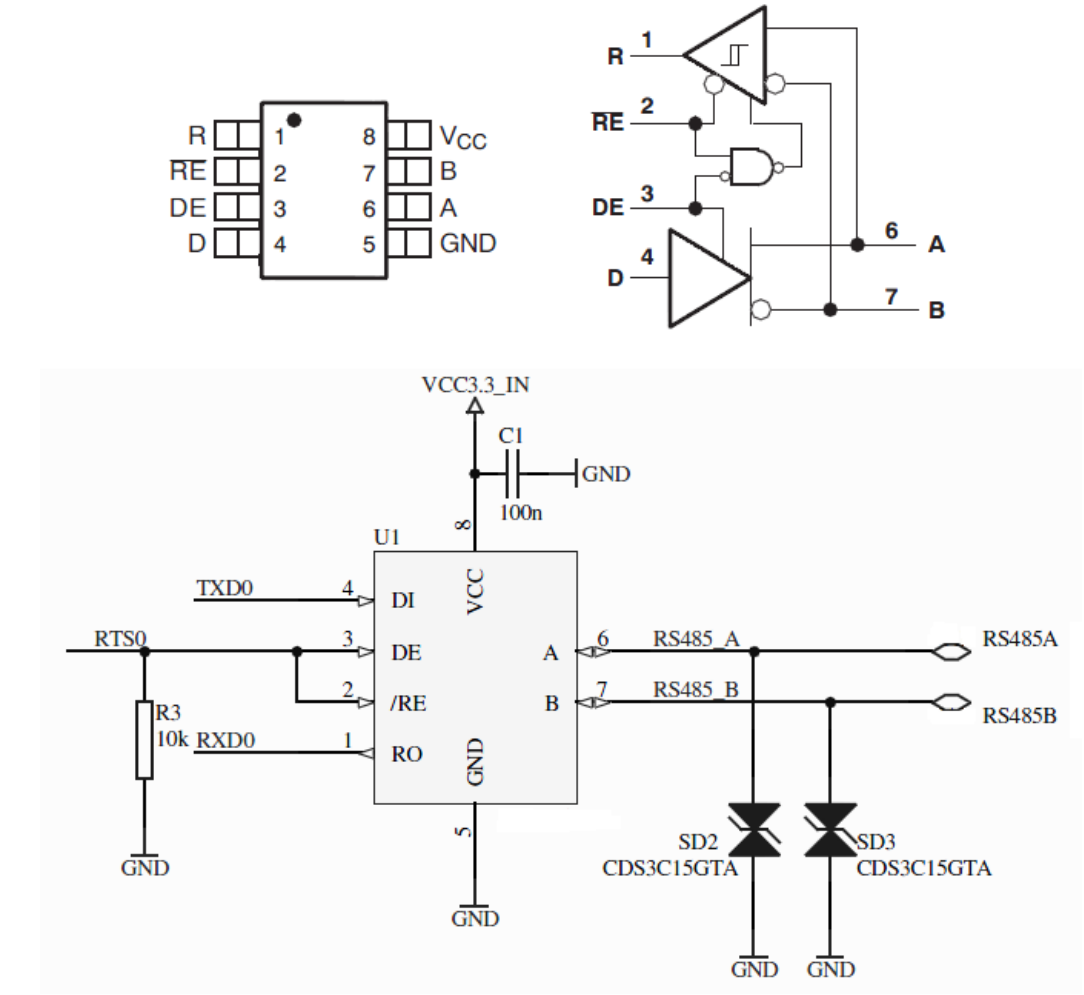
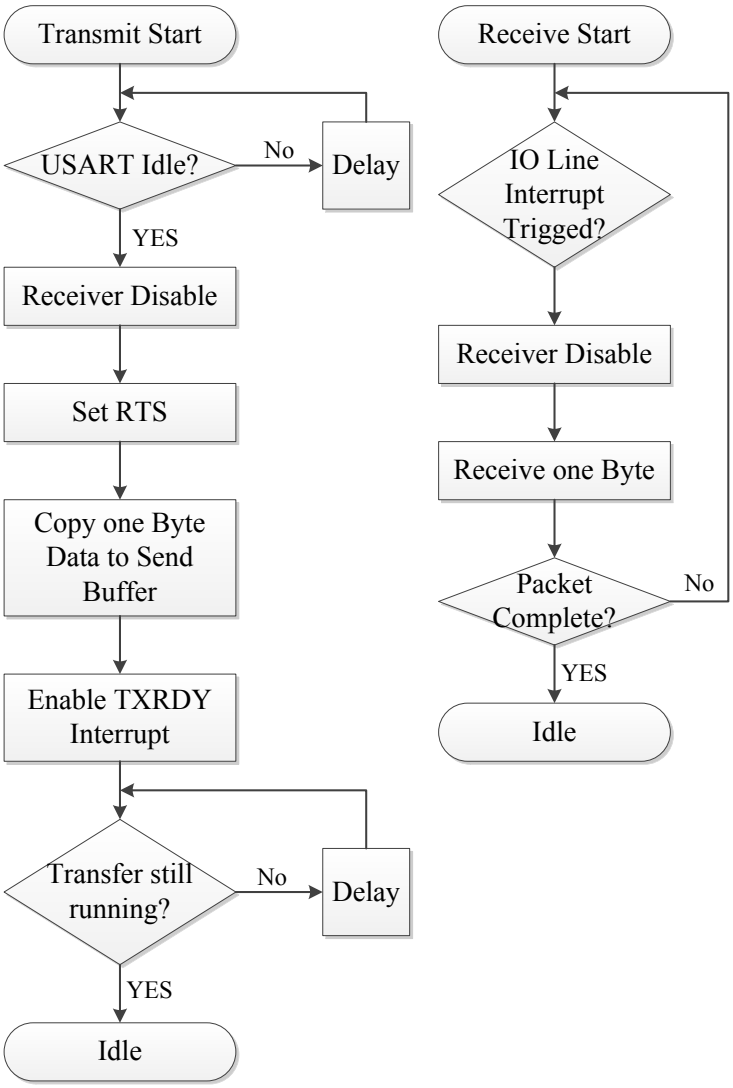


Figure 5.10: USART to RS-485 transceiver.

The data transmission can be trigged from either the kern system or another controller which is connected to it, e.g. this system can be set as a master or a slave device. If one request packet is to send out, the USART controller communicates actively through enabling two different bits TXRDY (Transmit

Ready) and RXRDY (Receive Ready) of USART Interrupt Enable Register alternatively to send this request packet and receive the corresponding answer packet (The RXRDY bit should always enabled unless there is data to send out since the slave cannot evaluate when a request packet is sent from the master.). If the request packet is sent from the other side, the signal will be firstly scanned by an IO interrupt in order to start a data receive progress. If no data exists on the TXD or RXD line, the USART will be switched to idle state through clearing USART interrupt and enabling IO interrupt. The details of the data transmission and the related interrupt can be referred in Figure 5.11.



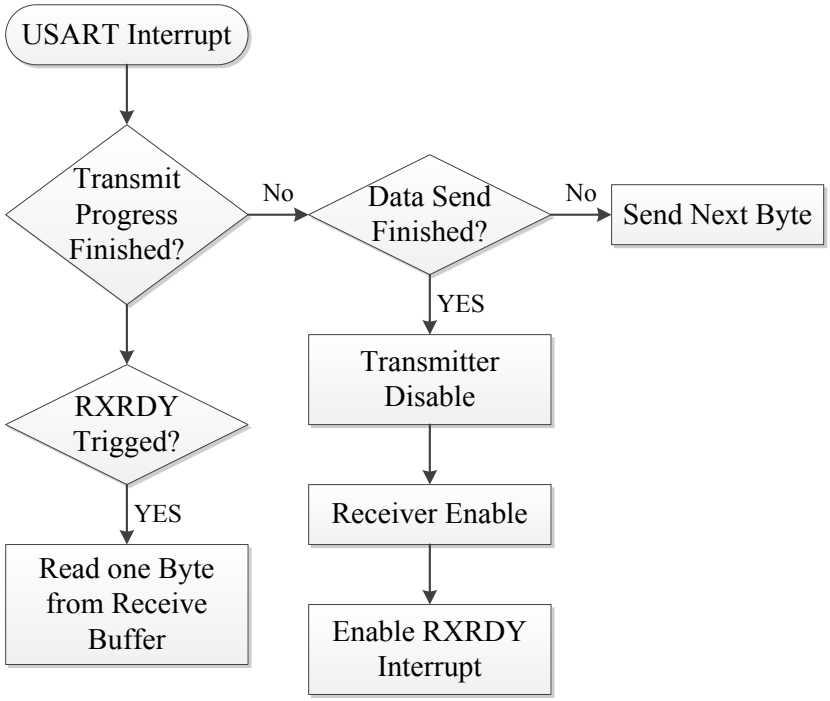


Figure 5.11: Flow diagrams of USART communications.

The data packet is implemented with 8 bits data; LSB (Least Significant Byte) (bit 0) first; no parity and 1 stop bits at various baud rates from 9600 to 115200. The data structure is shown in Figure 5.12. The packet length may vary but is limited to a maximum of 1024 data bytes.

START	LENGTH	CMD	SENDER	DEST	DATA1	...	DATAx	ERRCHK

Figure 5.12: Data structure of USART.

START

1 byte start symbol marks the beginning of a packet.

LENGTH

2 bytes packet length (bits 0…7 first), contains the number of data bytes DATA in packet. Not counted are START, LENGTH, CMD, SENDER, DESTINATION and

ERRCHK, so a packet with 30 data bytes has a length value of 30. Maximum packet length is 1024.

CMD

1 byte contains command for receiver.

SENDER

1 byte contains sender's address.

Address range: 0x00 … 0xFF (The Modules share range 0x01 … 0xEF; Host PC in bus mode uses 0x00; 0xF0 … 0xFF reserved)

DESTINATION

1 byte contains receiver's address.

Address range: 0x00 … 0xFF (Modules share range 0x01 … 0xEF; Host PC in bus mode uses 0x00; 0xF0 … 0xFF reserved)

DATA 1…x

0 … x bytes contain the data to be transmitted.

A packet can consist of 0 to 1024 data bytes.

ERRCHK

1 byte contains a CRC check sum for error detection.

The check sum is computed with the functions of CRC. All packet bytes in the given order (START, LENGTH, CMD, SENDER, DESTINATION, and DATA (1) … DATA (x)) are used for CRC calculation.

To ensure the reliability of the transmission, the receiver sends an ACK (acknowledge) which is not a packet but a single byte after the reception of an error-free packet. If the analysis of the check sum shows an erroneous transmission, the receiver ignores the packet. After a timeout the sender has to

retransmit the packet. If the module is busy it replies a NAK (negative acknowledge). In this case the sender can retransmit the packet immediately until an ACK is replied. If the sender receives an undefined or no answer, the packet is retransmitted up to 2 times. In case the sender does not receive a positive acknowledgement again, the process is terminated as the connection or the receiver might be faulty. If the sender receives an ACK and the packet from the sender requires an answer packet, the sender has to wait up to 5 seconds for the answer.

5.1.6 I^2C

I^2C is the kind of protocol which allows multi-master on the same bus and has no arbitrary logic but only two signal lines – SDA (serial data) and SCL (serial clock). It stipulates that each I^2C device has a unique 7-bit address, a data frame has 8 bit length and some bits in data frame are used for the controlling signals such as start, end, the direction of R/W, and acknowledge. According to its data rate, I^2C can be divided into various modes e.g. standard (100 kb/s), fast (400 kb/s), high speed (3.4 Mb/s) and some other variants.

There are two different I^2C components in the relay board: a serial EEPROM (24C01) and an 8-bit I/O expander for I^2C bus (PCF8574T). The EEPROM stores the matching code between the kern module and this relay board; the I/O expander analyses I^2C signal and converts it to 8-bits parallel signal to control different relays. They are connected to the same I^2C bus, which is shown in Figure 5.13. Pins A0-A2 are the input address lines of the chips, which are the low 3 bits of the complete address. The high 4 bits of the address of each chip have been programmed by the manufacture in advance, therefore these two chips can be easily differentiated according to these 4 bits and the pins A0-A2 are simply connected to the ground.

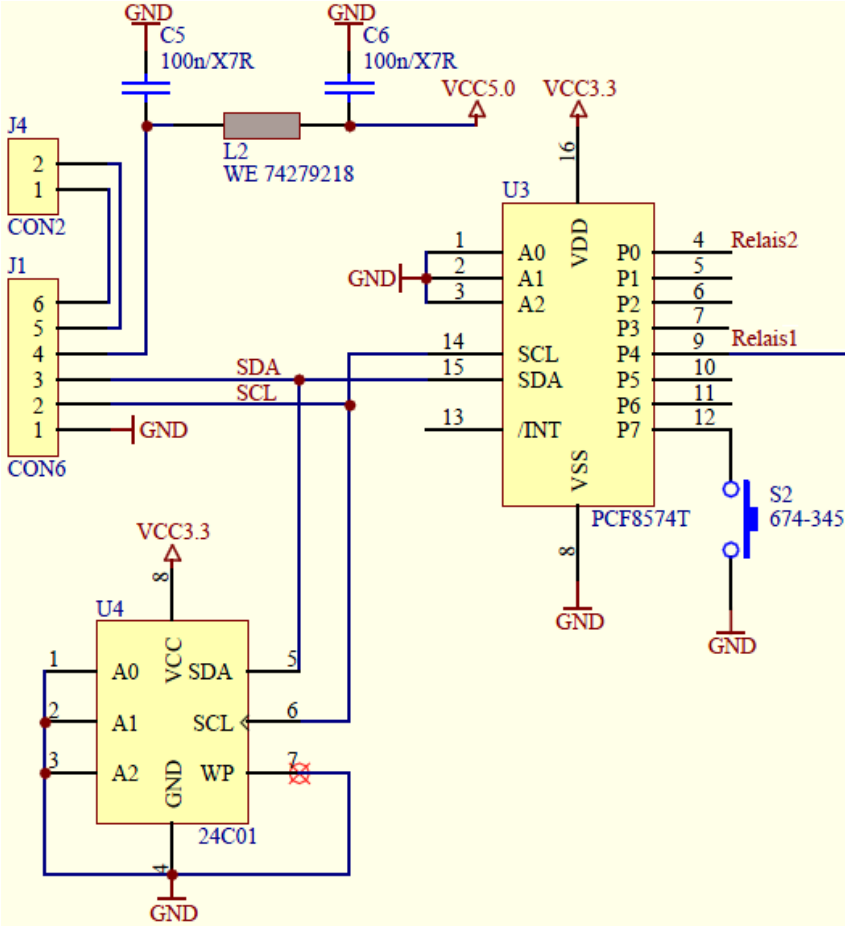


Figure 5.13: Schematic diagram of the I^{2}C relay board.

In practice software simulated I^{2}C with GPIO has been implemented instead of hardware I^{2}C on the kern module side, which has better operability and stability. The substance of the implementation is to simulate the timings of I^{2}C bus. In idle state SDA and SCL are pulled up by pull-up resistors. The master can control the both lines, but the slave can only control the SDA. The detailed timings are described as follows. When the master issues a START, it pulls down the SDA line, which will be detected by the slave and regarded as a signal for the slave to prepare for data receive. Then the master issues 1 Byte data bitwise through pulling down the SCL, updating 1 bit on the SDA and pulling up the SCL. When the slave detects the change of the SCL, it receives this bit from the SDA at once and stores it in its shift register. The master will pull down the SCL again for the

transfer of the next bit until 8 bits data are transferred. At this moment, SCL is at low level for a short time and SDA is at high level i.e. released. The slave will now issue an ACK through pulling down the SDA and holding on this signal until the master switches the level of SCL to high and then low again, which means that the master has received the ACK or NACK. If no more data is to issue, the master prepares a STOP signal to end the transmission through pulling down SDA in time of SCL at low level, and then pulling up the SCL and SDA in the end. The SCL and SDA return to idle state.

5.1.7 Infrared Keypad

The infrared component is an additional block of this system, which realizes some basic functions e.g. fingerprint enrollment and deletion, database administration and parameter modifications etc. instead of being executed with the commands from PC through RS485 interface. On the kern module side a miniaturized infrared receiver from VISHAY shown in Figure 5.14 is used for infrared signal detection, which has 38 kHz center frequency of the band-pass, and supports different kinds of infrared remote control transmission ICs such as NEC, RC5 and RC6 etc. On the remote control side an infrared keypad also shown in Figure 5.14 is designed integrated with a remote control transmission CMOS IC using the NEC transmission format.

In general there are two different infrared remote control formats: Pulse Width Modulation (PWM) and Pulse Position Modulation (PPM). PWM represents the “0” and “1” using the duty cycle of the infrared carrier, however PPM using the detection of the rising or falling edge of one bit period of the infrared carrier. Figure 5.15 shows the transmission code of the NEC format which is used in our infrared Keypad. The NEC format has the following characteristics: the frequency of the carrier wave is 38 kHz and duty cycle is 1/3; the interval of the Leader Code is 9ms + 4.5ms; the Custom Code has 16 bits; the Data Code has 16 bits; both the Custom Code and the Data Code include an 8-bits original code and its

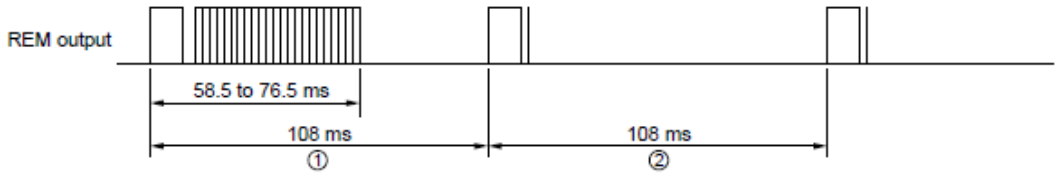
reversed 8-bits code. If a key is kept pressed, only the Leader Code will be iteratively transmitted. Each 8-bits data unit is transmitted from LSB to MSB, and so each data byte should be bit-reversed after reception.

Figure 5.14: Infrared receiver and remote controller.

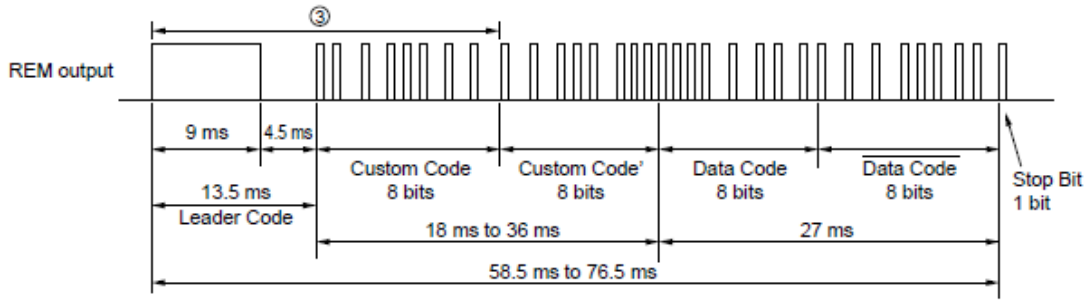
To capture the infrared signal, two different kinds of approaches can be used, which are polling and interrupt. By polling mode, the period of the polling should be firstly decided. Since the least interval of the signal is 0.56ms, the polling interval should not be greater than 0.56ms. The system has a general timer, which has 16 kHz in frequency. Therefore a suitable polling interval can be obtained through scaling this timer with a factor such as 4 (4 / (16 kHz) = 0.25ms). If the level is unchanged, only the counter for the level will be accumulated. If the level alters, the counter will be analyzed with the level before and after this alternation. By interrupt mode, the extern interrupt of the GPIO connected to the output pin of the receiver should be set as edge-trigged. Through measuring the interval between the rising edge and the next falling edge, the length of a logic 1 can be extracted and vice versa.

- **When fosc = 455 kHz**

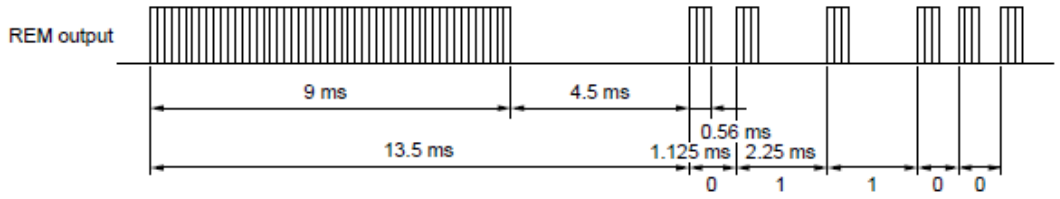
(1) Remote (REM) output (from stage ②, transmission occurs only when key is kept depressed)



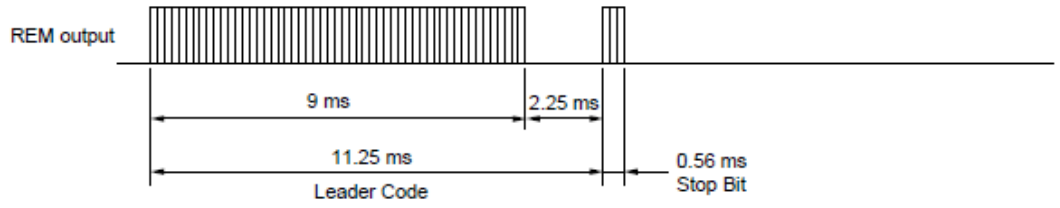
(2) Magnification of stage ①



(3) Magnification of waveform ③



(4) Magnification of waveform ②



(5) Carrier waveform (Magnification of HIGH period of codes)

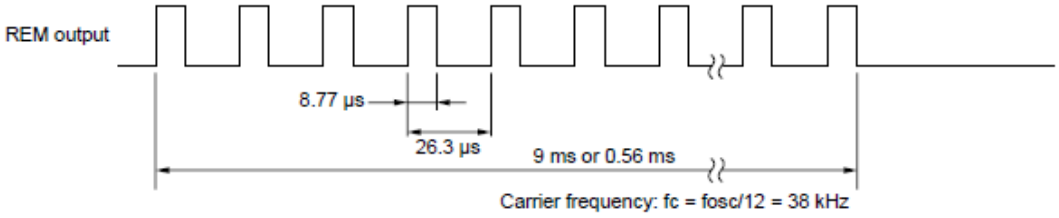


Figure 5.15: Remote output waveform of NEC format.

5.2 Software Implementation and Optimization

5.2.1 System Boot

In PC system the hardware initializations and configurations are performed by BIOS (Basic Input / Output System). However in the embedded system such program, also called bootloader, cannot entirely completed by the chip fabricators for the reason that even the same processor can be applied in many embedded environments with different peripherals which need different initialization codes. Therefore one microcontroller can only have a serial of general initialization commands programmed in internal ROM, with which a second-level bootloader or the main program can be successfully loaded and executed.

The boot strategies of AT91SAM9G20 can be described in Table 5.1. The system always boots at address 0x0 and which memory is assigned to this address depends on the pin BMS (Boot Mode Select) and Remap operation in the Remap Control Register. Therefore the state of the BMS pin by reset decides the boot mode, since the Remap can only be done after the system boots. If BMS is 0, the system boots with 16-bit data bus on 16-bit non-volatile memory, which is not compatible with our serial flash. If BMS is 1, which has been implemented through pull-up in our system, ROM occupies the address 0x0 and 0x100000 by start, and the pre-programmed boot program is executed to search available bootloader from supported interface such as serial flash, NandFlash, SD Card or EEPROM etc. Then the bootloader will be downloaded and executed from the external storage media into internal SRAM0 by the special initialization commands mentioned above. The code size of the bootloader should not exceed the size of the SRAM0. For this reason the size is pre-defined in the sequence of the exception vectors located at the first 8 words in serial flash. By serial flash boot, these 8 words will be firstly read into SRAM and checked whether they are valid exception vectors shown in Figure 5.16 and have valid size. If a valid

sequence is found, the bootloader will be loaded into SRAM0 and executed from the address 0x0 in SRAM0 after the boot program issues a Remap command.

Address	Remap = 0		Remap = 1
	BMS = 1	BMS = 0	
0x00000000	ROM	EBI_NCS0	SRAM0 16K
0x00100000	ROM		
0x00200000	SRAM0 16K		
0x00300000	SRAM1 16K		
0x00500000	USB Host User Interface		

Table 5.1: Internal memory mapping

```
SECTION .vectors:CODE:NOROOT(2)
resetVector:
  LDR     pc, =resetHandler       ; Reset
  LDR     pc, Undefined_Addr      ; Undefined instructions
  LDR     pc, SWI_Addr            ; Software interrupt (SWI/SYS)
  LDR     pc, Prefetch_Addr       ; Prefetch abort
  LDR     pc, Abort_Addr          ; Data abort
  DC32    0x4000                  ; Bootloader size (16 Kbyte)
  LDR     pc, =irqHandler         ; IRQ
  LDR     pc, FIQ_Addr            ; FIQ
```

Figure 5.16: Interrupt vector table.

The bootloader executed in SRAM0 includes the following steps. After the interrupt vector table is defined at the address 0x0 in SRAM0, some hardware initializations should be made to switch the running bootloader program from slow clock (32,768Hz) to fast clock (132MHz Master Clock) through setting PRES and PDIV bits in Master Clock register. Then the interrupt stack and system stack are set up under corresponding special arm modes, since such operations of Current Program Status Register (CPSR) cannot be processed

under user mode. Finally the program pointer will branch to the main program of the bootloader, in which SDRAM and serial flash are configured and the main embedded fingerprint firmware will be copied from flash to the SDRAM and set as the next level program to execute.

5.2.2 MMU and Caches

The ARM926EJ-S has an ARM architecture v5 MMU and contains a 32KB Instruction Cache (ICache), a 32KB Data Cache (DCache) and a write buffer. This MMU has standard mapping size of 1MB (sections), 64KB (large pages), 4KB (small pages) and 1KB (tiny pages)[97]. Because this core is a 32-bit processor, its virtual address space has the size of $2^{32} = 4G$ and the whole space can be divided into $\frac{4G}{1M} = 4K$ sections. Each section is recorded with 4 bytes shown in Figure 5.17. Therefore a block of memory with the length of $4K \times 4 = 16K$ should be allocated to this TTB (Translation Table Base) used by MMU. Due to the reason that other descriptors except section descriptor have been removed in the ARMv6 architecture, to ease the porting of code in future, only section descriptor will be used in our system, which can be differentiated from other descriptors with the last two bits - 10 of this descriptor.

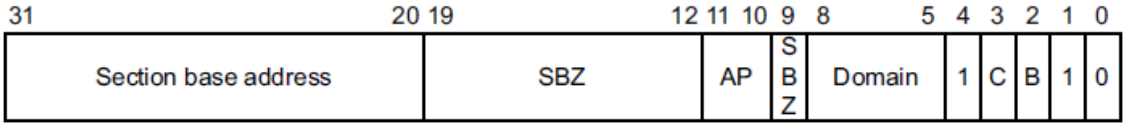


Figure 5.17: MMU section descriptor.

In Figure 5.17 the three fields “Section base address”, “C” and “B” must be correctly configured for each section. The “Section base address” occupies 12 bits, which stores the most important 12 bits of the address of each section to differentiate from other sections. The combination “C” and “B” field indicates that this section may be write-back cacheable, write-through cacheable, noncached

buffered, or noncached nonbuffered section. The difference between the write-back and the write through operations is that, when a cache write hit occurs, the write-through operation updates not only the DCache line but also the external memory, which avoids inconsistency between memory and cache at cost of efficiency (writing speed and bus busy time).

Memory	Address Range	Cacheable	Buffered
Boot	0x00000000 ... 0x000FFFFF	0	0
ROM	0x00100000 ... 0x001FFFFF		
SRAM0	0x00200000 ... 0x002FFFFF		
SRAM1	0x00300000 ... 0x003FFFFF		
UHP	0x00500000 ... 0x005FFFFF		
SDRAM (8MB)	0x20000000 ... 0x200FFFFF	1	1
	0x20100000 ... 0x201FFFFF		
	0x20200000 ... 0x202FFFFF		
	0x20300000 ... 0x203FFFFF		
	0x20400000 ... 0x204FFFFF		
	0x20500000 ... 0x205FFFFF		
	0x20600000 ... 0x206FFFFF		
	0x20700000 ... 0x207FFFFF		
Internal Peripherals	0xFFF00000 ... 0xFFFFFFFF	0	0

Table 5.2: Translation table definition

Table 5.2 shows the related memory spaces with cache and buffer features in Translation table. After system boot code and data will be transferred to SDRAM and executed in SDRAM. The complete SDRAM is therefore set as cacheable and write-back enabled except some buffers for special usage such as DMA transfer buffer for serial flash. The DMA controller serves as a master device connected to the AHB bus like ARM core. They can both access the internal or external via APB bridge memory interface shown in Figure 5.4. On a single-layer AHB bus such simultaneous accesses are exclusive. If the destination address of one DMA transmission triggers cache hit, one conflict will occur. In view of this

reason, the DMA transfer buffer can be allocated in SRAM0 which is noncached buffered. Another method to avoid this conflict is to flush the related source cached address before transmitting and flush the related destination cache address after transmitting. Another memory space, which should not be cached or buffered, is internal peripherals. Since write-back operation has delay and two successive reading from the same I/O may be different, these registers should be set as noncached and nonbuffered.

FVC2006 DB3_B	ICache = 0 DCache = 0	ICache = 1 DCache = 0	ICache = 1 DCache = 1
1_1.bmp	14.76s	6.66s	0.78s
2_1.bmp	13.71s	6.00s	0.76s
3_1.bmp	11.94s	5.42s	0.63s
4_1.bmp	9.51s	4.30s	0.52s
5_1.bmp	8.97s	4.09s	0.49s
6_1.bmp	9.73s	4.45s	0.50s

Table 5.3: Comparisons of the encoding performance with or without ICache and DCache

ICache and DCache can be independently enabled and disabled. If ICache is enabled, all memory space used by CPU read instruction can be cached. If DCache is enabled, MMU must also be enabled since its translation table manages the whole memory space to decide which section should be cached or buffered. Table 5.3 indicates the various performance of the ARM926EJ-S core with or without ICache and DCache. Several images are selected from FVC2006 DB3_B database, which have 500 dpi with image size 400×500. On average, ICache doubles the encoding speed meanwhile DCache and MMU increase the encoding speed by about 10 times. With enabled ICache and DCache this system can achieve real-time requirements.

5.2.3 Interrupt Handling

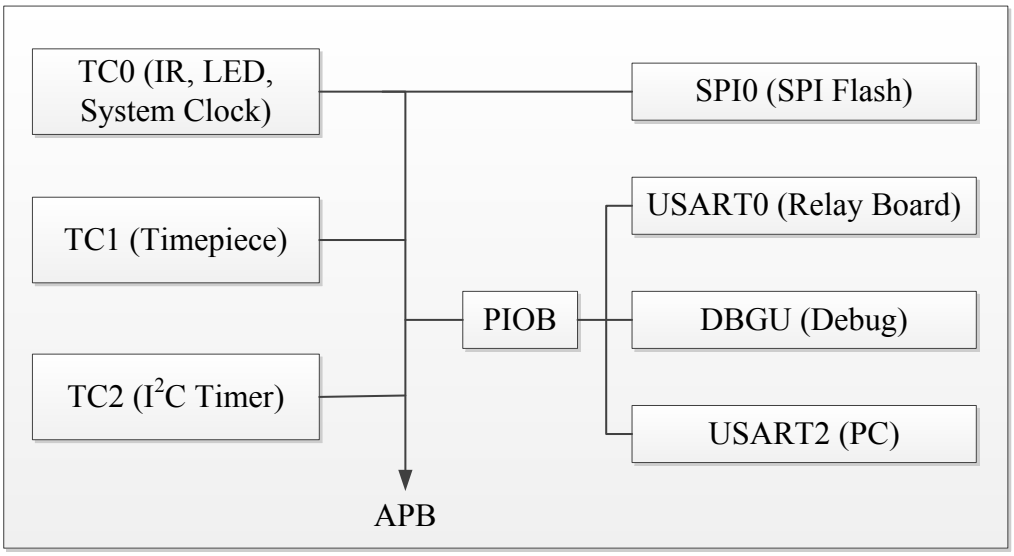


Figure 5.18: Enabled interrupts in BioKey4000.

As shown in Figure 5.18 BioKey4000 uses different interrupts to communicate with external devices or implement internal functions. If several interrupt sources are pending and enabled, they should be processed sequentially according to their significance. Some external functions such as data transmission via USART are bit-sensitive and therefore should not be delayed by other interrupts.

The ARM9G20 has an 8-level priority controller to drive the IRQ line of the processor. Each interrupt source can have a programmable priority level. To avoid losing packet by data transmission, external interrupts, especially USART2, are given relative higher priority. During the execution of the lower priority interrupts, such high priority interrupt can be inserted and handled in advance with a nested interrupt handler, in which the current interrupt number and its priority level are pushed into an embedded hardware stack and later restored when the higher priority interrupt is finished.

5.3 Summary

The BioKey4000 module is a standalone fingerprint identification and verification system. It uses ATMEL FingerChip AT77C102B to acquire fingerprint image and process this image with ARM9G20 processor. The module extends a 32-bits 8-MByte SDRAM and a 2-MByte serial flash for program execution and storage respectively. RS485 hardware driver is also integrated optionally, with which BioKey4000 can communicate with PC for fingerprint database management and software update and with relay board to supply identification signal for access control. In addition an infrared block is developed for operating convenience.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis main work can be described in following sequence:

i. Fingerprint image enhancement – in order to give a better representation of the input image. Contrast extending, normalization, and filters with both frequency-selective and orientation-selective properties will be discussed.

ii. Features extractions – obtaining features such as the position, angle and type of the minutiae and also ridge distribution from binary image. In this part we have used a special kind of method to locate minutiae, which is based on the contour of the ridges, instead of traditional method from thinning images. Through enclosing each singular ridge, we have located the possible minutiae positions at the turning points. After that, minutiae post-processing has been used to eliminate the spurious minutiae.

iii. Features matching – based mainly on matching of feature vector pairs theory making the decision between two fingerprint images whether they are from the same finger. According to the angel where the maximum of the number of matched feature vector pairs reach, the translation and

rotation parameters will be extracted. Thereafter the similarity between two templates is computed based on our multiple-matching algorithm.

iv. Algorithms implementation in embedded environment – ARM core based system is designed, with which real-time fingerprint identification and verification operation is expected. Before fingerprint verification and identification algorithm is applied in our embedded systems, some important hardware devices should be adequately selected.

Fingerprint sensor is the first one. Although optical sensors can obtain better image quality and resolution, its size and cost restrict its application in embedded devices. As an alternative solid-state sensors which overcome the size and cost problems are widely used in embedded devices. The swipe sensor selected from ATMEL is part of this kind and based on thermal technology. It has limited width which might be even narrower than some fingers. Therefore the reconstructed fingerprint image maybe be only part of instead of a whole finger.

The second important part in the system is the processor. ARM9g20 (400MHZ) is used to build hardware environments, with which a relative good performance of fingerprint verification and identification is obtained.

v. Finally we have made some statistical test with standard databases e.g. FVC2004, FVC2006 and also captured databases by ourselves. The statistic results of our experiments are built based on the above hardware platform, which has shown that the system can obtain satisfying performance in various databases.

6.2 Future Work

To make this fingerprint identification and verification system more robust and better for use, we believe there are still a lot of work to do. According to the significance, some of work is listed as follows.

i. Real-time is a most important feature of this system. In order to accomplish fingerprint identification between an actual template and a large template database, the time of each single matching is relative crucial. To optimize the matching algorithm further can realize a real-time identification with larger database.

ii. A typical ATMEL solid-state sensor is used in this thesis. There are still some different kinds of sensors widely used. The algorithm based on ATMEL sensor cannot process the images generated from other sensors very well. Aiming at those images with small size, few minutiae and bad contrast, the algorithm still needs to be corrected.

iii. One optional further development is to develop a LCD block on BioKey4000, since the acknowledgement of menus and messages can only be shown from three different LEDs currently. Some conflict occurs occasionally.

iv. Another optional further development is to develop an Ethernet block based on TCP/IP protocol directly on BioKey4000 Module, with which this system can be accessed into a local network as a network terminal.

Appendix A: FVC – Fingerprint Verification Competition

Fingerprint Verification Competition (FVC) is an international competition focused on fingerprint verification software assessment. The registered participants can test their algorithms and adjust the parameters in advance according to a subset of the database, acquired with various sensors, for competition. Enroll and Match algorithms should be compiled separately to executable files to submit, which will be used for the competition with the whole database. The result of this competition presents an overview of the state-of-the-art in fingerprint techniques. Till now FVC has been held four times in 2000, 2002, 2004 and 2006 respectively.

The software and hardware environment for the competition in 2006 is Windows XP Professional O.S. on PC INTEL PENTIUM 4 - 3.20 GHz - 1.00 GB RAM. The competition is divided into two categories: Open and Light. In Open category memory requirements and template size have no limits. The maximum enroll time is 5 seconds and the maximum matching time is 3 seconds. In Light category the maximum memory that can be allocated by the processes is 4M-

Bytes. The maximum template size is 2K-Bytes. The maximum enroll time is 0.3 seconds and the maximum matching time is 0.1 seconds. In the earlier competitions such limitations vary with the software and hardware environment at that time.

Competition	**Number of databases**	**Size of each Database:** A - Evaluation set B - Training set	**Notes**
FVC2000	4	A: 100 × 8 B: 10 × 8	- Volunteers are mainly unhabituated students. - Two sessions, no quality-check. - Low/Medium difficulty (DB1, DB2, DB4); Medium/High difficulty (DB3).
FVC2002	4	A: 100 × 8 B: 10 × 8	- Volunteers are mainly unhabituated students. - Three sessions, no quality-check. - Voluntarily exaggerated perturbations: displacement, rotation, wetness and dryness. - Low difficulty (DB1, DB2, DB3, DB4).
FVC2004	4	A: 100 × 8 B: 10 × 8	- Volunteers are mainly unhabituated students. - Three sessions, no quality-check. - Voluntarily exaggerated perturbations: distortion, wetness and dryness. - Medium difficulty (DB1, DB2, DB3, DB4).
FVC2006	4	A: 140 × 12 B: 10 × 12	- Heterogeneous population also includes unhabituated manual workers and elderly people. No quality check. - The final datasets were selected from a larger database by choosing the most difficult fingerprints according to a quality index. - High difficulty (DB1), Medium difficulty (DB3), Low difficulty (DB2, DB4).

Table A.1: A summary of FVC database. In the database size the notation M×N, denotes M fingers and N samples per finger. In all the competitions the first three databases were acquired by selecting commercial scanners of different types, including large-*area optical, small area optical, solid state capacitive and thermal sweep. The fourth database was synthetically generated by using the SFinGe tool.* [73]

Every competition contains four distinct databases for the benchmark: DB1, DB2, DB3 and DB4, which are acquired from heterogeneous people and different sensors (see Table A.1). Each database has two subsets DB_A and DB_B. DB_A contain more fingers and samples and are used for the algorithm performance evaluation. DB_B contains fewer fingers and are used for parameter tuning by the participants. All fingerprint images in databases are stored in 256 gray-levels, uncompressed BMP or Tiff format.

The competitions use each algorithm with each database to compute the False Non Match Rate FNMR (also referred as False Rejection Rate - FRR) through matching each sample in the subset DB_A against the remaining samples of the same finger, and to compute the False Match Rate FMR (also referred as False Acceptance Rate – FAR) through matching the first sample of each finger in the subset DB_A against the first sample of the remaining fingers in this subset. Furthermore other important indicators are also reported:

- REJENROLL (Number of rejected fingerprints during enrollment)
- REJNGRA (Number of rejected fingerprints during genuine matches)
- REJNIRA (Number of rejected fingerprints during impostor matches)
- Impostor and Genuine score distributions
- FMR(t)/FNMR(t) curves, where t is the acceptance threshold
- ROC(t) curve
- EER (equal-error-rate)
- EER* (the value that EER would take if the matching failures were excluded from the computation of FMR and FNMR)
- FMR100 (the lowest FNMR for FMR<=1%)
- FMR1000 (the lowest FNMR for FMR<=0.1%)
- ZeroFMR (the lowest FNMR for FMR=0%)
- ZeroFNMR (the lowest FMR for FNMR=0%)
- Average enrollment time
- Average matching time

- Average and maximum template size
- Maximum amount of memory allocated

Table A.2 shows a comparative summary of the performances (EER) obtained in the past four FVC competitions.

	DB1	DB2	DB3	DB4
FVC2000	2.30%	1.39%	4.34%	3.38%
FVC2002	0.20%	0.17%	0.63%	0.16%
FVC2004	1.61%	2.32%	1.34%	0.81%
FVC2006	5.88%	0.05%	1.59%	0.39%

Table A.2: Average accuracy (EER) of the three best performing algorithms over the different FVC databases. [73]

Bibliography

[1]. Ravi Das, “An Introduction to Biometrics: -A Concise Overview of the Most Important Biometric Technologies”, Keesing Journal of Documents & Identity, Issue 17, 2006.

[2] . A. K. Jain (2004). “An Introduction to Biometric Recognition”, *IEEE Transactions on Circuits and Systems for Video*, VOL. 14, NO. 1, January 2004.

[3]. A. K. Jain and A. Kumar, "Biometrics of Next Generation: An Overview", *Second Generation Biometrics (E. Mordini and D. Tzovaras, Eds.), Springer*, 2011.

[4]. Hong, Wan, and Jain (1998). Hong L., Wan Y., and Jain A.K., “Fingerprint Image Enhancement: Algorithms and Performance Evaluation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 777-789, 1998.

[5]. Mehtre et al. (1987). Mehtre B.M., Murthy N.N., Kapoor S., and Chatterjee B., “Segmentation of Fingerprint Images Using the Directional Image,” *Pattern Recognition*, vol. 20, no. 4, pp. 429-435, 1987.

[6]. Mehtre and Chatterjee (1991). Mehtre B.M. and Chatterjee B., “Segmentation of Fingerprint Images – A Composite Method,” *Pattern Recognition*, vol. 22, no. 4, pp. 381-385, 1989.

[7]. Ratha, Chen, and Jain (1995). Ratha N.K., Chen S.Y., and Jain A.K., “Adaptive Flow Orientation-Based Feature Extraction in Fingerprint Images,” *Pattern Recognition*, vol. 28, no. 11, pp. 1657-1672, 1995.

[8]. Maio and Maltoni (1997). Maio D. and Maltoni D., “Direct Grey-Scale Minutiae Detection in Fingerprints,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 1, 1997.

[9]. Shen, Kot, and Koo (2001). Shen L., Kot A., and Koo W.M., “Quality Measures of Fingerprint Images,” in *Proc. Int. Conf. on Audio- and Video-Based Biometric Person Authentication (3rd)*, pp. 266-271, 2001.

[10]. Pais Barreto Marques and Gay Thome (2005). Pais Barreto Marques A.C. and Gay Thome A.C., “A Neural Network Fingerprint Segmentation Method,” in *Proc. Int. Conf. on Hybrid Intelligent Systems*, 2005.

[11]. Wu, Tulyakov and Govindaraju (2007). Wu C., Tulyakov S. and Govindaraju V., “Robust Point-Based Feature Fingerprint Segmentation Algorithm,” in Proc. Int. Conf. on Biometrics, LNCS 4642, pp. 1095-1103, 2007.

[12]. Bazen and Gerez (2001b). Bazen A.M. and Gerez S.H., “Segmentation of Fingerprint Images,” in *Proc. Workshop on Circuits Systems and Signal Processing (ProRISC 2001)*, 2001b.

[13]. Chen et al. (2004). Chen X., Tian J., Cheng J. and Yang X., “Segmentation of fingerprint images using linear classifier,” *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 4, pp. 480-494, 2004.

[14]. Yin, Wang and Yang (2005). Yin Y., Wang Y. and Yang X., “Fingerprint Image Segmentation Based on Quadric Surface Model,” in *Proc. Int. Conf. on Audio- and Video-Based Biometric Person Authentication (5th)*, pp. 647-655, 2005.

[15]. Zhu et al. (2006). Zhu E., Yin J., Hu C. And Zhang G., “A systematic method for fingerprint ridge orientation and image segmentation,” *Pattern Recognition*, vol. 39, no. 8, pp. 1452-1472, 2006.

[16]. Bazen and Gerez (2002b). Bazen A.M. and Gerez S.H., “Systematic Methods for the Computation of the Directional Fields and Singular Points of Fingerprints,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 905-919, 2002.

[17]. He et al. (2003). He Y., Tian J., Luo X. and Zhang T., “Image enhancement and minutiae matching in fingerprint verification,” *Pattern Recognition Letters*, vol. 24, no. 9, pp. 1349–1360, 2003.

[18]. Oliveira and Leite (2008). Oliveira M.A. and Leite N.J., “A multiscale directional operator and morphological tools for reconnecting broken ridges in fingerprint images,” *Pattern Recognition*, vol. 41, no. 1, pp. 367–377, 2008.

[19]. Sherlock, Monro and Millard (1994). Sherlock B.G., Monro D.M. and Millard K., “Fingerprint enhancement by directional Fourier filtering,” *IEE Proceedings Vision Image and SignalProcessing*, vol. 141, no. 2, pp. 87–94, 1994.

[20]. Jiang (2000). Jiang X., “Fingerprint Image Ridge Frequency Estimation by Higher Order Spectrum,” in *Proc. Int. Conf. on Image Processing*, 2000.

[21]. Almansa and Lindeberg (1997). Almansa A. and Lindeberg T., “Enhancement of fingerprint images using shape-adapted scale-space operators,” in *Gaussian Scale-Space Theory*, J. Sporring,M. Nielsen, L. Florack and P. Johansen (Eds.), Kluwer, New York, pp. 21–30, 1997.

[22]. O’Gorman and Nickerson (1988). O’Gorman L. and Nickerson J., “Matched Filter Design for Fingerprint Image Enhancement,” in *Proc. Int. Conf. on Acoustic Speech and Signal Processing*, pp. 916–919, 1988.

[23]. O’Gorman and Nickerson (1989). O’Gorman L. and Nickerson J.V., “An approach to fingerprint filter design,” *Pattern Recognition*, vol. 22, no. 1, pp. 29–38, 1989.

[24]. Sherlock, Monro and Millard (1992). Sherlock B.G., Monro D.M. and Millard K., “Algorithm for enhancing fingerprint images,” *Electronics Letters*, vol. 28, no. 18, pp. 1720, 1992.

[25]. Daugman (1985). Daugman J.G., “Uncertainty relation for resolution in space, spatial-frequency, and orientation optimized by two-dimensional visual cortical filters,” *Journal Optical Society American*, vol. 2, pp. 1160–1169, 1985.

[26]. Turner M. “Texture discrimination by gabor functions,” *Bio-logical Cybernetics*, 1986, 55: 71-82.

[27]. Greenberg et al. (2000). Greenberg S., Aladjem M., Kogan D. and Dimitrov I., “Fingerprint Image Enhancement Using Filtering Techniques,” in *Proc. Int. Conf. on Pattern Recognition (15th)*, vol. 3, pp. 326–329, 2000.

[28]. Erol, Halici and Ongun (1999). Erol A., Halici U. and Ongun G., “Feature selective filtering for ridge extraction,” in *Intelligent Biometric Techniques in Fingerprint & Face Recognition*, L.C. Jain, U. Halici, I. Hayashi and S.B. Lee (Eds.), CRC Press, Baco Raton, FL, 1999.

[29]. Yang et al. (2003). Yang J., Liu L., Jiang T. and Fan Y., “A modified Gabor filter design method for fingerprint image enhancement,” *Pattern Recognition Letters*, vol. 24, no. 12, pp. 1805–1817, 2003.

[30]. Mehtre (1993). Mehtre B.M., “Fingerprint image analysis for automatic identification,” *Machine Vision and Applications*, vol. 6, no. 2–3, pp. 124–139, 1993.

[31]. Maio and Maltoni (1998). Maio D. and Maltoni D., “Ridge-Line Density Estimation in Digital Images,” in *Proc. Int. Conf. on Pattern Recognition (14th)*, pp. 1654–1658, 1998.

[32]. Wegstein (1982). Wegstein J.H., “An automated fingerprint identification system,” *U.S. Government Publication, U.S. Department of Commerce, National Bureau of Standards*, Washington, DC, 1982.

[33]. Moayer and Fu (1986). Moayer B. and Fu K., “A tree system approach for fingerprint pattern recognition,” *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 8, no. 3, pp. 376–388, 1986.

[34]. Xiao and Raafat (1991). Xiao Q. and Raafat H., “Combining statistical and structural information for fingerprint image processing classification and identification,” in *Pattern Recognition: Architectures Algorithms and Application*, R. Plamondon and H. Cheng (Eds.), World Scientific, Singapore, pp. 335–354, 1991.

[35]. Coetzee and Botha (1993). Coetzee L. and Botha E.C., “Fingerprint recognition in low quality images,” *Pattern Recognition*, vol. 26, no. 10, pp. 1441–1460, 1993.

[36]. Tico and Kuosmanen (1999a). Tico M. and Kuosmanen P., “A Topographic Method for Fingerprint Segmentation,” in *Proc. Int. Conf. on Image Processing*, 1999a.

[37]. Watson, Candela and Grother (1994). Watson C.I., Candela G.I. and Grother P.J., “Comparison of FFT Fingerprint Filtering Methods for Neural Network Classification,” *Tech. Report: NIST TR 5493*, Sept. 1994.

[38]. Abutaleb and Kamel (1999). Abutaleb A.S. and Kamel M., “A genetic algorithm for the estimation of ridges in fingerprints,” *IEEE Transactions on Image Processing*, vol. 8, no. 8, p. 1134, 1999.

[39]. Hung (1993). Hung D.C.D., “Enhancement and feature purification of fingerprint images,” *Pattern Recognition*, vol. 26, no. 11, 1661–1671, 1993.

[40]. Fitz and Green (1996). Fitz A.P. and Green R.J., “Fingerprint classification using hexagonal fast Fourier transform,” *Pattern Recognition*, vol. 29, no. 10, pp. 1587–1597, 1996.

[41]. Wahab, Chin and Tan (1998). Wahab A., Chin S.H. and Tan E.C., “Novel approach to automated fingerprint recognition,” *IEE Proceedings Vision Image and Signal Processing*, vol. 145, no. 3, pp. 160–166, 1998.

[42]. Luo and Tian (2000). Luo X. and Tian J., “Knowledge Based Fingerprint Image Enhancement,” in *Proc. Int. Conf. on Pattern Recognition (*15*th)*, vol. 4, pp. 783–786, 2000.

[43]. Ikeda et al. (2002). Ikeda N., Nakanishi M., Fujii K., Hatano T., Shigematsu S., Adachi T., Okazaki Y. and Kyuragi H., “Fingerprint Image Enhancement by Pixel-Parallel Processing,” in *Proc. Int. Conf. on Pattern Recognition (*16*th)*, vol. 3, pp. 752–755, 2002.

[44]. Chin, Wan and Stover et al. (1987). “A one-pass thinning algorithm and its parallel implementation [J],” *Computer Image Processing*, 1987, 40(1): 30-40.

[45]. Zhang and Suen (1984). “A fast parallel algorithm for thinning digital patterns [J],” *Communication of ACM*, 1984, 27(3): 236-239.

[46]. Guo and Richard (1989). “Parallel thinning with two-subiteration algorithms [J],” *Communication of ACM*, 1989, 32(3): 359-373.

[47]. Weian Deng, S. Sitharama Iyengar and Nathan E. Brener (2000), “A Fast Parallel Thinning Algorithm for the Binary Image Skeletonization,” *International Journal of High Performance Computing Applications* 2000 14: 65.

[48]. Wu and Tsai (1992). “A new one-pass parallel thinning algorithm for binary images,” *Pattern Recognition Letters* 13:715-23.

[49]. Li and Tian (2007). “Optimum cut-based clustering [J],” *Signal Processing*, 2007, 87(11): 2491-2502.

[50]. Wu, Z. Shi, and V. Govindaraju. “Fingerprint image enhancement method using directional median filter,” *Biometric Technology for Human Identification, SPIE*, volume 5404, pages 66–75, 2004.

[51]. Zhixin Shi, Venu Govindaraju, “A chaincode based scheme for fingerprint feature extraction”, *Pattern Recognition Letters*, vol. 27, 2006, pp. 462–468.

[52]. Zenzo, L. Cinque, and S. Levialdi, “Run-Based Algorithms for Binary Image Analysis and Processing”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 1, 1996, pp. 83-88.

[53]. J Hwan Shin, H. Y. Hwang, S Chien, “Detecting fingerprint minutiae by run length encoding scheme”, *Pattern Recognition*, vol. 39, 2005, pp. 1140-1154.

[54]. J. Xudong and Y. Wei-Yun, “Fingerprint minutiae matching based on the local and global structures”, in *Proc. of International Conference on Pattern Recognition (ICPR)*, vol. 2, 2000, pp. 1038–1041.

[55]. S. Prabhakar, A. K. Jain, and S. Pankanti, “Learning fingerprint minutiae location and type”, *Pattern Recognition*, vol. 36(8), 2003, pp. 1847–1857.

[56]. S. Chikkerur, V. Govindaraju, S. Pankanti, R. Bolle, and N. Ratha, “Novel approaches for minutiae verification in fingerprint images”, in *Seventh IEEE*

Workshops on Application of Computer Vision (WACV/MOTION'05), vol. 1, 2005, pp. 111–116.

[57]. Zhao Feng, Xiaou Tang, "Preprocessing and post processing for skeleton-based fingerprint minutiae extraction", *Pattern Recognition*, vol. 40, 2007, pp. 1270-1281.

[58]. R. Kaur, P. S. Sandhu and A. Kamra, "A Novel Method for Fingerprint Feature Extraction", In *Proc. International Conference on Networking and Information Technology*, 2010.

[59]. A.R. Patil, M. A. Zaveri, "A Novel Approach for Fingerprint Matching Using Minutiae," In *Proc. Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation (AMS)*, 2010, pp.317-322.

[60]. V. Humbe, S. S. Gornale, R. Manza and K. V. Kale, "Mathematical Morphology approach for Genuine Fingerprint Feature Extraction", *Int. Journal of Computer Science and Security (IJCSS)*, vol. 1, 2007, pp. 53-59.

[61]. R. Bansal, P. Sehgal, P. Bedi, "Effective Morphological Extraction of True Fingerprint Minutiae based on the Hit or Miss Transform", *International Journal of Biometrics and Bioinformatics(IJBB)*, vol. 4, 2010, pp. 71-85.

[62]. Maio and D. Maltoni, "Neural network based minutiae filtering in fingerprints", in *Fourteenth International Conference Pattern Recognition*, vol. 2, 1998, pp. 1654–1658.

[63]. X. Jiang, W.-Y. Yau, and W. Ser, "Detecting the fingerprint minutiae by adaptive tracing the grey-level ridge", *Pattern Recognition*, vol. 34(5), 2001, 999–1013.

[64]. L. Jinxiang, H. Zhongyang, and C. Kap Luk, "Direct minutiae extraction from grey-level fingerprint image by relationship examination", in *International Conference on Image Processing(ICIP)*, vol. 2, 2000, pp. 427–430.

[65]. K. Nilsson and J. Bign, "Using linear symmetry features as a pre-processing step for fingerprint images", in *AVBPA*, 2001, pp.247–252.

[66]. K. K. Hartwig Fronthaler and J. Bigun, “Local feature extraction in fingerprints by complex filtering,” In *Advances in Biometric Person Authentication*, *LNCS*, vol. 3781, 2005, pp.77–84.

[67]. R. M. Stock and C. W. Swonger, “Development and evaluation of a reader of fingerprint minutiae”, *Cornell Aeronautical Labaratory, Tecnical Report* CAL No. XM-2478-X-1:13-17, 1969.

[68]. M. R. Verma, A. K. Majumdar and B. Chatterjee, “Edge detection in fingerprints”, *Pattern Recognition*, v. 20, no. 5, pp. 513-523, 1987.

[69]. C. I. Watson and C. L. Wilson. Fingerprint database. *National Institute of Standards and Technology*, *Special Database 4*, April 18, 1992.

[70]. Angelo Chianese, Vincenzo Moscato, Antonio Penta, Antonio Picariello, “Improving minutiae detection in fingerprints using multiresolution contrast enhancement”, *ICIAR'06 Proceedings of the Third international conference on Image Analysis and Recognition - Volume Part II*, pp.283.

[71]. Q. Xiao and H. Raafat (1991), “Fingerprint image postprocessing: a combined statistical and structural approach”, *Pattern Recognition*, v. 24, no. 10, pp. 985-992.

[72]. A. M. Bazen and S. H. Gerez, “Fingerprint matching by thin-plate spline modeling of elastic deformations,” *Pattern Recognition*, vol. 36, no. 8, pp. 1859–1867, 2003.

[73]. Maltoni, D., Maio, D., Jain, A.K., Prabhakar, S., “Handbook of Fingerprint Recognition”. *Springer*, New York, pp.135-170, 2003.

[74]. K. Nandakumar , Jain, A.K., “Local Correlation-based Fingerprint Matching”, in *Proceedings of ICVGIP*, Kolkata, pp.2, December 2004.

[75]. A. M. Bazen, G. T. B. Verwaaijen, S. H. Gerez, L. P. J. Veelenturf, and B. J. van der Zwaag. “A Correlation-Based Fingerprint Verification System,” *Proceedings of Workshop on Circuits Systems and Signal Processing* (ProRISC 2000), pp. 205-213, 2000.

[76]. Crouzil, Massip-Pailhes and Castan (1996). Crouzil A., Massip-Pailhes L. and Castan S., “A New Correlation Criterion Based on Gradient Fields

Similarity," in *Proc. Int. Conf. on Pattern Recognition (13th)*, pp. 632–636, 1996.

[77]. Hatano et al. (2002). Hatano T., Adachi T., Shigematsu S., Morimura H., Onishi S., Okazaki Y. and Kyuragi H., "A Fingerprint Verification Algorithm Using the Differential Matching Rate," in *Proc. Int. Conf. on Pattern Recognition (16th)*, vol. 3, pp. 799–802, 2002.

[78]. Ito et al. (2005). Ito K., Morita A., Aoki T., Higuchi T., Nakajima H. and Kobayashi K., "A Fingerprint Recognition Algorithm Using Phase-Based Image Matching for Low-Quality Fingerprints," in *Proc. Int. Conf. on Image Processing*, vol. 2, pp. 33–36, 2005.

[79]. Shuai, Zhang and Hao (2007). Shuai X., Zhang C. and Hao P., "The Optimal ROS-Based Symmetric Phase-Only Filter for Fingerprint Verification," in *Proc. Int. Conf. on Image Processing*, vol. 2, pp.381–384, 2007.

[80]. Ballard (1981). Ballard D.H., "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 3, no. 2, pp. 110–122, 1981.

[81]. Stockman, Kopstein and Benett (1982). Stockman G., Kopstein S. and Benett S., "Matching images to models for registration of and object detection via clustering," IEEE Transactions on Pattern Analysis Machine Intelligence, vol. 4, no. 3, pp. 229–241, 1982.

[82]. Ratha et al. (1996). Ratha N.K., Karu K., Chen S. and Jain A.K., "A real-time matching system for large fingerprint databases," IEEE Transactions on Pattern Analysis Machine Intelligence, vol. 18, no. 8, pp. 799–813, 1996.

[83]. Chang et al. (1997). 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, pp.311–320, 1997.

[84]. Jain, Hong, and Bolle (1997). Jain A.K., Hong L., and Bolle R., "On-line Fingerprint Verification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 302−313, 1997.

[85]. Jea and Govindaraju (2005). Jea T.Y. and Govindaraju V., “A minutia-based partial fingerprint recognition system,” *Pattern Recognition*, vol. 38, no. 10, pp. 1672–1684, 2005.

[86]. Chen, Tian and Yang (2006). Chen X., Tian J. and Yang X., “A new algorithm for distorted fingerprints matching based on normalized fuzzy similarity measure,” *IEEE Transactions on Image Processing*, vol. 15, no. 3, pp. 767–776, 2006.

[87]. Germain, Califano and Colville (1997). Germain R., Califano A. and Colville S., “Fingerprint matching using transformation parameters,” *IEEE Computational Science and Engineering*, vol. 4, no. 4, pp. 42–49, 1997.

[88]. Tan and Bhanu (2003). Tan X. and Bhanu B., “A robust two step approach for fingerprint identification,” *Pattern Recognition Letters*, vol. 24, no. 13, pp. 2127–2134, 2003.

[89]. Chen et al. (2006). Chen X., Tian J., Yang X. and Zhang Y., “An algorithm for distorted fingerprint matching based on local triangle feature set,” *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 169–177, 2006.

[90]. Tico and Kuosmanen (2003). Tico M. and Kuosmanen P., “Fingerprint matching using an orientationbased minutia descriptor,” *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 25, no. 8, pp. 1009–1014, 2003.

[91]. A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti., “Filterbank-Based Fingerprint Matching,” *IEEE Transactions on Image Processing*, 9:846–859, 2000.

[92]. Xie, Su and Cai (2006). Xie X., Su F. and Cai A., “Ridge-Based Fingerprint Recognition,” in Proc. Int. Conf. on Biometrics, LNCS 3832, pp. 273–279, 2006.

[93]. Ratha et al. (2000). Ratha N.K., Pandit V.D., Bolle R.M. and Vaish, V., “Robust Fingerprint Authentication Using Local Structural Similarity,” in *Proc. Workshop on Applications of Computer Vision*, pp. 29–34, 2000.

[94]. Raffaele Cappelli, Dario Maio, Davide Maltoni, James L. Wayman, and Anil K. Jain, “Performance Evaluation of Fingerprint Verification Systems,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.28, No.1, pp. 12, January 2006.

[95]. ATMEL Corporation, “AT77C102B FingerChip”.

[96]. Han and Koshimoto (2008). Han H. and Koshimoto Y., “Characteristics of Thermal-Type Fingerprint Sensor,” in *Proc. SPIE Conf. on Biometric Technology for Human Identification V*, 2008.

[97]. ATMEL Corporation, “ARM926EJ-S Technical Reference Manual”.