Fingerprint Image Analysis Using Computational Geometric Techniques

by

Xuefeng LIANG

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Tetsuo Asano

School of Information Science
Japan Advanced Institute of Science and Technology

March, 2006

Abstract

Fingerprint recognition is a complex pattern recognition problem; designing algorithms capable of extracting salient features and matching in a robust way is quite hard, especially for poor quality fingerprint images. There is a popular misconception that automatic fingerprint recognition is a fully solved problem since it was one of the first applications of machine pattern recognition almost fifty years ago. On the contrary, fingerprint recognition is still a challenging and important pattern recognition problem.

Automatic Fingerprint Identification Systems (AFIS) provide widely used biometric techniques for personal identification (e.g. authentication, forensic decision, etc.). Fingerprints are useful for biometric purposes because of their well known properties of distinctiveness and persistence over time. Existing AFIS face two critical problems. First is the preprocessing phase is known to consume almost $90 \sim 95\%$ of the total time of fingerprint identification. That is the reason why a considerable amount of research has been focused on this area. Second is the fingerprint distortion which changes geometric relationship among minutiae. This change makes minutiae matching quite difficult, meanwhile, decreases accurate of AFIS so as not to satisfy some strict applications (e.g. Bank Security System, etc.).

In this thesis, we have addressed the above two issues, and developed several novel algorithms for them using computational geometric techniques.

A crucial idea of the research behind reducing preprocessing time is a linear time *Euclidean distance transform* (EDT). The same feature of Euclidean distance transform can be used for binarization, denoising, minutiae extraction and matching, almost through whole AFIS.

A matrix of Euclidean distance transform values is generated in binarization step. Through inheriting and using this same EDT matrix, denoising and minutiae extraction steps can efficiently obtain our expected results by several novel approaches. And this strategy in real application can save a lot of time. Experiments show our method decreases $20 \sim 30\%$ of computing time than other fast methods in the preprocessing stage.

To cope with fingerprint distortion, we in-depth investigated property of finger tips, and then propose an combined RBF distortion model to correct a non-rigid transformation of distortion according to several control points. Our method is dependent only on a few parameters determined by experiences and avoids definition of the size and shape of conventional tolerance box. These benefits guarantee our method more automatic and robust. Matching results prove combined RBF model has a high capability, which improves the accuracy about 70% than a rigid transformation, to cope with fingerprint distortion, even can catch up with some manual distortion model.

Acknowledgments

Completing a PhD is truly a tough event, and I would not have been able to complete this task without the aid and support of countless people over the past three years. I must first express my sincere gratitude towards my principal advisor Professor Tetsuo Asano for his constant encouragement and kind guidance during this work. I would like to thank Associate Arijit Bishnu for his helpful discussions and constructive suggestions. Their insights and comments were invaluable over the years, and I look forward to a continuing collaboration with them in the future.

I also wishes to express many thanks to my minor-research advisor Associate Professor Kazunori Kotani for his suggestions and continuous encouragements.

Over the years, I have enjoyed the aid of support and scholarship which have supported me while I completed my PhD. During 2003–2006 I was supported by the "Fostering Talent in Emergent Research Fields" in Special Coordination Funds for Promoting Science and Technology by Ministry of Education, Culture, Sports, Science and Technology; in 2004, 2005 I received the Special Scholarship twice from JAIST; and for these three years, my work has been partly supported by FUJITSU LABORATORIES LTD.

I am grateful to all who have affected or suggested my areas of research. Associate Professor Ryuhei Uehara and Associate Mitsuo Motoki gave me kind encouragements. I also thank some of my fellow PhD students and friends: Shinji Sasahara, Sachio Teramoto, Hui Zhang, Zonghua Zhang, Shuang Wang, Xiangrong Zhang. They each helped make my time in the PhD program more fun and interesting.

Finally, I deeply thank my parents and sisters for instilling in my confidence and a drive for pursuing my PhD. Especially my wife, she took most house work to let me concentrate on my research, and never lost faith in this long-term program.

Contents

Chapter 1

Introduction

Biometric recognition refers to the use of distinctive physiological and behavioral characteristics of human being (e.g. fingerprint, face, palm, iris, gait, signature) for automatically recognizing a people. Fingerprints have the properties of distinctiveness or individuality, and the fingerprints of a particular person remain almost the same (persistence) over time. These properties make fingerprints suitable for biometric uses. *Automatic fingerprint identification systems* (AFIS) provide widely used biometric techniques for personal identification. AFIS are usually based on minutiae matching. Minutiae, or Galton's characteristics [34] are local discontinuities in terms of terminations and bifurcations of the ridge flow patterns that constitute a fingerprint. These two types of minutiae have been considered by Federal Bureau of Investigation for identification purposes. AFIS based on minutiae matching involves different stages (see Figure 1.1 for an illustration):

1. **Fingerprint image acquisition** [18, 81, 50]**;**
2. **Preprocessing of the fingerprint image (binarization, denoising and ridge extraction);**
3. **Feature extraction (e.g. minutiae) from the image;**
4. **Matching of fingerprint images for identification.**

More than a century ago, the Home Ministry Office, UK, already used the fingerprints to determine the identity of criminals. From then on, hundreds of scientific methods were developed for visual matching of fingerprint and applied the art of fingerprint recognition for nailing down the perpetrators [76, 56]. Therefore, there is a popular misconception that automatic fingerprint recognition is a fully solved problem since it was one of the first applications of machine pattern recognition almost fifty years ago. On the contrary, fingerprint recognition is still a challenging and important pattern recognition problem. There are several intractable problems besetting fingerprint recognition. For instances, fingerprint recognition is a kind of online application, hence AFIS are demanded to give out a correct answer as fast as possible. Whereas most proposed methods only concentrate on a certain step of AFIS, therefore, a whole AFIS have to apply various techniques in each step. No more helpful information, which can accelerate system, is inherited from previous step except the step results. It causes identification procedure, especially the preprocessing, takes a long time. The preprocessing phase is known to consume almost

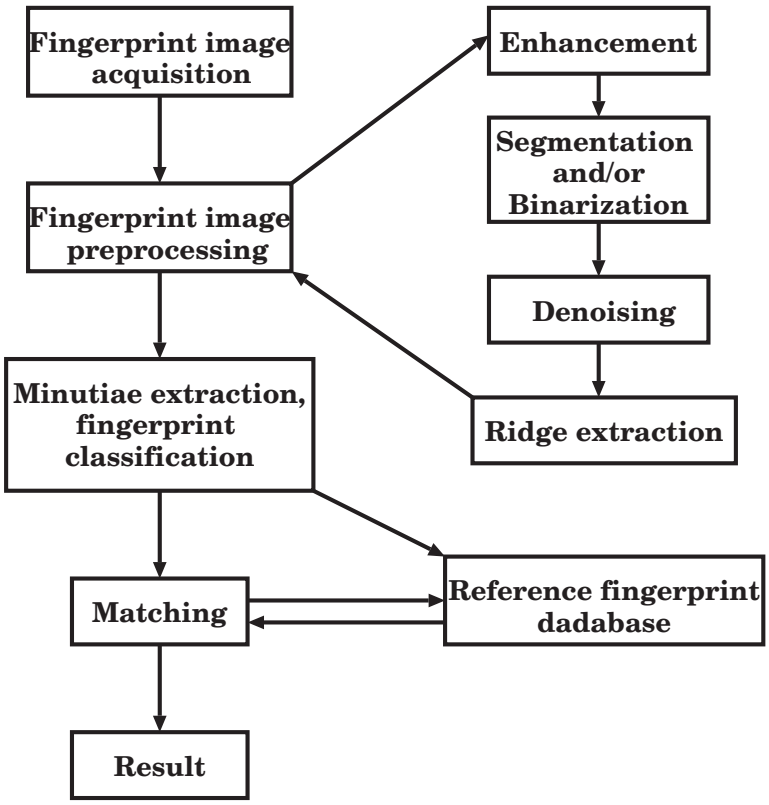


Figure 1.1: A flowchart showing different phases of fingerprint analysis.

$90 \sim 95\%$ of the total time of fingerprint identification [9]. That is the reason why a considerable amount of research has been focused on this area. Furthermore, fingerprints may be physiological in nature but the usage of the input device (e.g., how a user presents a finger to the fingerprint scanner) depends on human behavior. Thus, the scanned prints are a combination of physiological and behavioral characteristics. This causes the phenomena that in most cases the location of the same feature points (minutiae) will not be exact in the same in different versions of the same fingerprint, namely *distortion*. Distortion leads to a great difficulties in establishing a match among multiple fingerprints acquired from a single finger.

Other than a theoretical problem, fingerprint identification is an application research. With increasing demand of higher accuracy of AFIS, more automatic and robust systems are required. Most conventional techniques, however, were developed are heuristic with a lot of parameters for particular situations. These make systems complicated and time consuming. Digital fingerprint image analysis is basically a geometric problem. Techniques borrowed from computational geometry can provide good solutions to these problems. This thesis reviews the major scientific techniques made over the past two decades for above two intractable problems in previous paragraph, and proposes our novel algorithms with a few parameters to cope with them using computational geometric techniques.

After an introductory chapter, the thesis chapters are organized logically into nine chapters: Preliminaries of fingerprint recognition (Chapter 2). A linear time Euclidean distance transform algorithm (Chapter 3). Chapter 4, 5 and 6 describe three major steps of preprocessing. All of them use Euclidean distance transform as common basic

technique to reduce preprocessing time. In detail, a novel binarization algorithm (Chapter 4); eliminating impulsive noise and useless components (Chapter 5); minutiae extraction (Chapter 6). Distortion correction (Chapter 7) and fingerprint matching (Chapter 8). Conclusions and future works (Chapter 9).

Chapter 2 roughly reviewed constituents of fingerprint recognition system, elementary conceptions and definitions of fingerprint representation and fingerprint matching. This will provide the reviewers some basic and primary understanding of our work in this thesis.

Within image analysis the distance transform has many applications. The distance transform measures the distance of each object point from the nearest boundary. Chapter 3 introduces a linear time Euclidean distance transform algorithm used in our AFIS to reduce preprocessing time. This efficient algorithm works by performing a 1D distance transform on each row of the image, and then combines the results in each column. The resulting values are used to draw an envelope of parabolas, which can occur in the lower envelope at most once, to compute the Euclidean distance transform in linear time.

Chapter 4 introduces our novel fingerprint binarization algorithm. Fingerprint images are characterized by alternating spatial distribution of gray-level intensity values of ridges and ravines/valleys of almost equal width. Most of fingerprint matching techniques require extraction of minutiae that are the terminations and bifurcations of the ridge lines in a fingerprint image. Crucial to this step is either detecting ridges from the gray-level image or binarizing the image and then extracting the minutiae. In this work, we exploit the property of almost equal widths of ridges and valleys for binarization. Computing the width of arbitrary shapes is a non-trivial task. So, we estimate width using Euclidean distance transform and provide a near-linear time algorithm for binarization. Unlike many other previous methods, our work depends minimally on any arbitrary selection of parameters. Another significant advantage of our algorithm is that binarization and fingerprint area segmentation can be done simultaneously.

Chapter 5 discusses two kinds of noise of fingerprint images and introduce how we solve this problems. Owing to skin conditions or incorrect finger pressure, original fingerprint images always contain noise. Especially, some of them contain useless components, which are often mistaken for the terminations that are an essential minutia of a fingerprint. Mathematical Morphology (MM) is a powerful tool in image processing. In this chapter, we propose a linear time algorithm to eliminate impulsive noise and useless components, which employs generalized and ordinary morphological operators based on Euclidean distance transform. There are two contributions. The first is the simple and efficient MM method to eliminate impulsive noise, which can be restricted to a minimum number of pixels. We know the performance of MM is heavily dependent on structuring elements (SEs), but finding an optimal SE is a difficult and nontrivial task. So the second contribution is providing an automatic approach without any experiential parameter for choosing appropriate SEs to eliminate useless components. The information of distance transform values can be obtained directly from the binarization phase. The results show that using this method on fingerprint images with impulsive noise and useless components is faster than existing denoising methods and achieves better quality than earlier methods.

Chapter 6 describes a new way of minutiae detection using Euclidean distance transform values. Most of the minutiae detection algorithms find out the minutiae from the

one-pixel thick ridge image obtained from thinning after binarization. Once our binarization and denoising processes finish, we have a matrix with each ridge pixel having a corresponding EDT value. Our goal is now to exploit this information to detect minutiae straightaway from the thick ridges of the binary image without thinning. This will be useful in saving time in real life application. The location of the same minutiae in most of the cases will not be exact in different versions of the same fingerprint owing to distortion and as such it makes sense to find out a small region in which the minutiae lies; in other words, this region is a set of connected pixels. So, our purpose is to find these minutiae regions for further matching. The minutiae treated will be bifurcation and termination.

Chapter 7 explains a distortion model for correcting deformation of fingerprint images. The techniques of matching based on minutiae still suffers from problems associated with the handling of poor quality prints. One problem besetting fingerprint matching is distortion. Distortion changes both geometric position and orientation, and leads to difficulties in establishing a match among multiple impressions acquired from the same finger tip. In this paper, we first determine an appropriate basis function and locality parameters for a normal RBF model through theoretical analysis of RBF and mass experiments. This normal RBF model is designed to deal with distortion for each possible matched-pair of fingerprints. According to the particularity of fingerprint distortion, we further propose an improved combined RBF model, which separately builds rigid and nonrigid transformations, for attacking the distortion problem. Combined RBF model provides more accurate mapping function between a possible matched-pair. Experiments on real data demonstrate the efficacy of our method in improving the correction of fingerprint distortion.

Chapter 8 finishes our AFIS with a deterministic fingerprint matching algorithm based on a novel minutia polygon. Conventional fingerprint minutiae matching algorithms use tolerance box (e.g. circle, square or rectangle). Although, these methods employ some geometry information of minutiae (e.g. type, coordinate and orientation). However, other particular information of minutiae (especially for bifurcation) were lost, such as the shape of bifurcation. Moreover, these tolerance boxes usually are determined experientially, hence matching result heavily depends on the pre-defined size and shape of tolerance box. Aim at this issue, we design a simple polygon that include all previous information for bifurcation (and termination) minutia, and a corresponding efficient matching algorithm. One significant advantage of minutia polygon is that it is adaptive and does not depend on the size in an appropriate range. In other words, minutia polygon can be enlarged or shrunk by a reasonable scale for matching without losing accuracy.

Finally, in Chapter 9, we summarize the work reported in this thesis, and point out a few research problems would be solved in the future.

Chapter 2

Preliminaries

More than a century has passed since Alphonse Bertillon first conceived and then industriously practiced the idea of using body measurements for solving crimes [70]. Just as his idea was gaining popularity, it faded into relative obscurity by a far more significant and practical discovery of the distinctiveness of the human fingerprints. In 1893, the Home Ministry Office, UK, accepted that no two individuals have the same fingerprints. Soon after this discovery, many major law enforcement departments embraced the idea of first "booking" the fingerprints for determining the identity of criminals, so that their records are readily available and later using leftover fingerprints smudges, they could determine the identity of criminals. These agencies sponsored a rigorous study of fingerprint, developed scientific methods and training experts for visual matching of fingerprints.

Despite the ingenious methods improvised to increase the efficiency of the manual approach to fingerprint indexing and search, the ever growing demands on manual fingerprint recognition quickly became overwhelming. The manual method of fingerprint indexing resulted in a highly skewed distribution of fingerprints into bins (types): most fingerprints fell into a few bins and this did not improve search efficiency. Manual visual matchings were time-consuming and increasing workload due to a higher demand on fingerprint recognition service, all prompted the law enforcement agencies to initiate research into acquiring fingerprints through electronic media and automate fingerprint recognition based on the digital representation of fingerprint. These efforts led to development of *Automatic Fingerprint Identification Systems* (AFIS) [91, 49] over the past few decades.

Henry Fauld first scientifically suggested the individuality of fingerprints based on an empirical observation. At the same time, Herschel asserted that he had practiced fingerprint recognition for about 20 years [65]. These findings established the foundation of modern fingerprint recognition. In the late nineteenth century, Francis Galton conducted an extensive study on fingerprints [34]. Till now, hundreds of scientific methods were developed for visual matching of fingerprint and applied the art of fingerprint recognition for nailing down the perpetrators [76, 56]. Their efforts were so successful that today almost every law enforcement agency worldwide uses an AFIS. These systems have greatly improved the operational productivity of law enforcement agencies. However, this makes a popular misconception in the pattern recognition and image processing academic community that automatic fingerprint recognition is a fully solved problem since it was one of the first applications of machine pattern recognition almost fifty years ago. On the contrary, fingerprint recognition is still a challenging and important pattern recognition

problem.

2.1 Fingerprint Recognition System

A fingerprint recognition system is essentially a pattern recognition system that recognizes a person by determining the authenticity of a fingerprint characteristic possessed by that person (see Figure 2.1). An important issue in designing a practical fingerprint recognition system is to determine how an individual is recognized. Depending on the application context, a fingerprint recognition system may be called either a *verification* system or an *identification* system (see Figure 2.2 and 2.3):

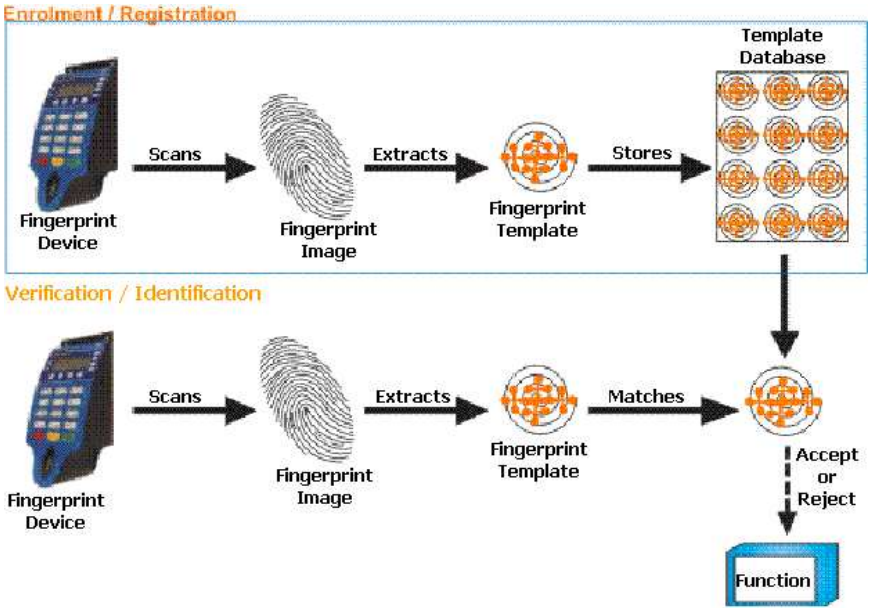


Figure 2.1: A fingerprint recognition system.

- A **verification system** authenticates a person's identity by comparing the captured fingerprint characteristic with his/her own fingerprint template pre-stored in the system [47]. It conducts one-to-one comparison to determine whether the identity claimed by the individual is true. A verification system either rejects or accepts the submitted claim of identity, in other words, it answers the question: *Am I whom I claim I am* ?
- An **identification system** recognizes an individual by searching the entire template database for a match. It conducts one-to-many comparison to establish the identity of the the individual [48]. In an identification system, the system establishes a subject's identity (or fails if the subject is not enrolled in the system database) without the subject having to claim an identity. It answers the question: *Who am I* ?

Depending on the application domain, a fingerprint recognition system could operate either as an *on-line* system or an *off-line* system. An on-line system requires the recognition to be performed quickly and immediate response is imposed (e.g., an ATM of bank

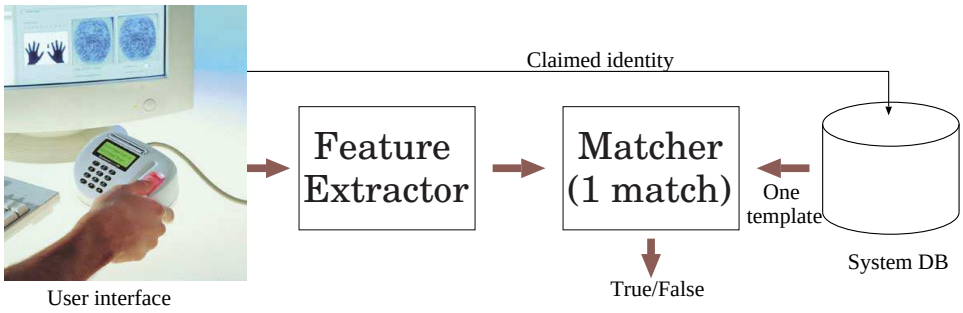


Figure 2.2: A fingerprint verification system.

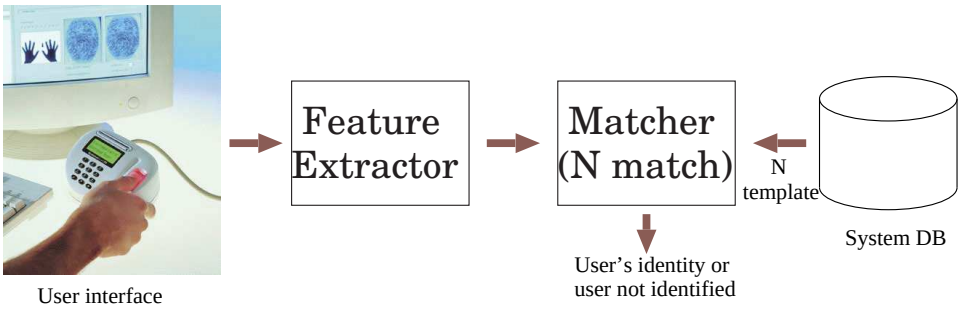


Figure 2.3: A fingerprint identification system.

login application). On the other hand, an off-line system usually delay is allowed (e.g., an person background check application). Typically, on-line systems are fully automatic and require that the fingerprint characteristic be captured using a live-scan sensor, the enrollment process be unattended, there be no (manual) quality control, and the matching and decision be fully automatic. Off-line systems, however, are typically semi-automatic, where the fingerprint acquisition could be through an off-line scanner (e.g., scanning a fingerprint image from a latent or inked fingerprint card), the enrollment may be supervised, a manual quality check may be performed to ensure good quality acquisition, and the matcher may return a list of candidates which are then manually examined by a forensic expert to arrive at a final (human) decision.

An application could operate either in a *positive* or *negative* recognition mode:

- In a **positive** recognition application, the system establishes whether the person is who he (implicitly or explicitly) claims to be. The purpose of a positive recognition is to prevent multiple people from using the same identity. For example, if only Liang is authorized to enter a certain secure area, then the system will grant access only to Liang. If the system fails to match the enrolled template of Liang with the input, a rejection results; otherwise, an acceptance results;

- In a **negative** recognition application, the system establishes whether the person is who he (implicitly or explicitly) denies being. The purpose of negative recognition is to prevent a single person from using multiple identities. For example, if Liang has already received welfare benefits and now he claims that he is Zhang and would like to receive the welfare benefits of Zhang (this is called "double dipping"), the system will establish that Zhang is not who she claims to be. If the system fails

to match the input biometric of Zhang with a database of people who have already received benefits, an acceptance results; otherwise, a rejection results.

Note that positive recognition application can operate both in verification or identification mode, but negative recognition applications cannot work in verification mode: in fact, the system has to search the entire archive to prove that the given input is not already present.

In this thesis, all of techniques to be discussed are applied on a on-line fingerprint identification system.

2.2 Fingerprint Representation

A good fingerprint representation should have the following two properties: *saliency* and *suitability*. Saliency means that a representation should contain distinctive information about the fingerprint. Suitability means that the representation can be easily extracted, stored in a compact fashion, and be useful for matching. Saliency and suitability properties are not generally correlated. A salient representation is not necessarily a suitable representation.

Image-based representations, constituted by raw pixel intensity information, are prevalent among the recognition systems using optical matching and correlation-based matching. However, the utility of the systems using such representation schemes may be limited due to several factors such as brightness variations, image quality variations, scars, and large global distortions present in the fingerprint image. Furthermore, an image-based representation requires a considerable amount of storage. On the other hand, an image-based representation preserves the maximum amount of information, makes fewer assumptions about the application domain, and therefore has the potential to be robust to wider varieties of fingerprint images. For instance, it is extremely difficult to extract robust features from a (degenerate) finger devoid of any ridge structure.

The fingerprint pattern, when analyzed at different scales, exhibits different types of features.

- At the global level, the ridge line flow delineates a pattern similar to one of those shown in Figure 2.4. *Singular points*, called loop and delta (denoted as squares and triangles, respectively in Figure 2.4), are a sort of control points around which the ridge lines are "wrapped"[57]. Singular points and coarse ridge line shape are very important for fingerprint classification and indexing, but their distinctiveness is not sufficient for accurate matching. External fingerprint shape, orientation image, and frequency image also belong to the set of features that can be detected at the global level.

- At the local level, a total of 150 different local ridge characteristics, called *minute details*, have been identified [65]. These local ridge characteristics are not evenly distributed. Most of them depend heavily on the impression conditions and quality of fingerprints and are rarely observed in fingerprints. The two most prominent ridge characteristics, called *minutiae* (see Figure 2.5), are: *ridge ending (termination)* and

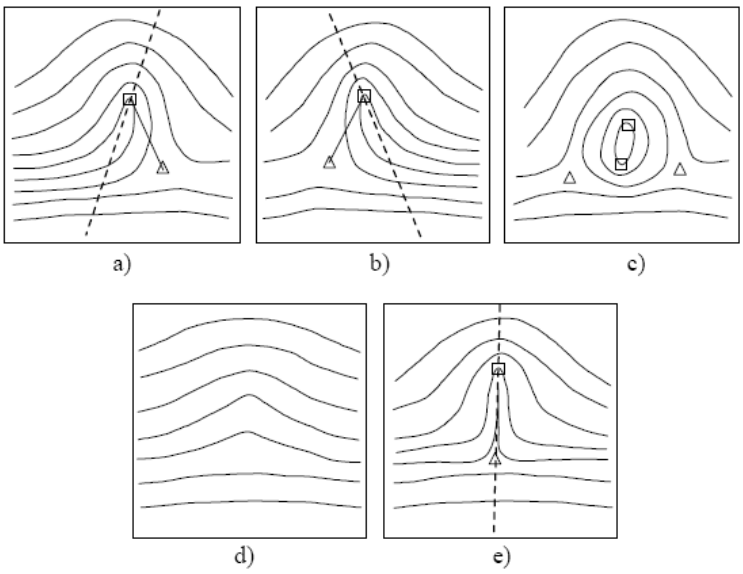


Figure 2.4: Fingerprint patterns as they appear at a coarse level: a) left loop; b) right loop; c) whorl; d) arch; and e) tented arch; squares denote loop-type singular points, and triangles delta-type singular points.

ridge bifurcation. A ridge ending is defined as the ridge point where a ridge ends abruptly. A ridge bifurcation is defined as the ridge point where a ridge forks or diverges into branch ridges. Minutiae in fingerprints are generally stable and robust to fingerprint impression conditions. Although a minutiae-based representation is characterized by a high saliency, a reliable automatic minutiae extraction cannot be problematic in normal-quality fingerprints (hence the suitability of this kind of representation is adequate).

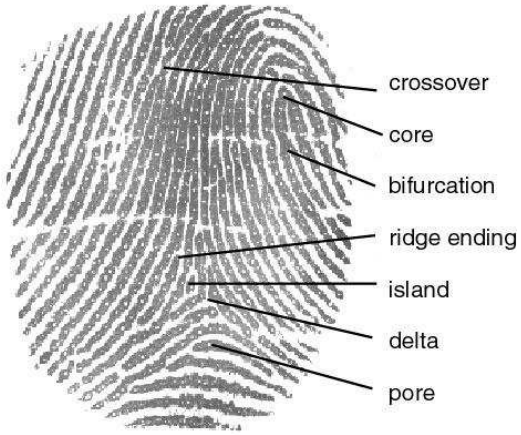


Figure 2.5: Fingerprint minutiae types.

2.3 Fingerprint Matching

Reliably matching fingerprint images is an extremely difficult problem, mainly due to the large variability in different impressions of the same finger (i.e., large intra-class variations). The main factors responsible for the intra-class variations are: displacement, rotation, partial overlap, non-linear distortion, variable pressure, changing skin condition, noise, and feature extraction errors. Therefore, fingerprints from the same finger may sometimes look quite different whereas fingerprints from different fingers may appear quite similar (see Figure 2.6).

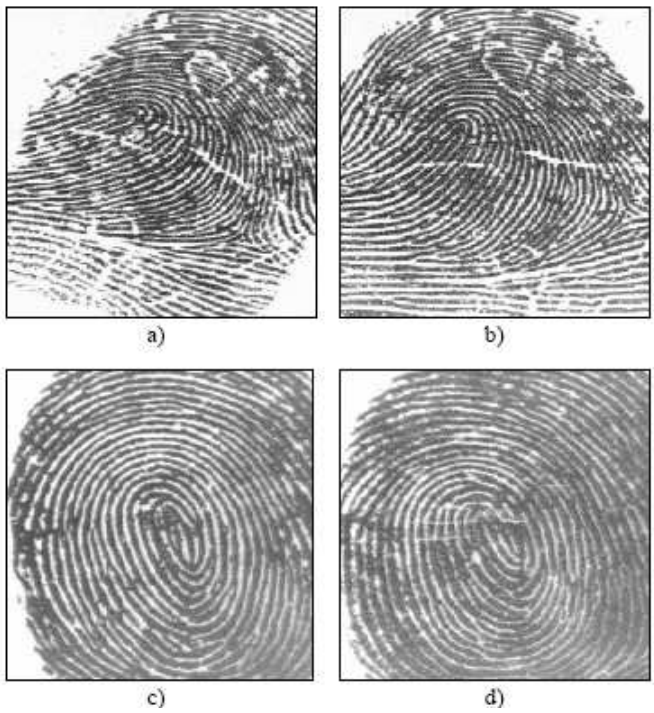


Figure 2.6: Difficulty in fingerprint matching. Fingerprint images in a) and b) look different to an untrained eye but they are impressions of the same finger. Fingerprint images in c) and d) look similar to an untrained eye but they are from different fingers.

Human fingerprint examiners, in order to claim that two fingerprints are from the same finger, evaluate several factors: i) global pattern configuration agreement, which means that two fingerprints must be of the same type, ii) qualitative concordance, which requires that the corresponding minute details must be identical, iii) quantitative factor, which specifies that at least a certain number (a minimum of 12 according to the forensic guidelines in the United States) of corresponding minute details must be found, and iv) corresponding minute details, which must be identically inter-related.

Most approaches of automatic fingerprint matching is *minutiae-based matching*: minutiae are extracted from the two fingerprints and stored as sets of points in the two-dimensional plane. Minutiae matching essentially consists of finding the alignment between the template and the input minutiae sets that results in the maximum number of minutiae pairings.

The response of a matcher in a fingerprint recognition system is typically a matching

score s (without loss of generality, ranging in the interval [0,1]) that quantifies the similarity between the input and the database template representations. The closer the score is to 1, the more certain is the system that the two fingerprints come from the same finger; the closer the score is to 0, the smaller is the system confidence that the two fingerprints do not come from the same finger. The system decision is regulated by a *threshold* t: pairs of fingerprints generating scores higher than or equal to t are inferred as *matching pairs* (i.e., belonging to the same finger); pairs of fingerprints generating scores lower than t are inferred as *non-matching pairs* (i.e., belonging to different fingers).

A typical fingerprint verification system commits two types of errors: mistaking measurements from two different fingers to be from the same finger (called *false match*) and mistaking two measurements from the same finger to be from two different fingers (called *false non-match*). Note that these two types of errors are also often denoted as *false acceptance* and *false rejection*; a distinction has to be made between positive and negative recognition; in positive recognition systems (e.g., an access control system) a false match determines the false acceptance of an impostor, whereas a false non-match causes the false rejection of a genuine user. On the other hand, in a negative recognition application (e.g., preventing users from obtaining welfare benefits under false identities), a false match results in rejecting a genuine request, whereas a false non-match results in falsely accepting an impostor attempt. The notation "false match/false non-match" is not application dependent and therefore, in principle, is preferable to "false acceptance/false rejection". However, the use of *false acceptance rate* (FAR) and *false rejection rate* (FRR) is more popular and largely used in the commercial environment.

Chapter 3

A Linear Time Euclidean Distance Transform

3.1 Introduction

Many image analysis applications require the measurement of objects, the components of objects or the relationship between objects. One technique that may be used in a wide variety of applications is the distance transform or Euclidean distance map [71, 74]. Let pixels $I(x, y)$ within a two-dimensional digital image be divided into two classes: object pixels and background pixels.

$$I(x, y) \in \{Ob, Bg\} \tag{3.1}$$

The distance transform of this image, $I_d(x, y)$ then labels each object pixel of this binary image with the distance between that pixel and the nearest background pixel. Mathematically,

$$I_d(x, y) = \begin{cases} 0 & I(x, y) \in Bg \\ min(\|x - x_0, y - y_0\|, \forall I(x_0, y_0) \in Bg) & I(x, y) \in Ob \end{cases} \tag{3.2}$$

where $\|x, y\|$ is two dimensional distance metric. Different distance metrics result in different distance transformations. From a measurement perspective, the Euclidean distance is the most useful because it corresponds to the way objects are measured in the real world. The Euclidean distance metric uses the L_2 norm and is defined as:

$$\|x, y\|_{L_2} = \sqrt{x^2 + y^2} \tag{3.3}$$

This metric is isotropic in that distances measured are independent of object orientation, subject of course to the limitation that the object boundary is digital, and therefore in discrete locations. The major limitation of the Euclidean metric, however, is that it is not easy to calculate efficiently for complex shapes. Therefore several approximations have been developed that are simpler to calculate for two-dimensional digital images using a rectangular coordinate system. The first of these is *the city block*, or *Manhattan* metric, which uses the L_1 norm:

$$\|x, y\|_{L_1} = |x| + |y| \tag{3.4}$$

where the distance is measured by the number of horizontal and vertical steps required to traverse (x, y). If each pixel is considered a node on a graph with each node connected to its 4 nearest neighbors, the city block metric therefore measures the distance as the minimum number of 4-connected nodes that must be passed through. Diagonal distances are over-estimated by this metric because a diagonal connection counts as 2 steps, rather than $\sqrt{2}$.

Another measure commonly used is *the chessboard* metric, using the L_∞ norm:

$$\|x, y\|_{L_\infty} = max(|x|, |y|) \tag{3.5}$$

which measures the number of steps required by a king on a chess board to traverse (x, y). The chessboard metric considers each pixel to be connected to its 8 nearest neighbors, and measures the distance as the minimum number of 8-connected nodes that must be passed through. Diagonal distances are under-estimated by this metric as a diagonal connection counts as only 1 step.

A wide range of other metrics have been proposed that aim to approximate the Euclidean distance while retaining the simplicity of calculation of the city block and chessboard metrics.

$$\|x, y\|_{quasi-Euclidean} = \begin{cases} |x| + (\sqrt{2} - 1)|y| & |x| > |y| \\ (\sqrt{2} - 1)|x| + |y| & otherwise \end{cases} \tag{3.6}$$

Perhaps the simplest of these is to simply average the city block and chessboard distance maps:

$$\|x, y\|_{Hybrid} = \frac{1}{2}(|x| + |y| + max(|x|, |y|)) \tag{3.7}$$

Figure 3.1 graphically compares these different metrics in measuring the distance from a point in the center of an image. The anisotropy of the non-Euclidean distance measures is clearly visible.

There exists several approaches to calculate the distance transform. the simplest approach is the so-called grassfire transform using morphology operators [41, 89, 19]. The two pass chamfer distance algorithm may be adapted to measure the Euclidean distance by propagating vectors instead of the scalar distance [23, 68, 80]. Another class of techniques combines the idea of the grassfire transform with the propagation approach described in the previous section. These methods maintain a list of boundary pixels, and propagate these in a non-raster fashion [21, 86, 27].

3.2 Linear Time Independent Scanning

From Pythagoras' theorem, the distance squared to a background pixel can be determined by considering the x and y components separately. Therefore it is possible to independently consider the rows and columns. The first step looks along each row to determine

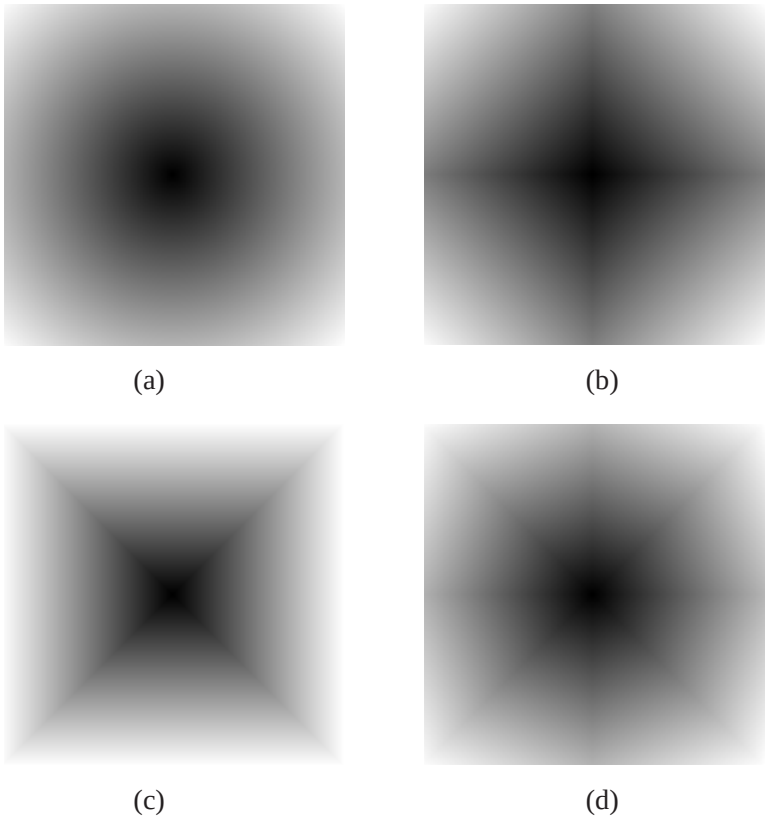


Figure 3.1: Four commonly used distance metrics C measuring the distance from the center of the image: (a) Euclidean metric, Equation 3.3; (b) city block metric, Equation 3.4; (c) chessboard metric, Equation 3.5; (d) a quasi-Euclidean metric, Equation 3.6

the distance of each object point from the nearest boundary point on that row. This requires two scans, from left to right and right to left to measure the distances from the left and right edges of the object respectively. The second step then considers each column, and for each pixel in that column determines the closest background point by examining only the row distances in that column:

$$I_d^2(x,y) = \min_n(I(x,y_n)^2 + (y-y_n)^2) \tag{3.8}$$

Thus the search has been reduced from two dimensions in equation 3.2 to one dimension. The search can be accomplished with a scan down and up the column propagating the row distances and selecting the global minima at each pixel [75]. Unfortunately, as applied, this algorithm requires that multiple row points be propagated simultaneously. The effect is that in the worst case the algorithm as described is not linear in the number of pixels.

The key to making an independent scanning algorithm operate in linear time is to determine in advance exactly which pixels in a column that a particular row will influence. This information may be obtained by constructing a partial Voronoi diagram for each column.

3.2.1 Row Scanning

The first step operates on each row independently. It consists of two passes (see Figure 3.2 for illustration): from left to right and then right to left. The left to right pass determines the distance to the left boundary of an object:

$$I(x,y) = \begin{cases} I(x-1,y)+1 & I(x,y) \in Ob \\ 0 & I(x,y) \in Bg \end{cases} \tag{3.9}$$

If the pixel on the edge of the image is an object pixel, its distance is set to ∞. The right to left pass replaces this with the distance to the right boundary if it is shorter:

$$I(x,y) = \begin{cases} min(I(x,y), I(x+1,y)+1) & I(x,y) \in Ob \\ 0 & I(x,y) \in Bg \end{cases} \tag{3.10}$$

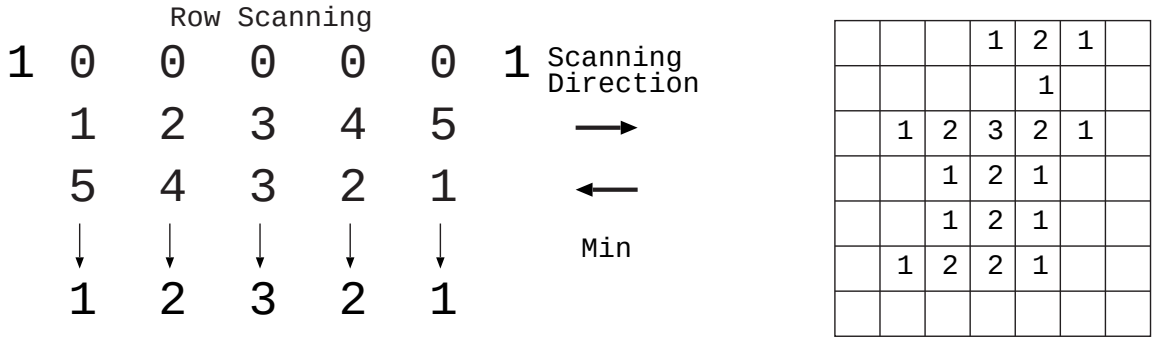


Figure 3.2: Compute horizontal distance by bidirectional scanning and take minimum of each such distance.

3.2.2 Column Scanning

Consider an image with two background pixels at $I(x_1, y_1)$ and $I(x_2, y_2)$, with $y_1 < y_2$. Let I_1 and I_2 be the corresponding minimum row distances in column x. The distance squared function in column x is illustrated in Figure 3.3. The column is split into two with part of the column coming under the influence of (x_1, y_1) and part coming under the influence of (x_2, y_2). The boundary between the two regions is given from the intersection of the two parabola:

$$I_1^2 + (y' - y_1)^2 = I_2^2 + (y' + y_2)^2 \tag{3.11}$$

Solving this for the position of the intersection gives:

$$y' = y_2 + \frac{I_2^2 - I_1^2 - (y_2 - y_1)^2}{2(y_2 - y_1)} \tag{3.12}$$

Note that there will always be exactly one intersection point, corresponding to where the perpendicular bisector between $I(x_1, y_1)$ and $I(x_2, y_2)$ intersects column x, although

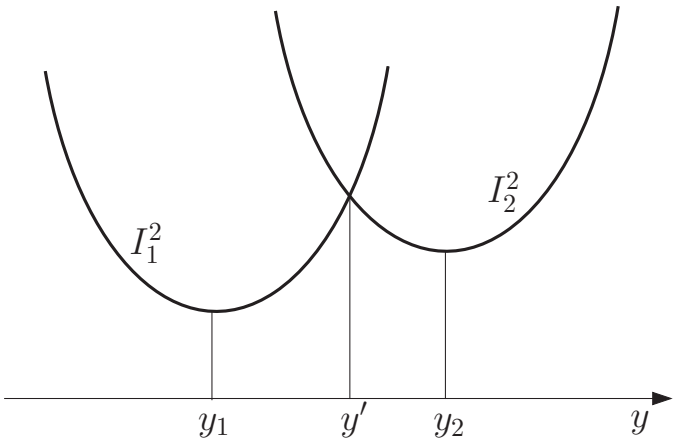


Figure 3.3: The distance squared along a column, showing the regions of influence of two background points.

the bisector may not necessarily be between y_1 and y_2. As the distance is only evaluated for integer values of y, it is not necessary to know the precise location of the intersection, only which two pixels it falls between. This means that integer division may be used, and the remainder or fractional part discarded. If the numerator is positive, the number calculated is the last pixel under the influence of y_1. If negative, it is the first pixel under the influence of y_2.

Assume that the image is being scanned in the increasing y direction. Now consider adding a third background point $I(x_3, y_3)$, where $y_2 < y_3$ with intersection between parabolas 2 and 3 at y'' . If $y' < y''$ then there are three regions of influence, corresponding to the sets of points nearest to each of the background pixel. However if $y'' < y'$ then background point 2 has no influence in column x because its parabola will be greater than the minimum of parabolas 1 and 3 at every point. The boundary between parabolas 1 and 3 may then be found from equation 3.12.

Extending this search to N points would require N^2 tests in the worst case. However, by making use of the fact that the points are ordered, and scanning in only one direction at a time, the number of tests may be reduced to N.

The basic data structure used to maintain the information is a stack. Each stack item contains a pair of values (y, y^I) representing respectively a row number, y, and the maximum row which y row influences, y^I. The stack is initialized as $(0, N)$. This is saying that in the absence of further information, the first row will influence the whole image.

For each successive row, equation 3.12 is evaluated with y_1 as the row number from the top of stack, and y_2 the new row. There are three cases of interest:

1. $y' > N$. The boundary of influence between y_1 and y_2 is past the end of the image, so the new row will have no influence.

2. $y' > y_0^I$, where y_0^I is the influence from the previous stack entry, and corresponds to the start of the influence of row y_1. In this case row y_1 has a range of influence, and y_1^I is set to y' . The new row, y_2 is added to the stack, with y_2^I set to N.

3. $y \leq y_0^I$. In this case, row y_1 has no influence on the distance transform in this

column. Row y_1 is therefore popped off the top of the stack, and equation 3.12 is reevaluated with the new top of stack. This process is repeated until either the stack is empty (the new row will influence all previous rows) or case 2 is met (the start of the influence of the new row has been found).

After processing all of the rows, the boundary points between each of the key influencing rows is known. Since the row that will provide the minimum distance for each row is known, it is simply a matter of using the stack as a queue for a second pass down the column to evaluate the distances.

Since equation 3.12 may be evaluated multiple times for each row, it is necessary to demonstrate that this algorithm actually executes in linear time. Observe that in cases 1 and 2, equation 3.12 is evaluated once as the new row is added (or discarded). If case 3 is selected, one existing row will always be eliminated from the stack for each additional time, then equation 3.12 is evaluated. These subsequent evaluations may therefore be associated with the row being eliminated rather than the row being added. As a row may only be eliminated once at most, the total number of times that equation 3.12 is evaluated will be between N and $2N$. Therefore the total number of operations is proportional to N and the above algorithm executes with time proportional to the number of pixels in the image.

3.3 Efficient Implementation

First note that both equations 3.8 and 3.12 involve squaring operations. Rather than calculate this each time using multiplications, a lookup table can be precalculated and used. The maximum size of this lookup table is the maximum of the number of rows or columns in the image. Rather than use multiplications to populate the lookup table, it may be filled as follows:

$$x^2 = \begin{cases} 0 & x = 0 \\ (x-1)^2 + 2x - 1 & x > 0 \end{cases} \tag{3.13}$$

3.3.1 Row Scan

The minimum operation of equation 3.10 may be eliminated if the width of the object on row y is known. So as the row is scanned, the distance from the left edge of the object is determined, as in equation 3.9. However, when the next background pixel is encountered, the width of the object is known from the distance of the last pixel filled. Therefore as the line is filled back, it only needs to be filled back half of the width. This right-to-left fill is performed immediately rather than waiting for a second pass since the position of the right edge is now known.

Storing the squared distance rather than store just distance is more useful since it needs to be squared in equation 3.12.

3.3.2 Column Scan–Pass 1

The most expensive operation within the column scanning is the division in equation 3.12. Therefore the speed may be increased by reducing the number of times equation 3.12 is evaluated. Since, in general, many of the rows are eliminated, if those rows may be eliminated beforehand this can save time. Separating the scan into two passes, first down the column and then up the column, and propagating the distances while scanning can achieve this.

Referring to equation 3.12, observe that if $I_2 \leq I_1$ then $y' < y_2$. This implies that if the image is being scanned in the positive direction, the intersection point has already been passed, and as far as the rest of the scan is concerned, y_1 may be eliminated. For a typical image, this implies that approximately half of the initial scans in the first pass may be eliminated by a simple comparison.

Secondly, in assigning the distances during the first pass, if the distance on any row is decreased, that row will have no influence in the second pass. This is because any background pixel that causes such a reduction must be closer to that object pixel (for the reduction to occur) and also be in a row above it (to have influence in the first pass). In the second pass, back up the column, if equation 3.12 was applied to those two rows, the boundary would be below the row that was modified. This implies that it will have no influence in the upward pass. Therefore all such rows may be ignored in the second pass. This may be accomplished by setting the minimum row distance of that pixel to ∞.

Taking these into account, the first column pass may be implemented as follows:

1. **Skip over background pixels–they will have zero distance. Reset the stack and push the row number of the last background pixel onto the stack. To avoid scanning through these pixels in the second pass, the location of the first background pixel may be recorded in a list.**

2. **If $I^2(x, y)$ is infinite (there are no background pixels in this row), skip to step 10 to update the distance map.**

3. **If the stack is empty, skip to step 5. Otherwise calculate the new distance that would be propagated to the current row from the bottom of the stack, y_c:**

$$I^2_{new} = I(x, y_c)^2 + (y - y_c)^2 \tag{3.14}$$

4. **If $I^2_{new}(x, y) < I^2(x, y)$ then the previous rows have no influence over the current row. Therefore the complete stack is reset, and the current row number is pushed onto the stack. Proceed with processing the next pixel (step 11).**

   Steps 5 to 9 consist of a loop that updates the stack.

5. **If the stack is empty, push the current row onto the stack, and go to step 10.**

6. **If the current distance is less than that on the row pointed to by the top of stack, ($I^2(x,y) < I^2_{tos}(x,y)$) then the current top of stack will no longer have any influence. Pop the entry from the top of the stack, and loop back to step 5.**

7. **Calculate the influence boundary between the top of stack and the current row using equation 3.12. If this boundary is past the end of the image, the current row will have no influence. Set $I^2(x,y)$ to ∞ and skip to step 10.**

8. **If the boundary is greater than that of the previous stack entry (top-of-stack–1, if it exists) then adjust the boundary on the top of stack to the value just calculated. Push the current row onto the stack and skip to step 10.**

9. **Otherwise the current top of stack has no influence, so pop the top entry from the stack and loop back to step 5.**

10. **If the new value was not calculated in step 3, then calculate it now (if the stack is empty, skip to step 11). This value is written to the output image, $I^2_d(x,y)$. If $I^2_{new}(x,y) < I^2(x,y)$ then set $I^2(x,y)$ to ∞ because this row will not have any influence on the second pass. If the boundary of influence of the entry on the bottom of the stack ends at the current row, then the entry may be pulled from the bottom of the stack (that entry will have no further influence on the rest of the column).**

11. **Move to the next pixel in the current row, and repeat.**

At this stage, all of the distances that need to be propagated down the image will have been propagated. Most of the rows that are unlikely to influence the propagation back up the image have also been eliminated.

3.3.3 Column Scan–Pass 2

The second column scan, from the bottom of the image to the top, proceeds in the same manner as the first scan. The exceptions are:

Step 1: Rather than scanning through the background pixels a second time, use the previously recorded top of the run.

Step 4: Also check if $I^2_{new}(x,y) > I^2_d(x,y)$. In this case, the distance being propagated up will no longer have any influence (the pixels have already been set with a lower distance). Therefore clear the stack, and continue scanning with the next pixel (step 11).

Steps 7 and 10: $I^2(x,y)$ does not need to updated, as this is not used any more.

3.3.4 Analysis of Complexity

Scanning through the image requires 1 increment and 1 comparison for every pixel visited. During the row pass, the whole image is scanned once in the left to right direction. Half of the object pixels are scanned a second time from right to left to update the distance from the right edge. Testing to see if a pixel is object or background requires 1 comparison. While the object pixels are being updated, a separate counter is maintained to keep track of the distance, requiring 1 addition, and 1 squaring operation (via table lookup).

For the column scanning, the exact complexity of the algorithm is made more difficult to calculate by the loop in steps 5 to 9 of the column pass. However, it was argued that equation 3.12 would be evaluated somewhere between 1 and 2 times per object pixel on average. The worst case is actually less than 2 because that would imply that no row had any influence! The average gains made by splitting the column analysis into two passes will not necessarily result in gains in the worst case.

The whole image is scanned during the first pass of column scanning. This results in 1 increment and 1 comparison per pixel, plus a test for a background pixel at each pixel. In the second pass, only the object pixels are processed.

The tests in steps 2-4 require 1 comparison each, and are executed during both passes through the object rows. equation 3.14 is evaluated either on step 3 or 10, and requires 2 additions, 1 squaring operation and 1 stack access to obtain the row to be propagated. It will be evaluated at most twice per object pixel (once in each pass). The test of step 4 ensures that the loop (steps 5-9) will only be entered in only one of the passes. Therefore the operations in the loop may be executed up to 2 times per object pixel. Accessing the top of stack (an array lookup) is performed in steps 6 and 8 (with a subtraction in step 8 to access the previous entry). Evaluation of equation 3.12 requires 3 additions, 1 squaring operation, 1 division, and 1 stack access. The tests in steps 5-8 require 1 comparison each. As a result of the tests, a value is either pushed into the stack (an addition to adjust the stack pointer, and a stack access) or popped off the stack (adjusting the stack pointer only). As these are also associated with the looping, they will be executed once each per object pixel in the worst case. Finally, in step 10, there are 2 comparisons, a stack access, and an addition to adjust the stack if the bottom entry has no further influence.

It demonstrates that this linear-time Euclidean squared distance transform may be implemented efficiently in terms of computation using only integer arithmetic.

Chapter 4

Binarization of Fingerprint Images

4.1 Introduction

AFIS are usually based on minutiae matching [28, 40, 48, 53]. Minutiae, or Galton's characteristics [34] are local discontinuities in terms of terminations and bifurcations of the ridge flow patterns that constitute a fingerprint. These two types of minutiae have been considered by Federal Bureau of Investigation for identification purposes [91]. Most of the fingerprint matching techniques require extraction of minutiae that are the terminations and bifurcations of the ridge lines in a fingerprint image. Crucial to this step, is either detecting ridges from the gray-level image or binarizing the image and then extracting the minutiae. Our work adopts the binarization of fingerprint images (see Figure 4.1 for an illustration).

Most of the previous algorithms designed for fingerprint binarization [17, 64, 69] are heuristics, in that they do not start with a definition of an optimal threshold. In contrast, we define a condition for an optimal threshold based on equal widths of ridges and valleys and then design a combinatorial algorithm for finding such an optimal threshold. Width is estimated using EDT. Most of the methods developed in fingerprint analysis use arbitrarily selected parameters and in many cases they are not justified; it is just experimentally selected. Our work depends minimally on an arbitrary choice of parameters because EDT value adapt itself to the data.

Our work proposed in this chapter involves binarization of fingerprint images that is to be preceded by an enhancement step. So, below we discuss briefly enhancement. Also, we briefly discuss and review segmentation and binarization methods applied to fingerprint images.

4.1.1 Enhancement of Fingerprint Images

Fingerprint images require specialized enhancement techniques owing to their inherent characteristics like high noise content, particular structural content of alternating ridges and valleys. Conventional image processing enhancement techniques are not very suitable for a fingerprint image [26]. Fingerprint image enhancement algorithms are available both for binary and gray level images. A binary fingerprint image consists of ridges marked as

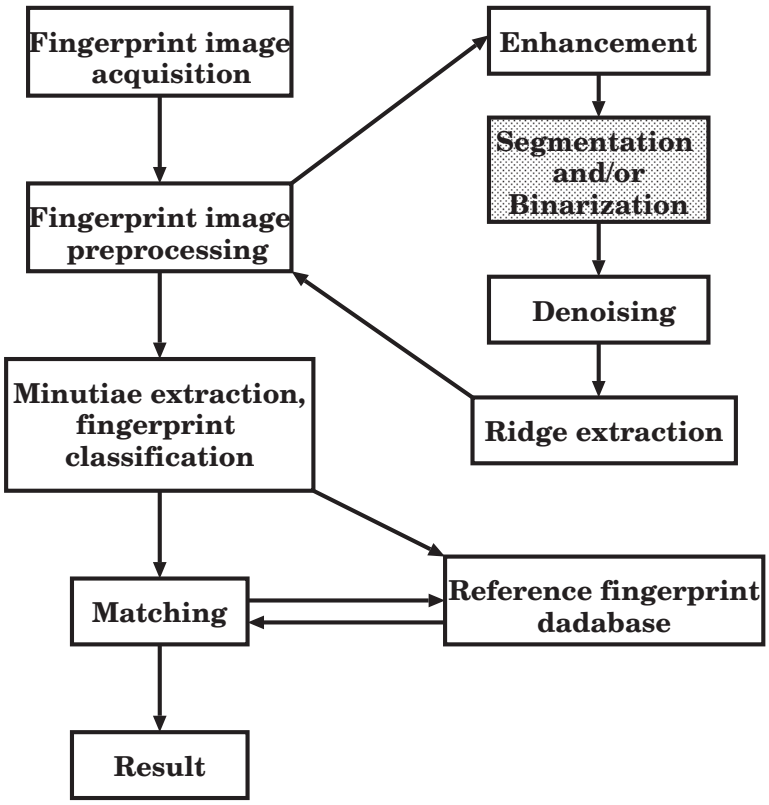


Figure 4.1: A flowchart showing different phases of fingerprint analysis. The highlighted block shows our work in this chapter.

object (1) pixels and the rest as background pixels (0). Hung [26] designed an algorithm for enhancing a binary fingerprint image based on the structural information of its ridges. Ridge widths are normalized based on some region index. Ridge breaks are corrected using the dual relationship between ridge breaks and valley bridges. However, obtaining a binary fingerprint image from a gray-tone image involves inherent problems of binarization and thinning or ridge extraction procedures [15]. Thus, most of the enhancement algorithms are designed for gray-level fingerprint images. The much widely used PCASYS package [13] uses an enhancement algorithm described earlier [24]. It involves cutting out subregions of the images (a 32×32 block to be specific), taking their FFT and suppression of a band of low and high frequency components followed by some non-linear operations in the frequency domain and transforming it back to the spatial domain. This algorithm was also used by Kovács-Vajna [53]. We have also used this enhancement algorithm in our work owing to its simplicity and elegance.

4.1.2 Segmentation of Fingerprint Images

In literature concerning fingerprints, some authors have used the term segmentation to mean the process of generating a binary image from a gray-level fingerprint image. But, as suggested in [62], the most widely held view about segmentation of a fingerprint image is the process of separation of fingerprint area (ridge, valley and slope areas in between ridge and valley areas) from the image background. The process of segmentation is

useful to extract meaningful areas from the fingerprint, so that features of the fingerprint are extracted from these areas only. Fingerprint images are characterized by alternating spatial distribution of varying gray-level intensity values of ridges and ravines/valley. This pattern is unique to a fingerprint area compared to the background which does not have this spatial distribution of gray-level values. Also, global thresholding for segmentation does not work as the spatial distribution of gray-level values keeping their alternating structure intact, can vary in the absolute magnitude of their gray-level values. Thus, local thresholding is needed. Exploitation of these property have been the key of most of the segmentation algorithms. O'Gorman and Nickerson [37] used a $k \times k$ spatial filter mask with an appropriate orientation based on user inputs for labeling the pixels as foreground (crest) or background. Mehtre and Chatterjee [63] described a method of segmenting a fingerprint image into ridge zones and background based on some statistics of local orientations of ridges of the original image. A gray-scale variance method is used in the image blocks having uniform gray-level, where the directional method of segmentation fails. Ratha et al. [69] used the fact that noisy regions show no directional dependence, whereas, fingerprint regions exhibit a high variance of their orientation values across the ridge and a low variance along the ridge to design a segmentation algorithm that works on 16×16 block. Maio and Maltoni [61] used the average magnitude of gradient values to discriminate foreground and background regions. The idea behind this is that fingerprint regions are supposed to have more edges than background region and as such would have higher gradient values.

4.1.3 Binarization of Fingerprint Images

A general problem of image binarization is to obtain a threshold value so that all pixels above or equal to the threshold value are set to object pixel (1) and below the threshold value are set to background (0) [84]. Thresholding can be done globally where a single threshold is applied globally or locally where different thresholds are applied to different image regions. Images, in general, have different contrast and intensity, and as such local thresholds work better. The thresholding problem can be viewed as follows. Given an image I with $N \times N$ pixel entries, and gray-level intensity value g ranging from 0, 1, ... to $M - 1$, select a value $t \in [0, M - 1]$ based on some condition so that a pixel (i, j) is assigned a value of 1 if the gray-level intensity value is greater or equal to t, else assign 0 to the pixel (i, j). The condition mentioned above is decided based on the application at hand. The binarization methods applicable to fingerprint images draw heavily on the special characteristics of a fingerprint image. Moayer and Fu [64] proposed an iterative algorithm using repeated convolution by a Laplacian operator and a pair of dynamic thresholds that are progressively moved towards an unique value. The pair of dynamic thresholds change with each iteration and control the convergence rate to the binary pattern. Xiao and Raafat [92] improved the above method by using a local threshold, to take care of regions with different contrast, and applied after the convolution step. Both of these methods requiring repeated convolution operations are time consuming and the final result depends on the choice of the pair of dynamic thresholds and some other design parameters. Coetzee and Botha [17] proposed an algorithm based on the use of edges in conjunction with the gray-scale image. The resultant binary image is a logical OR of two binary images. One binary image is obtained by a local threshold on the gray scale image

and the other binary image is obtained by filling in the area delimited by the edges. The efficiency of this algorithm depends heavily on the efficiency of the edge finding algorithm to find delimiting edges. Ratha et al. [69] proposed a binarization approach based on the peak detection in the gray-level profiles along sections orthogonal to the ridge orientation. The gray-level profiles are obtained by projection of the pixel intensities onto the central section. This heuristic algorithm though working well in practice has a deficiency that it does not retain the full width of the ridges, and as such is not a true binary reflection of the original fingerprint image.

Fingerprint images are characterized by alternating spatial distribution of gray-level intensity values of ridges and ravines/valleys of almost equal width. In this work, we propose a combinatorial algorithm for binarization of fingerprint images using linear time Euclidean distance transform algorithms. Most of the previous algorithms discussed here are heuristics in that they do not start with a definition of an optimal threshold. In contrast, we define a condition for an optimal threshold based on equal widths of ridges and valleys. Computing the width of arbitrary shapes is a non-trivial task. So, we estimate width using distance transform and provide an $O(N^2 \log M)$ time algorithm for binarization where M is the number of gray-level intensity values in the image and the image dimension is $N \times N$. With M for all purposes being a constant, the algorithm runs in near-linear time in the number of pixels in the image. We show how distance transform can be used as a measure for width and then design an algorithm to efficiently compute the threshold for binarization. Another significant advantage of our algorithm is that binarization and fingerprint area segmentation can be done simultaneously.

4.2 Euclidean Distance Transform

A two-dimensional binary image I of $N \times N$ pixels is a matrix of size $N \times N$ whose entries are 0 or 1. The pixel in a row i and column j is associated with the Cartesian co-ordinate (i, j). For a given distance function, the *Euclidean distance transform* of a binary image I is defined in [11] as an assignment to each background pixel (i, j) a value equal to the Euclidean distance between (i, j) and the closest feature pixel, i.e. a pixel having a value 1. Breu et al. [11] proposed an optimal $O(N \times N)$ algorithm for computing the *Euclidean distance transform* as defined using Voronoi diagrams. Construction and querying the Voronoi diagrams for each pixel (i, j) take time $\theta(N^2 \log N)$. But, the authors use the fact that both the sites and query points of the Voronoi diagrams are subsets of a two-dimensional pixel array to bring down the complexity to $\theta(N^2)$. In [39], Hirata and Katoh define *Euclidean distance transform* in an almost same way as the assignment to each 1 pixel a value equal to the Euclidean distance to the closest 0 pixel. The authors use a bi-directional scan along rows and columns of the matrix to find out the closest 0. Then, they use an envelope of parabolas whose parameters are obtained from the values of the bi-directional scan. They use the fact that two such parabolas can intersect in at most one point to show that each parabola can occur in the lower envelope at most once to compute the *Euclidean distance transform* in optimal $\theta(N^2)$ time. In keeping with the above, we define two types of *Euclidean distance transform* values. The first one $DT_{1,0}$ is the same as the above. The second one is $DT_{0,1}$ which is the value assigned to a 0 pixel equal to the Euclidean distance to the nearest 1 pixel. Using the results given in [39], we

have the following fact:

Fact 1 *Both $DT_{1,0}$ and $DT_{0,1}$ can be computed in optimal time $O(N^2)$ for an $N \times N$ binary image. Also, the values of both $DT_{1,0}$ and $DT_{0,1}$ are greater than or equal to* 1.

4.3 Euclidean Distance Transform and Width

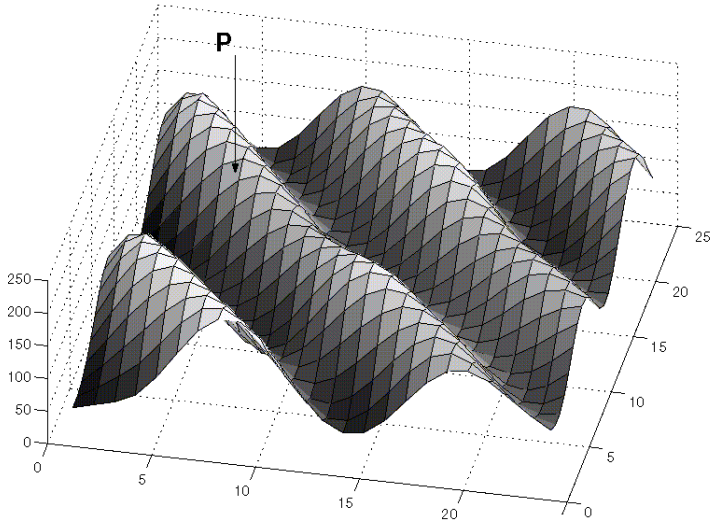


Figure 4.2: Magnified view of a part of the gray scale topology of a fingerprint image.

The fingerprint images are characterized by almost equal width ridges and valleys as shown in Figure 4.2. We will use this particular characteristic of the fingerprint image for binarization. Measuring the width for arbitrary shapes is a difficult, non-trivial problem. In this section, we model the problem in a continuous domain to show how distance transform can be used to find equal width ridges and valleys.

4.3.1 Model in The Continuous Domain

The fingerprint image can be modeled as shown in Figure 4.3. In the continuous domain, the image is a continuous function $f : (x, y) \rightarrow I\!R$. A cross section of this function along a direction perpendicular to the ridge increases till it reaches the ridge point which is a maxima, then decreases till it reaches the valley, which is a minima; and this cycle repeats. Let $t \in [0, M]$ be a threshold, such that if f is thresholded at t, and if the value of f is greater than t, it is mapped to 1, else to 0. See Figure 4.3. The highlighted part shown on the right is the part mapped to 1. After thresholding, the parts would be rectangles as shown in Figure 4.4. We compute the total distance transform values of the rectangles. Consider a rectangular object $ABCD$ of width w and height h, with $h > w$. The medial axis of this object is given by the line segments $\overline{AE}$, $\overline{BE}$, $\overline{EF}$, $\overline{FD}$, $\overline{FC}$. The medial axis divides the rectangular shape into four regions such that the nearest boundary line from any point in the region is determined. As an example, the region 1 has $\overline{AD}$ as

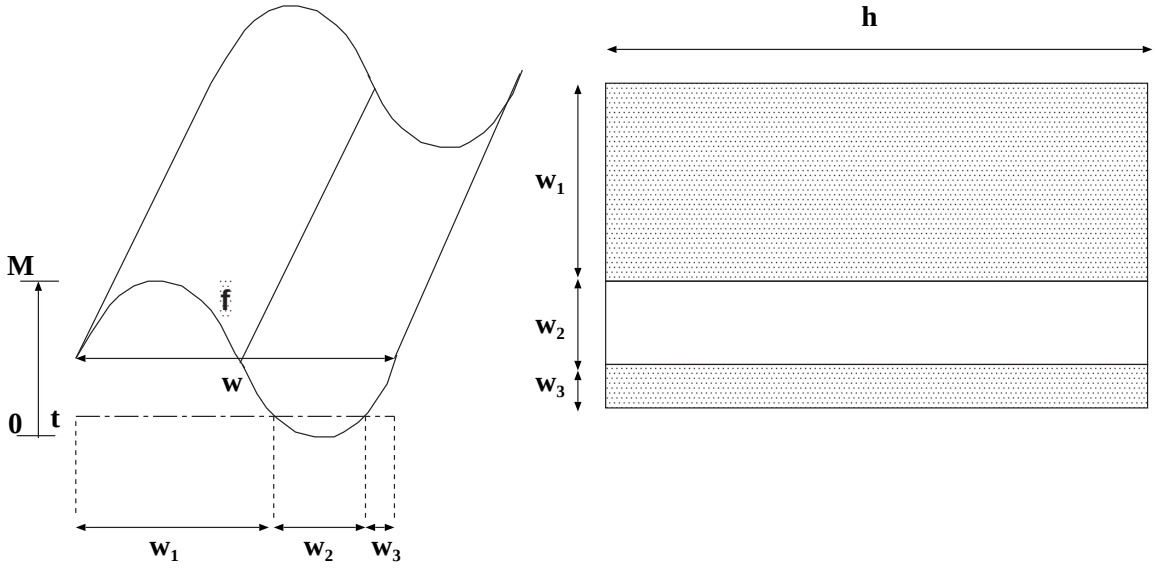


Figure 4.3: Diagram of the model.

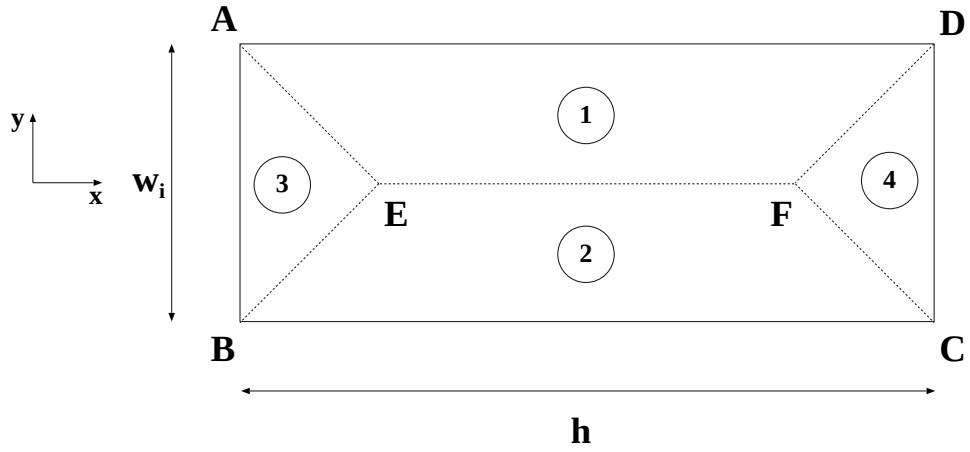


Figure 4.4: Diagram for computing total distance transform.

its nearest boundary line and region 3 has $\overline{AB}$ as its nearest boundary line. The total distance transform value for region 1 is

$$\int_0^{w_i/2} \int_{-y+w_i/2}^{y+(h-w_i/2)} (w_i/2 - y)\, dxdy = (w_i^2 h)/8 - w_i^3/12$$

Similarly, the total distance transform value for region 3 is

$$\int_{x-w_i/2}^{-x+w_i/2} \int_0^{w_i/2} x\, dxdy = w_i^3/24$$

So, the total distance transform value $\phi_{dt}(w_i)$ of the rectangle is

$$w_i^2 h/4 - w_i^3/12 = w_i^2/4(h - w_i/3)$$

Note that, the total distance transform value increases (decreases) with the increase (decrease) of width because $\phi_{dt}(w_i)' > 0$ and $h > w$. Now, the total distance transform $DT_{1,0}$

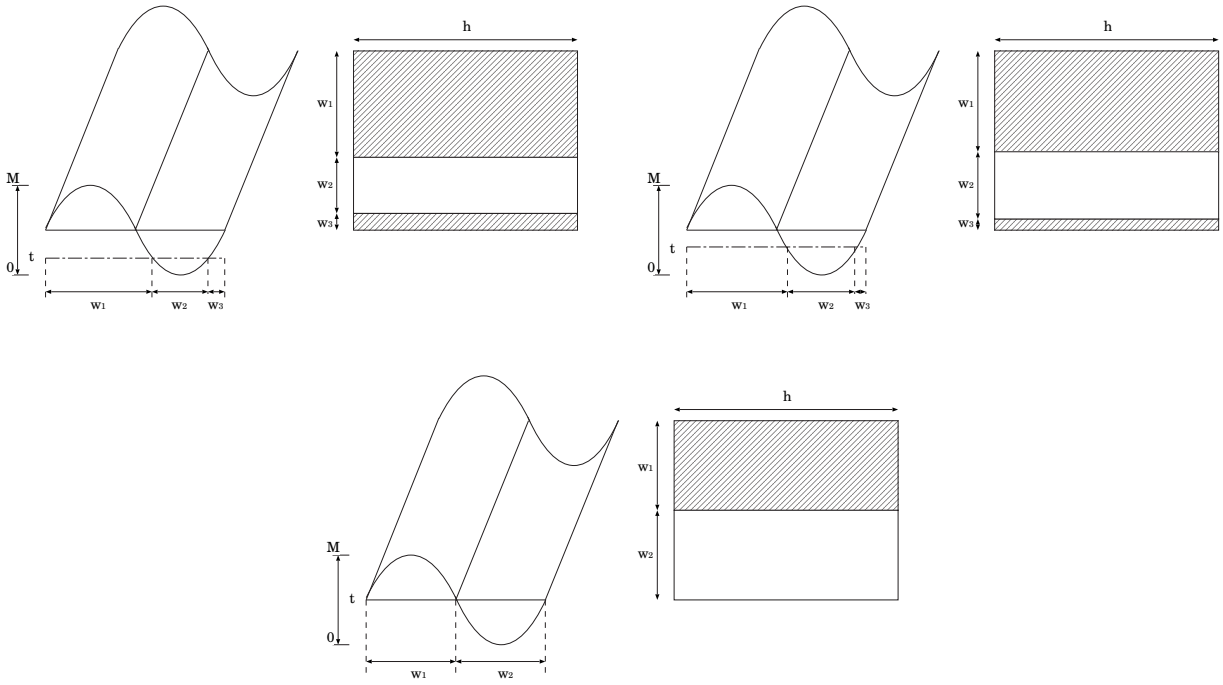


Figure 4.5: Sequence of binarization with increase of t.

is $w_1^2h/4 - w_1^3/12 + w_3^2h/4 - w_3^3/12$ and the total distance transform $DT_{0,1}$ is $w_2^2h/4 - w_2^3/12$. Now, as t increases, both w_1 and w_3 decrease and w_2 increases. This implies that with increase of t, $DT_{1,0}$ decreases and $DT_{0,1}$ increases (see Figure 4.5 for an illustration). So, $DT_{1,0}$ and $DT_{0,1}$ can intersect only once and evidently, $DT_{1,0}$ is equal to $DT_{0,1}$ when $w_1 = w_2$. That is, the optimal value of threshold is reached when $DT_{0,1} = DT_{1,0}$, implying $w_1 = w_2$. This simple analysis shows that total distance transform can be used as a measure of finding a threshold that gives equal width ridges and valleys. Our goal in this work is to find an optimal threshold to binarize the fingerprint image. The optimality criteria is given by the equal width of ridge and valley. So, more formally we have the following definition.

Definition 1 *The optimal threshold is a value $t \in [0, M]$ that binarizes the image such that the ridge width is equal to the valley width or sum total of distance transform values are equal.*

4.3.2 Discrete Image and Distance Transform

In the discrete model, the co-ordinates are discrete given by the pixel locations. The gray-level values g are also discrete taking values from 0 to $M - 1$. So, the observations from the previous subsection do not directly apply. But, the crucial observation from the previous subsection is that sum total of $DT_{1,0}$ values decreases with t and the sum total of $DT_{0,1}$ values increases with t. Then, the optimal threshold t can be obtained as that value of t that makes the width of the 1 region and 0 region equal and can be computed from the intersection of the curves of the sum total of $DT_{0,1}$ and $DT_{1,0}$ values. For, the analysis,

we make the following assumption. The pixels take the gray-level intensity values such that all the intermediate gray-level values between the maximum and the minimum are present. With that assumption, we have the following lemma.

Lemma 1 *The sum total of $DT_{1,0}$ values decreases with the threshold t. Similarly, the sum total of $DT_{0,1}$ values increase with the threshold t.*

The proof is easy. We know that each of the Euclidean distance transform values in the discrete domain is greater than or equal to 1 (see Fact 1). So, with the threshold t increasing, pixels in the binary image move from the regions of 1 to 0, thus making $DT_{1,0}$ and $DT_{0,1}$ decreasing and increasing respectively. Also, note that the assumption that the pixels take the gray-level intensity values such that all the intermediate gray-level values between the maximum and the minimum are present, ensures the strictly decreasing and increasing relations of sum total of $DT_{1,0}$ and $DT_{0,1}$ values. Otherwise, it would have been non-increasing and non-decreasing respectively.

Also, in the discrete case, we may not be able to locate a single value, where the functions of sum total of $DT_{1,0}$ and $DT_{0,1}$ meet. So, we modify the definition of the optimal threshold in the discrete case as follows.

Definition 2 *The optimal threshold can be two values t_1 and t_2 such that $t_2 - t_1 = 1$ and the sum total of $DT_{1,0}$ values is greater than the sum total of $DT_{1,0}$ values at t_1 and their relation reverses at t_2.*

With this definition in place, we are in a position to design the algorithm in the next section.

4.4 Algorithm

To take care of different contrast and intensity across different image regions, we apply local thresholding. We cut out sub-blocks of image region and apply the enhancement algorithm due to [24] followed by our binarization algorithm. Figure 4.6 is a visual illustration.

Algorithm for binarization
Input: A gray-level fingerprint image I with gray-level intensity varying from 0 to $M-1$, and of size $N \times N$;
Output: A thresholded binary image

1. **do for all** sub-block B_i of the image I;
2. Apply the enhancement algorithm given in [24];
3. $t_1 = 0$, $t_2 = M - 1$; $mid \leftarrow \lceil (t_1 + t_2)/2 \rceil$;
4. **do**
5. $mid \leftarrow \lceil (t_1 + t_2)/2 \rceil$;
6. Compute $SumDT_{1,0}^{mid}$ and $SumDT_{0,1}^{mid}$;
7. **if**$(SumDT_{1,0}^{mid} > SumDT_{0,1}^{mid})$ $t_1 \leftarrow mid$;

8. **else** $t_2 \leftarrow mid$;
while$(t_2 - t_1 > 1)$
9. Threshold obtained for binarization is t_1 or t_2;

Theorem 1 *The binarization algorithm under Definition 2 runs in time* $O(N^2 \log M)$.

The loop originating in Step 4 runs $O(\log M)$ times and the dominant computation is the computation of Euclidean Distance Transform and its sum which takes $O(N^2)$ time (see Fact 1). Thus the total time complexity of the binarization process is $O(N^2 \log M)$. With M, the number of gray-levels, being a constant for all practical purposes, the algorithm for binarization runs in time that is linear in the number of pixel entries which is $O(N^2)$.

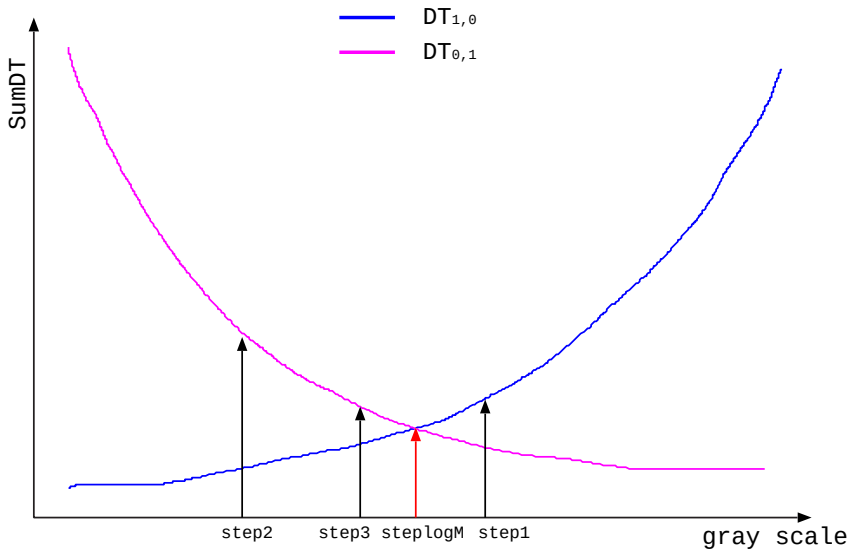


Figure 4.6: Diagram of binarization algorithm.

Remark 1 *Note that the monotonic curves corresponding to sum of* $DT_{1,0}$ *and* $DT_{0,1}$ *may not intersect when sum of* $DT_{1,0}$ *is always greater than sum of* $DT_{0,1}$. *This happens for image blocks which are noisy or do not have a proper gray level distribution, like a fingerprint image. We characterize such blocks as non-binarizable blocks. Thus, our binarization algorithm has an added property of segmenting the fingerprint image region.*

4.5 Previous Binarization Algorithms and Comparisons

Here, we review three previous works and compare these methods with ours.

4.5.1 Three Previous Major Binarization Algorithm

Moayer and Fu [64] proposed an iterative algorithm using repeated convolution by a Laplacian operator and a pair of global dynamic thresholds that are progressively moved towards an unique value. The pair of global dynamic thresholds changes with each iteration and controls the convergence rate to the binary pattern. This method, which requires repeated convolution operations, is time consuming and the final result depends on the choice of the pair of dynamic thresholds and some other design parameters. As local information can not be dealt with well, some impulsive noise needing further preprocessing will always remain.

Coetzee and Botha [17] proposed an algorithm that generates the resultant binary image as a logical OR of two binary images. One binary image is obtained by a local threshold on the gray scale image and the other binary image is obtained by filling in the area delimited by the edges. The efficiency of this algorithm depends heavily on the efficiency of the edge finding algorithm to find delimiting edges. As this method uses local thresholds, the results are fairly clear. But, on the flip side, this method uses many arbitrary parameters: (i) the authors use a $2d \times 2d$ (adaptive) window with $d = 2$; (ii) in the gray-scale window, they chose a threshold of 0.6 of the maximum gray value (see Ps. 1443 and 1444 of [17]). The choice of none of these parameters is explained. In effect, our experiments show that a threshold of 0.4 of the maximum gray value works better. Also, as the algorithm uses two windows (gray-scale and edge), the running time is prohibitively long.

Ratha et al. [69] proposed a binarization approach based on peak detection in the gray-level profiles along sections orthogonal to the ridge orientation. The gray-level profiles are obtained by projection of the pixel intensities onto the central section. This algorithm has two positive factors: (i) the orientation field computed for binarization is also used for the following steps; (ii) this algorithm is the fastest among all of the four methods (including ours) we compared. This method uses a fixed length line segment in certain orientations to binarize, and as such generates the following drawbacks: (i) If the ridges are very close in the image, the binarized ridges are merged together; (ii) Most of the bifurcations and even some ridges are broken by the oriented line segments. Again, like the other two algorithms, this method depends on many arbitrary parameters (see P. 1667 of [69]). Also, the authors did not mention anything about the size of the one-dimensional averaging mask used (see P. 1664 of [69]). This heuristic algorithm, though working well in practice, has a deficiency that it does not retain the full width of the ridges, and as such is not a true binary reflection of the original fingerprint image.

4.5.2 Comparisons

There are two distinct features of our binarization algorithm compared to the other three methods: (i) as we view binarization as an optimization process, we start with a very clear definition of an optimal threshold; this in turn, allows us to design a truly deterministic algorithm; (ii) our algorithm uses no arbitrary parameter; the choice of a 32×32 window for local thresholding (see Section 4.4) is dictated by the algorithm for enhancement in [24, 13]. As there is no mathematical way of comparing the results, below we give qualitative comparisons.

Table 4.1: Comparative results of quality and time.

Algorithm	Rank by quality		Time in secs.
	Mean	Variance	
Our method	1.12	0.0027	1.474
Moayer and Fu [64]	2.51	0.0057	2.092
Ratha et al. [69]	2.76	0.0359	0.543
Coetzee and Botha [17]	3.61	0.0233	38.769

Fingerprint images used

We used the fingerprint images from (i) NIST Special Database 4 [90], (ii) NIST Special Database 14 [13], (iii) Database B1 of FVC2000 [33], (iv) Database B2 of FVC2000 [33] and (v) fingerprint images scanned by the FUJITSU Fingerprint Sensor (model: FS-210u). The images of (i) and (ii) are of size 480×512. The images of (iii) and (v) are of size 300×300 and (iv) are of size 364×256. All of the images are of 500 dpi resolution. From each database, two images of good quality and two images of bad quality (four in all) were manually selected, thus making 20 images. The enhanced version of the selected images are shown in Figure 4.7. For each data set, the two images on the left are of good quality and the other two on the right are of bad quality. The quality was determined manually.

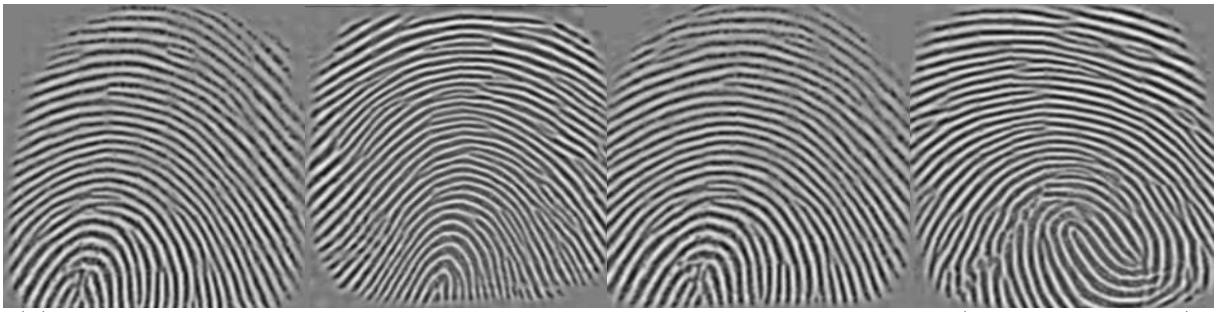
Qualitative comparison

First of all, the images were enhanced using the enhancement algorithm of [24]. Then, we ran all the four binarization algorithms on these 20 enhanced images on a Sun Blade 150, sparc, Solaris 5.9, 550 MHz (see Figures 4.8, 4.9, 4.11 and 4.10). For a qualitative comparison, we did the following. We asked 5 unbiased volunteers conversant with fingerprint technology to rank the resulting binarized images. Obviously, we kept the identity of the authors of all the algorithms a secret from them! The mean and variance of the ranks are shown in Table 4.1. It can be seen that on an average the users ranked the output of our binarization algorithm better than others. As to time, our deterministic algorithm ranks second; the fastest being the algorithm from Ratha et al. [69].

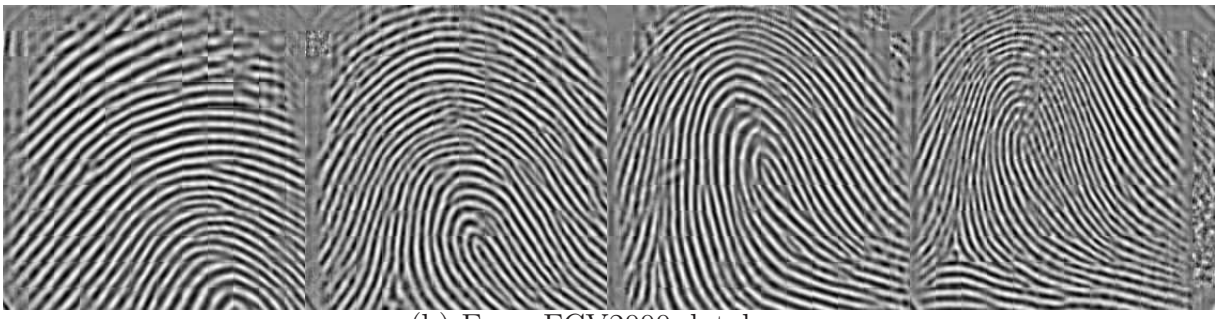
4.6 Discussion and Conclusion

We have developed a combinatorial algorithm for binarization of fingerprint images expoiting the fingerprint characteristics of equal width ridge and valleys. We used Euclidean Distance Transform as a measure of width as determining width for arbitrary discrete shapes is a non-trivial task. We have reported relevant results from standard image databases widely used. But, the definition 2 used for our algorithm has a drawback in realistic terms. During the acquisition of fingerprints, ridges, being the elevated structures on the finger, exert more pressure on the device making the acquisition. And as such, the widths of the ridges should be greater than the width of the valley for a more realistic model. But, still the lemma 1 will hold and the algorithm instead of trying to find the crossover

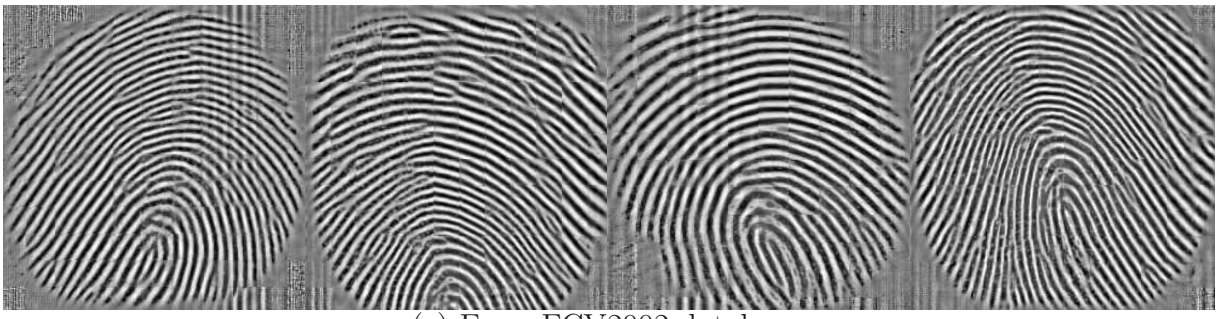
point of sum total of $SumDT_{1,0}$ and $SumDT_{0,1}$ will terminate when $SumDT_{1,0}$ is greater than $SumDT_{0,1}$ by a certain ϵ. Determining this ϵ from real fingerprint images is a future problem we would like to address.



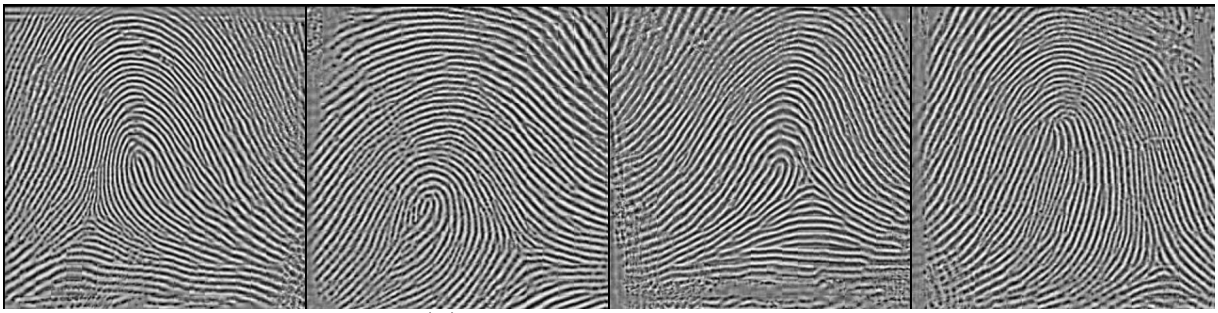
(a) Fingerprint images scanned by the FUJITSU Fingerprint Sensor (model: FS-210u).



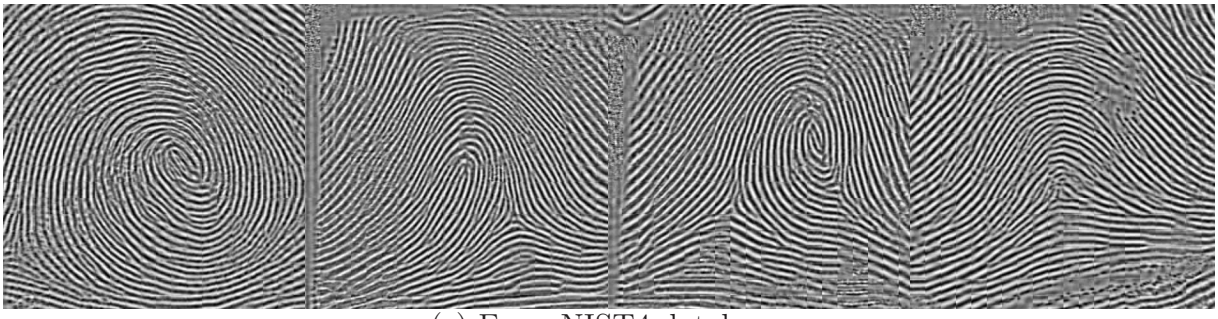
(b) From FCV2000 database.



(c) From FCV2002 database.



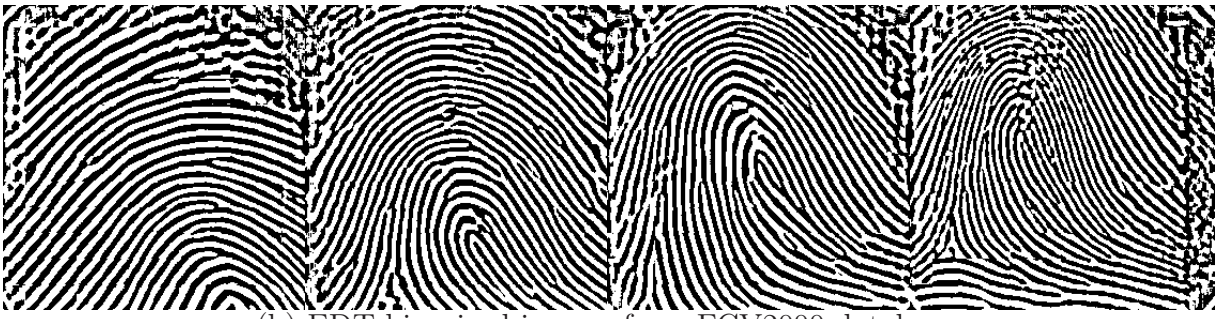
(d) From NIST14 database.



(e) From NIST4 database.

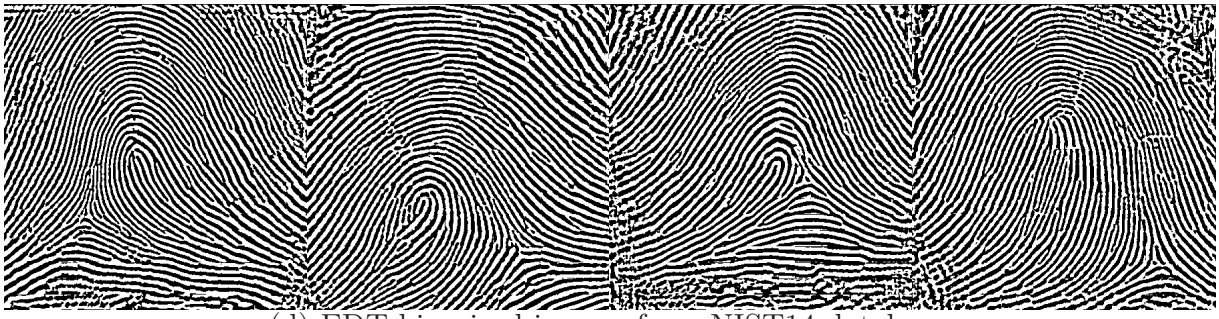
Figure 4.7: Enhanced Images from 5 databases.

(a) EDT binarized images from FUJITSU database.

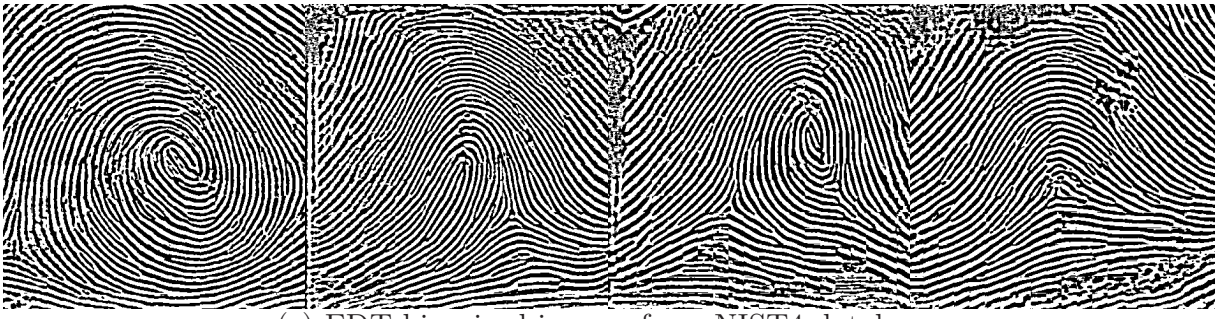


(b) EDT binarized images from FCV2000 database.

(c) EDT binarized images from FCV2002 database.



(d) EDT binarized images from NIST14 database.

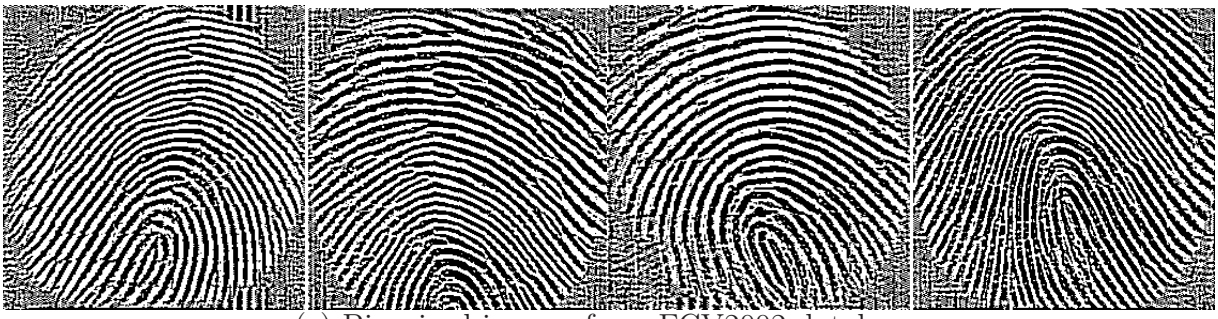


(e) EDT binarized images from NIST4 database.

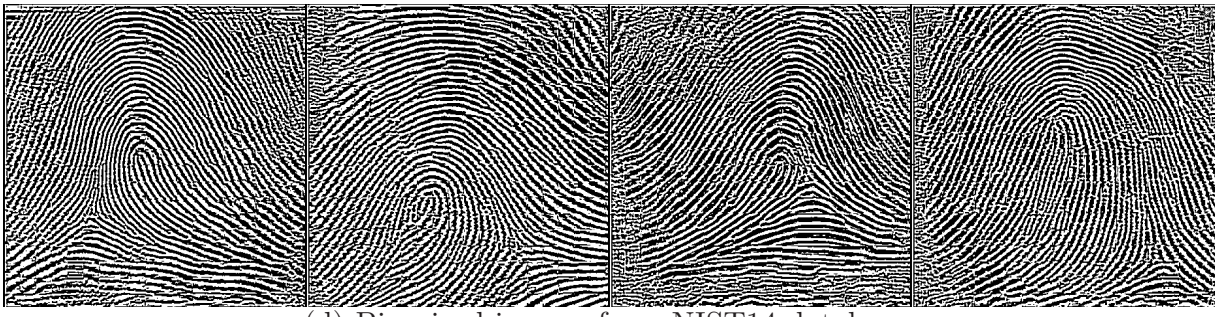
Figure 4.8: Binarized images by EDT.

(a) Binarized images from FUJITSU database.

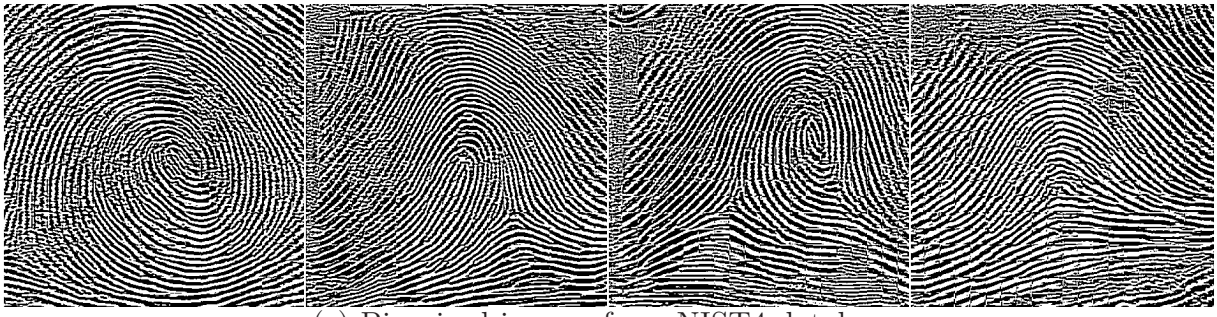
(b) Binarized images from FCV2000 database.



(c) Binarized images from FCV2002 database.



(d) Binarized images from NIST14 database.



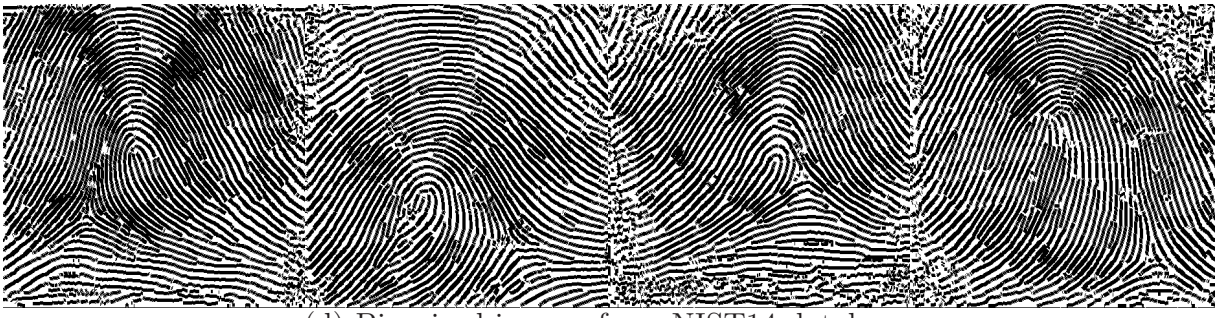
(e) Binarized images from NIST4 database.

Figure 4.9: Binarized images by Moayer and Fu's algorithm.

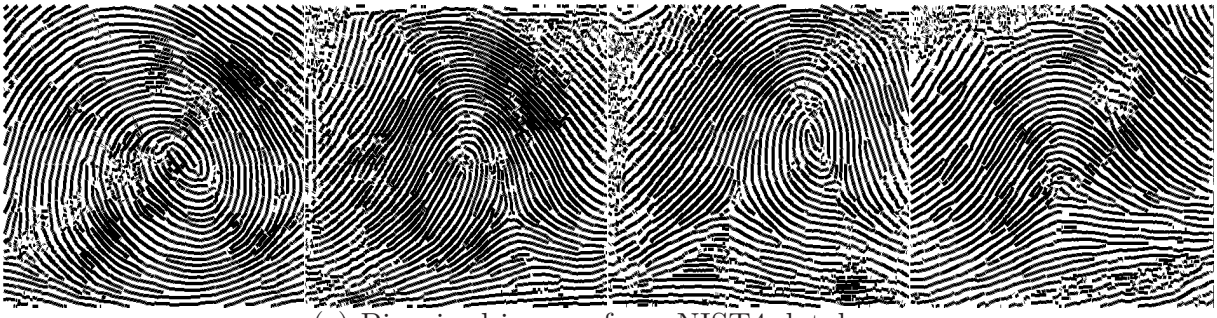
(a) Binarized images from FUJITSU database.

(b) Binarized images from FCV2000 database.

(c) Binarized images from FCV2002 database.



(d) Binarized images from NIST14 database.



(e) Binarized images from NIST4 database.

Figure 4.10: Binarized images by Ratha et al's algorithm.

(a) Binarized images from FUJITSU database.

(b) Binarized images from FCV2000 database.

(c) Binarized images from FCV2002 database.

(d) Binarized images from NIST14 database.

(e) Binarized images from NIST4 database.

Figure 4.11: Binarized images by Coetzee and Botha' algorithm.

Chapter 5

Eliminating Impulsive Noise and Useless Components

5.1 Introduction

The performance of fingerprint recognition relies heavily on the quality of the input fingerprint image. However, in practice, due to skin conditions (e.g., wet or dry), sensor noise, incorrect finger pressure, and inherently low-quality fingerprints, a significant percentage of fingerprint images contain a lot of noise (see Figure 5.6(a)). Noises in fingerprint images fall into two categories: *impulsive noise* ("salt and pepper" noise) and *useless components* (see Definition 4). Note that useless components are often mistaken for the terminations that are an essential minutia of a fingerprint. In this chapter, we are going to introduce our linear time algorithm for binary fingerprint image denoising using Euclidean distance transform. Since the information of distance transform values can be obtained directly from the binarization phase (see Figure 5.1 for an illustration). The results show that using this method on fingerprint images with impulsive noise and useless components is faster than existing denoising methods and achieves better quality than earlier methods.

Several techniques have been developed for eliminating impulsive noise. Ratha, Chen, and Jain [69] implement a morphological opening in which the structuring element is a small box oriented according to the local ridge orientation. Wahab, Chin, and Tan [88] correct the binary image at locations where orientation estimates deviate from their neighboring estimates. This correction is performed by substituting the noisy pixels according to certain oriented templates. Ikeda et al. [45] use morphological operators to enhance ridges and valleys in the fingerprint binary image. Since the first two methods consider the orientation, a fine selection of the directional filters is necessary. Although the third method employs an isotropic constructing element and as a result keeps the original shape of the fingerprint, the impulsive noise cannot be completely eliminated. However, the time complexity of these three is at least $O(N^2 \times d^2)$ for an image with $N \times N$ pixel entries and a filter whose radius is d. It is time consuming. In this chapter, we first propose a simple and linear time complexity method, which employs generalized morphological operators (GMO) [1] based on distance transform and integral image [87], to eliminate the impulsive noise.

Up to now, there has been little in the literature with regard to eliminating the useless

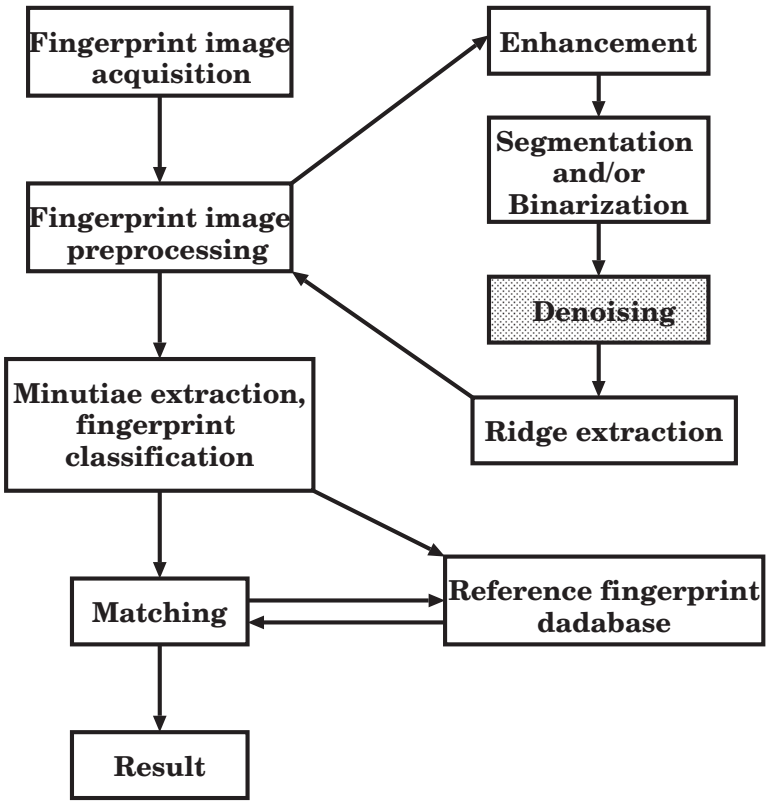


Figure 5.1: A flowchart showing different phases of fingerprint analysis. The highlighted block shows our work in this chapter.

components. In order to do so, the structuring element chosen must have a good fit. There are three existing categories of methods for choosing the optimal or appropriate SEs. S. Fejes and F. Vajda's [29, 30, 31] algorithm employs the least mean square. It needs a reference image, which means this method will first do a training phase. But during fingerprint recognition process, it is unlikely that system could provide a reference image. Anelli, Loncaric and Dhawan's [2, 59, 58] algorithm uses a genetic algorithm (GA) to choose an optimal SE. Although GA really can find the optimal results with correct criteria, it is a time-consuming algorithm, and fingerprint recognition is a real time application. GA cannot satisfy this realistic restriction. T. Kikuchi and S. Murakami's [52] algorithm is based on the standard deviation of a linear SE with directionality. In their method, the length of the linear SE is determined by experience. In contrast, we define a condition for eliminating useless components based on the fact that the width of useless component must be less than the average width of finger ridges. We show how distance transform can be used as a measure for width and then design an algorithm to efficiently determine the size of the SE.

5.2 Ordinary and Generalized Morphological Operators

5.2.1 Ordinary Morphological Operators (OMO)

In mathematical morphology, signal transformations are called morphological filters, which are nonlinear operators that locally modify the geometrical features of signals. More details can be found in Serra and Soille [78].

Let $B \subset Z^2$ be a simple compact set of small size called structuring element. F denotes a set of foreground pixels (black pixels) and F^c denotes the background (white pixels).

Translation:

The translation of a set F by a point x denoted by $F + x$ is defined as:

$$F + x = \{f + x : \exists f \in F\} \tag{5.1}$$

Where F denote a set of foreground pixels (Black pixels) and F^c denote background (white pixels).

Reflection:

$\breve{B}$ is the reflection of B given by:

$$\breve{B} = \{x : \exists b \in B; x = -b\} \tag{5.2}$$

Erosion:

The fundamental operation of mathematical morphology is erosion. The erosion of set F by set B is denoted by $F \ominus B$ and is defined by:

$$F \ominus B = \bigcap \{F + b : \exists b \in \breve{B}\} \tag{5.3}$$

Dilation:

The second most basic operation of binary mathematical morphology is dilation. It is a dual operation to erosion, meaning that it is defined via erosion by set complementarily. The dilation of set F by B is denoted by $F \oplus B$ and is defined by :

$$F \oplus B = \bigcup \{F + b : \exists b \in B\} \tag{5.4}$$

There are two secondary operations that play a key role in morphological image processing, these being opening and its dual, closing.

Opening:

The opening of image F by image B is denoted by $F \circ B$ and is defined as erosion followed by dilation, namely:

$$F \circ B = (F \ominus B) \oplus B \tag{5.5}$$

Closing:

The closing of F by B is denoted by $F \bullet B$ and is defined as dilation followed by erosion, namely:

$$F \bullet B = (F \oplus B) \ominus B \tag{5.6}$$

From the properties of morphological operators, it is obvious that the shape and size of the structuring element determine the nature and degree of the morphological transformation. The most common structuring elements are horizontal cross, diagonal cross (4-connected set) and 3×3 matrix (8-connected set), respectively (see the Figure 5.2).

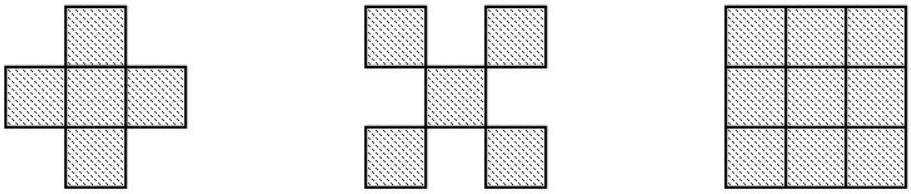


Figure 5.2: Structuring element: Left (1): Horizontal cross. Middle (2): Diagonal cross. Right (3): 3×3 matrix.

Morphological operators detailed so far, which in the following will be called ordinary morphological operators (OMO), consider only the intersection of the structuring elements B with F (or F^c). Therefore, when a large morphological kernel is used, the ordinary morphological operators have excessive operation (e.g., erosion, dilation). For this reason, it is suggested to define new morphological operators that generalize the ordinary morphological operators, thus achieving control over their strictness. They are described briefly in the following subsection.

5.2.2 Generalized Morphological Operators (GMO)

The morphological dilation of F by B can be generalized by combining the size of the intersection into the dilation process. In that sense, the dilation of F would be done if and only if the intersection between F and the shifted $\breve{B}$ is big enough. The obtained advantage of the generalized dilation is avoiding excessive dilation caused by small intersections. That is, the mass of an intersection should be big enough to cause a change.

The **generalized dilation** of F by B with strictness s is defined by:

$$F \overset{s}{\oplus} B = \{x : \#(F \bigcap \breve{B}) \geq s\}; s \in [1, min(\#F, \#B)] \tag{5.7}$$

where # denotes the cardinality of a set.

Ordinary dilation is obtained as a special case of generalized dilation when $s = 1$.

The **generalized erosion** of F by B with strictness s is defined by:

$$F \overset{s}{\ominus} B = \{x : \#(F^c \bigcap B) < s\}; s \in [1, \#B] \tag{5.8}$$

where it is assumed that $\#F < \infty$.

Similarly, the ordinary erosion is also obtained as a special case of generalized erosion when $s = 1$.

The properties of generalized morphological operators can be found in [1].

It is important to note that, using the generalized operators in existing algorithms with strictness greater than one may increase the resistivity of the algorithms to noise and small intrusions.

5.3 Description of Structuring Elements Using Euclidean Distance Transform

Symmetrical and circular SEs play a fairly central role in mathematical morphology in the continuous plane, since they provide an isotropic treatment of the image. In the continuous domain, we use B_d to denote a circular SE whose radius is d, (see Figure 5.3). It is defined as:

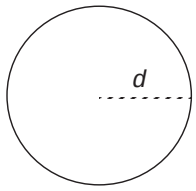


Figure 5.3: Structuring element defined by distance value in continuous domain.

$$B_d = \{b : d(b, 0) \leq d\} \tag{5.9}$$

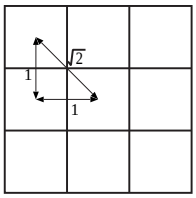
Where $d(b, F)$ is the distance from SE center point $b(i, j)$ to the nearest pixel belonging to F. Then, erosion and dilation by B_d can also be expressed as the threshold of a distance value.

$$\begin{aligned} F \ominus B &= \bigcap\{F - b : d(b, F) \leq d\} \\ F \oplus B &= \bigcup\{F + b : d(b, F) \leq d\} \end{aligned} \tag{5.10}$$

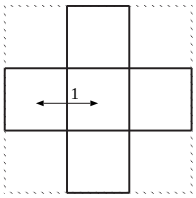
The above equations show that morphological operators only deal with the pixels whose distance values are not greater than d.

For digital images, however, circular SEs are rarely used because there is no "real" circular SE on a discrete lattice. The straightforward method to describe an SE is to

pick up all pixels around the center point $b(i, j)$. However, this method has a drawback; whenever the center moves to the position $b(i', j')$, we have to update all other pixels in B. The process is naive and time consuming, and induce time complexity of the algorithm to $O(N^2 \times d^2)$ for an image of size $N \times N$ and a SE of radius d. Fortunately, we can employ Euclidean distance transform value to easily describe a "circular" SE in the discrete domain and further reduce time complexity of algorithm to $O(l \times d^2)$. O. Cuisenaire and B. Macq proposed a similar idea using local distance transformation [22]. 3×3 SEs and cross SEs are much in use. In the following discussion, they will be used as examples to explain this principle, (see Figure 5.4).



(a) 3×3 SE: $d = \sqrt{2}$



(b) Cross SE: $d = 1$

Figure 5.4: Use of Euclidean distance value to describe "circular" SEs.

In the discrete space, we assume each pixel is a unit square. For a 3×3 SE, the distance between a horizontal (or vertical) neighbor and the center is 1. Similarly, the diagonal neighbor is at distance $\sqrt{1^2 + 1^2} = \sqrt{2}$ from the center. This means every pixel whose distance from point $b(i, j)$ lies in $[1, \sqrt{2}]$ will be covered by a 3×3 SE centering on the point $b(i, j)$. Thus, we can easily denote 3×3 SE by $B_{d=\sqrt{2}}$ (Figure.5.4 (a)). In a corresponding way, since the cross SE only has horizontal and vertical neighbors, it can be denoted by $B_{d=1}$ (Figure.5.4 (b)). Next, the "circular" SEs denoted by Euclidean distance transform value are given.

$$\begin{cases} \text{SE with 5 pixels}, & B_{d=1} \\ \text{SE with 9 pixels}, & B_{d=\sqrt{2}} \\ \text{SE with 13 pixels}, & B_{d=2} \\ \text{SE with 21 pixels}, & B_{d=\sqrt{5}} \\ \text{SE with 25 pixels}, & B_{d=\sqrt{8}} \\ \text{SE with 29 pixels}, & B_{d=3} \\ \vdots & \vdots \end{cases} \tag{5.11}$$

5.4 Eliminating Impulsive Noise using Linear Time GMO

In practice, due to factors like skin conditions (e.g. wet or dry), sensor noise, or incorrect finger pressure, fingerprint images obtained from sensors often contain a lot of noise, which heavily affects the accuracy of further processing. The noises of fingerprint image have many shapes and directions. Thus, a fine selection of the directional ordinary morphological operators is required; a large morphological kernel may be also required.

In such cases, the effects of denoising may damage the expected results due to extreme strictness of the ordinary morphological operators. So, we employ GMO in our system.

5.4.1 Advantages of GMO using Distance Transform

First, the GMOs have controllable strictness, and thus excessive erosion and dilation can be prevented. On the other hand, by controlling the strictness of a GMO, the GMO can adapt itself to the orientation and shape of fingerprint without adopting many directional operators.

Second, according to the properties of morphological operators, only the edge of a set F needs to be considered for computing these morphological operations. More accurately, equation 5.10 can be modified as follows,

$$\begin{aligned} F \ominus B &= F \bigcap (\partial(F) \ominus B) \\ F \oplus B &= F \bigcup (\partial(F) \oplus B) \end{aligned} \tag{5.12}$$

where $\partial(F)$ is the edge of F, that is the set of pixels of F with at least a direct neighbor not belonging to F and the distance transform value not greater than d. Assume l is the length of the contour of F (l is the cardinality of the set $\partial(F)$), then the computational complexity is reduced from $O(N^2 \times d^2)$ to $O(l \times d^2)$ for any SE of radius d. If the strictness of the GMOs is greater than 1, operators should count the number of intersected pixels between B and F^c (or $\breve{B}$ and F). Naive methods are dependent on the size of SEs. So, $O(l \times d^2) + N^2$ time is needed to check if the pixels belong to $\partial(F)$ by distance transform values.

Note that we usually use rectangular SE to do denoising (e.g. 3×3 SE, 2×3 SE, or 3×2 SE, etc.). Our method can further reduce $O(l \times d^2)$ to linear time by using integral image method.

5.4.2 Reducing Time Complexity by Integral Image

Integral Image was first used by Viola and Jones [87]. It is very similar to the summed area table used in computer graphics for text mapping [20]. The integral image can be computed from an image using a few operations per pixel. Once computed, any rectangular SEs can be computed at any scale or location in *constant* time.

The integral image at location (x, y) contains the sum of the pixels above and to the left of (x, y), inclusive:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{5.13}$$

where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image. Using the following pair of recurrences:

$$\begin{aligned} s(x, y) &= s(x, y - 1) + i(x, y) \\ ii(x, y) &= ii(x - 1, y) + s(x, y) \end{aligned}$$

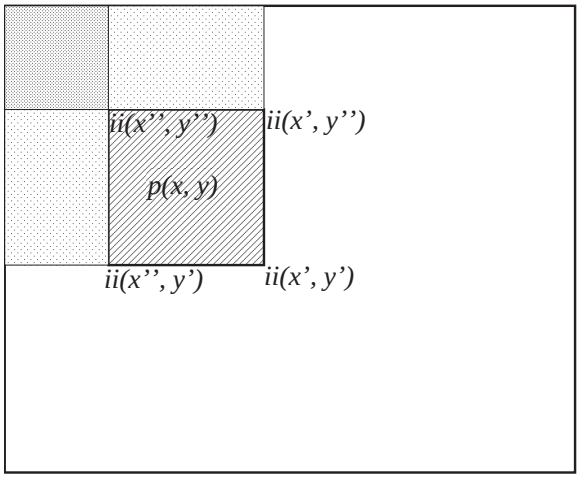


Figure 5.5: Count inverse pixels of center using integral image.

where $s(x, y)$ is the cumulative row sum, the integral image can be computed in one pass over the original image. Then any rectangle sum can be computed in four array references.

Owing to GMO's strictness $s > 1$, during morphological operations, operators should know the number of intersected pixels between B and F^c (or $\breve{B}$ and F). Our strategy is to first represent the original binary image using an integral image (see Figure **??**). When SE scans the point $p(x, y)$, we do the following operation.

$$n = ii(x', y') - ii(x'', y') - ii(x', y'') + ii(x'', y'')$$

where n is the number of background pixels in SE. For erosion, the state of the point $p(x, y)$ is converted if and only if $n \geq s$. It is similar to dilation.

From above, we get the following conclusion:

$$\begin{cases} O(N^2) & \textit{if rectangular SEs were used.} \\ O(N^2 \times d) & \textit{if isotropic SEs were used.} \end{cases} \tag{5.14}$$

In our method we employ rectangular SEs. So, the complexity of eliminating impulsive noise is **linear**.

5.4.3 Algorithm and Results

As the impulsive noise is fairly small and thin (specifically, one or two pixels wide in our case). And according to the property of GMO [1], strictness s must lie in interval $[2, \lfloor \#B/2 \rfloor]$. For a small SE, the integer midpoint $\lfloor \frac{2+\lfloor \#B/2 \rfloor}{2} \rfloor$ of the interval is reasonable strictness value. Therefore, a square SE of 3×3 pixels (corresponding to $d = \sqrt{2}$) with strictness of value 3 is used. The processing steps of this phase are as follows:

Input: A binary fingerprint image I of size $N \times N$ with impulsive noise. $p(i, j) \in I$.
Output: A binary image without impulsive noise.

1. Implement Euclidean distance transform for foreground pixels (F);
2. Represent binary image by integral image;
3. **IF** $p(i,j) = 1 or \sqrt{2}$
   $F \overset{3}{\circ} B_{d=\sqrt{2}}$;
   End.

It is known that computing times of Euclidean distance transform and integral image are both linear. In step 3, finding available pixels takes exactly N^2 time. As analyzed in the previous subsection, generalized opening using integral image can be done in $O(l)$ time. Therefore, the total time complexity is linear.

We tested 60 fingerprint images scanned from FUJITSU Fingerprint Sensor (model: FS-210u). These images are of size 300×300 and 500 dpi resolution. To make comparisons with our result easier, we list the average real computing times of these images in Table 5.1 and show result images in Figure 5.6, where OMO and GMO denote the methods based on ordinary and generalized morphological operators respectively. O. MO (Ratha) and O. MO (Wahab) are approaches proposed by Ratha et al [69] and Wahab et al [88] respectively. OMO results show broken ridges due to its excessive operation. For the Ratha, Chen, and Jain results [69], most impulsive noise has been removed, but computing time is longer than others, due to the need of finding the orientation of ridges. The Wahab, Chin, and Tan results [88] are fast but a little noise remains since only a few templates are employed. The approach of Ikeda et al. [45] needs special hardware, so we could not simulate it. Therefore, it is not shown in the Table 5.1 or Figure 5.6. From their paper, however, we learn that their approach does not remove the noise on the boundary of fingerprint ridges. In our results, the image is fairly clean and the fingerprint is less affected, because a GMO with strictness can adapt itself to the orientation and shape of fingerprint without adopting many directional operators. Moreover, another advantage of using GMO is that it needn't apply a closing to refill, as happens with the ordinary morphological operation. Thus, with GMO it is possible to carry out less operations. However, some useless components remain.

Method	Time (sec)
OMO	0.059
GMO	0.087
Ratha[69]	0.323
Wahab[88]	0.088

Table 5.1: The average real computing time among four methods to eliminate impulsive noise.

5.5 Automatically Choosing Appropriately-Sized SEs to Eliminate Useless Components

Owing to skin condition (e.g., wet) or incorrect finger pressure, some fingerprint images contain useless components, which are often mistaken for the terminations; this makes

(a) Binary images with noise.

it very difficult to correctly identify the minutiae relationships of a person's fingerprint image. Thus, AFIS recognize the fingerprint with useless components as a distinct print.

Definition 3 Useless Component *is an object disjoint from other objects and whose largest width is less than the mean width of fingerprint ridges.*

In order to eliminate useless components, a good SE is necessary. It is well known that the performance of mathematical morphology is heavily dependent on structuring elements (SEs). SEs have two aspects, size and shape. For denoising applications, the size of an SE is enlarged if membership values of the local area are uniform. If membership values of a local region are dispersed compared to other regions, minutiae may exist in that region. In this case, the SE must be small. For enhancement or feature extraction, in which expected objects either are distributed in the background with masses of other objects or are almost blended with background, the shape of the SE should approximate the target as closely as possible. So far, how to choose an optimal or adaptive SE is still a hot topic.

There are several methods to find optimal or adaptive SEs, as described in the introduction. None of them, however, start with a definition of appropriate SEs. In contrast, we define a condition for eliminating useless components based on the width of the useless component, which must be less than the average width of finger ridges. We also employ distance transform and show how it can be used as a measure for width and then present our algorithm to efficiently determine the size of SEs.

5.5.1 Distance Transform and Ridge Width

Fingerprint images are characterized by almost equal width ridges (a small part of the image is shown in Figure 5.7). We will use this particular characteristic of fingerprint to estimate the size of SEs. Measuring the width for arbitrary shapes, however, is a difficult, non-trivial task. For easy understanding, we first model this problem in continuous domain to show how distance transform can be used to estimate the widths of ridges and useless components in fingerprint images and then, generalize it to the discrete domain.

(b) Using OMOs.

(c) Using Ratha et al.'s approach.

(d) Using Wahab et al.'s approach.

(e) Using GMO by integral image.

Figure 5.6: Results obtained by various methods.

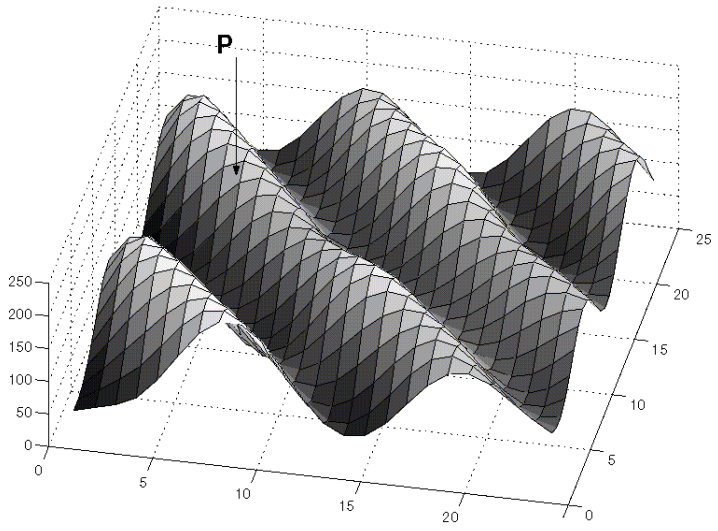
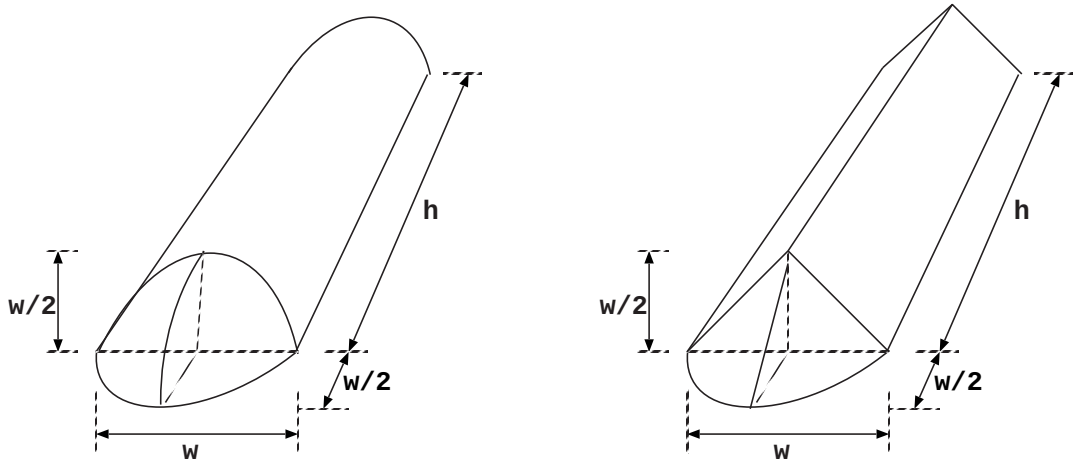


Figure 5.7: Magnified view of a part of the gray scale topology of a fingerprint image.

Model in the Continuous Domain

The fingerprint ridge can be modelled in 3D domain as shown in Figure 5.8(a). In the continuous domain, the image is a continuous function $f : (x, y) \rightarrow I\!R$. This function increases along a direction perpendicular to the ridge till it reaches the ridge top point, then decreases till it reaches the bottom. Since distance transform values are used to approximate to gray scale in our method, we obtain a distance transform 3D model as shown in Figure 5.8(b).



(a) 3D model of real ridge.

(b) Distance transform 3D model.

Figure 5.8: Diagrams of the 3D model.

After projecting distance transform 3D model onto a plane, we get the planar diagram shown in Figure 5.9. We compute the total distance transform value of this shape. Consider this special geometric object of width w and height h, with $h \gg w$. The medial axis of this object is given by the dot line segment $\overline{AB}$, $\overline{CD}$ and $\overline{EF}$. These dot line segments divide the geometric shape into four regions such that the nearest boundary line from any

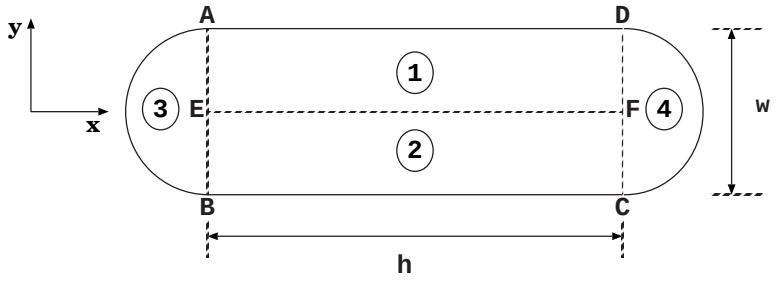


Figure 5.9: Diagram for computing total distance value.

point in the region is determined. For instance, the region 1 has $\overline{AD}$ as its nearest boundary line and region 3 has $\widehat{AB}$ as its nearest boundary arc. The total distance transform value for region 1 is $\int_0^{\frac{w}{2}} \int_0^h (\frac{w}{2} - y)\ dxdy = \frac{1}{8}w^2h$. Similarly, the total distance transform value for region 3 is $\int_{\frac{\pi}{2}}^{\frac{3}{2}\pi} \int_0^{\frac{w}{2}} (\frac{w}{2} - r)r\ drd\theta = \frac{1}{48}\pi w^3$. So, the total distance transform value is:

$$\begin{aligned} \phi_{dt}(w) &= \frac{1}{8}w^2h \times 2 + \frac{1}{48}\pi w^3 \times 2 \\ &= \frac{1}{4}w^2h + \frac{1}{24}\pi w^3 \\ &= \frac{1}{4}w^2(h + \frac{1}{6}\pi w) \end{aligned} \tag{5.15}$$

From this result, we can find some relationship between the distance transform value and the width of the ridge. The length of the ridge, however, is fairly difficult to measure. Given this difficulty, it is reasonable to use the average distance transform value to measure ridge width. We first compute the area of the geometric object. The areas for region 1 and 3 are $\frac{1}{2}w \times h$, and $\pi(\frac{1}{2}w)^2 \times \frac{1}{2}$ respectively. So, the total area is:

$$\begin{aligned} \phi_S(w) &= \frac{1}{2}w \times h \times 2 + \pi(\frac{1}{2}w)^2 \times \frac{1}{2} \times 2 \\ &= wh + \frac{1}{4}\pi w^2 \\ &= w(h + \frac{1}{4}\pi w) \end{aligned} \tag{5.16}$$

The average distance transform value is equal to total distance transform value (equation 5.15) divided by area (equation 5.16). So,

$$\begin{aligned} E_{dt}(w) &= \frac{\frac{1}{4}w^2(h + \frac{1}{6}\pi w)}{w(h + \frac{1}{4}\pi w)} \\ &= \frac{1}{4}w\frac{h + \frac{1}{6}\pi w}{h + \frac{1}{4}\pi w} \\ &< \frac{1}{4}w \\ &\approx \frac{1}{4}w \qquad if \quad h \gg w \end{aligned} \tag{5.17}$$

In the definition of a fingerprint 3D model, the length is much greater than width. It implies that the average distance transform value can be approximated to one-fourth of the ridge width with a negligible error. So four times the average distance transform value is less than but almost equal to the average ridge width. Useless components are objects disjoint from other objects and their largest width is less than the mean width of the fingerprint ridges. This shows that the average distance transform value can be used as a good measure of estimating appropriate SEs. Our goal is to automatically determine appropriate "circular" SEs for eliminating useless components. The criteria is given by the definition of useless component (see Definition 1). So, more formally, we have the following lemma.

Lemma 2 *When an isotropic SE satisfies the following condition:*

$$max(w_{useless}) \leq 2d \leq mean(w_{ridge})$$

the useless components can be eliminated, but some eroded ridges of the fingerprint shall remain.

PROOF. According to the erosion of ordinary morphology, eroding an object can be found by intersecting all translates of the object by the reflection of the SE. As in our previous assumption, the diameter of isotropic SE is not less than the maximum width of the useless component. Then the object translated by the reflection of SE is too far to intersect any other translated object, i.e. $\bigcap\{F - b : b \exists B\} = \phi$. Thus, the useless components can be eliminated under such condition. The diameter $2d$, however, is not greater than the mean width of fingerprint. That means some wider portions, which are wider than SE, are not translated so far away that they can intersect each other. So, some of skeleton of the fingerprint remains.

Discrete Image and Distance Transform

In the discrete model, the co-ordinates are discrete given from the pixel locations. So, the observations from the previous subsection do not directly apply. But, the crucial observation from the previous subsection is that the average distance transform value approximates to one-fourth the width of the ridge. Then, an appropriate SE B can be obtained from the observation: Radius of the SE is less than but approximates to half the average width of the ridges. This observation still works in the discrete domain. However, the circular SE can move in an arbitrary direction in continuous domain. On the contrary, a "circular" SE only can move straight (in a horizontal or vertical direction) or diagonally with one step in the discrete domain. In this case, the SE radius must be remeasured by the same method as that used to measure ridge width instead of the simple radius d. We also need to consider the equal probability of moving straight or diagonally. Thus, the radius r of SE is the mean of radiuses in these two cases. Then, we have the following Definition:

Definition 4 *An appropriate SE can be determined when half the average width of ridges is located in the corresponding interval.*

$$\begin{cases} 2 \leq 2E_{dt}(w) < 2.828, & B_{d=1} \\ 2.828 \leq 2E_{dt}(w) < 3, & B_{d=\sqrt{2}} \\ 3 \leq 2E_{dt}(w) < 3.606, & B_{d=2} \\ 3.606 \leq 2E_{dt}(w) < 4, & B_{d=\sqrt{5}} \\ 4 \leq 2E_{dt}(w) < 4.243, & B_{d=\sqrt{8}} \\ 4.243 \leq 2E_{dt}(w) < 4.472, & B_{d=3} \\ \vdots & \vdots \end{cases} \tag{5.18}$$

where $E_{dt}(w)$ is the average distance transform value. B_d is a circular SE with radius d.

With this definition in place, we are in a position to design an algorithm in the next section.

5.5.2 Algorithm and Results

Our strategy is to take information from an original image I with useless components and an eroded image I_e by using the SE that satisfies Definition 4, in which useless components are eliminated completely but fingerprints cannot totally be deleted although they are affected to some degree. Then, we integrate fingerprint images I and I_e to restore the expected image I_r without useless components. More precisely, we establish a correspondence between the two images. The objects which do not correspond to any information in the eroded image I_e, are classified as useless components. We then eliminate them. The remaining objects, which have correspondences, are classified as fingerprint and remain in the final output image I_r.

To take care of non uniform ridge width across different image regions, we cut out sub-block of image region and pick up the minimal average distance transform value as criteria.

Input: A binary fingerprint image I of size $N \times N$ with useless components.

Output: A binary image without useless components.

1. Apply Euclidean distance transform on entire image;
2. **do for all** sub-block of the image I;
   Compute average distance transform value $E_{dt}(w_i)$;
3. Pick up $min(E_{dt}(w_i))$, then select appropriate SE B by definition 1;
4. Erode I by B;
5. Restore I_r by integrating I and I_e.

The detail of step 5 is described as follows:

Input: A binary image I with useless components and eroded image I'.
$p(i,j) \in I; p'(i,j) \in I'$.

Output: Restored image I_r.

Q is a queue.
BEGIN
For i,j = 1 to n
  If $p(i,j) \in F$ and not visited
    Push($p(i,j)$), count = 0;
Mark useless components.
    While Q is not empty
      Pop($p(i,j)$), label $p(i,j)$ visited;
      If any neighbor of $p(i,j)$ is in F and not visited
        Push it into Q and label it visited;
      If $p'(i,j) \in F$ then count++;
Eliminate useless component.
    If count == 0
      Push($p(i,j)$);
      While Q is not empty
        Pop($p(i,j)$), let $p(i,j) \in F^c$;
        If any neighbor of $p(i,j)$ is in F
          Push it into Q;
END.

Time complexity of eliminating useless components algorithm is analyzed as follows:

- Because $p(i,j)$ is pushed and popped only once, Step 1 takes n^2 time to mark useless components.
- Step 2 is similar to step 1. Thus, it takes n^2 time for eliminating useless components.

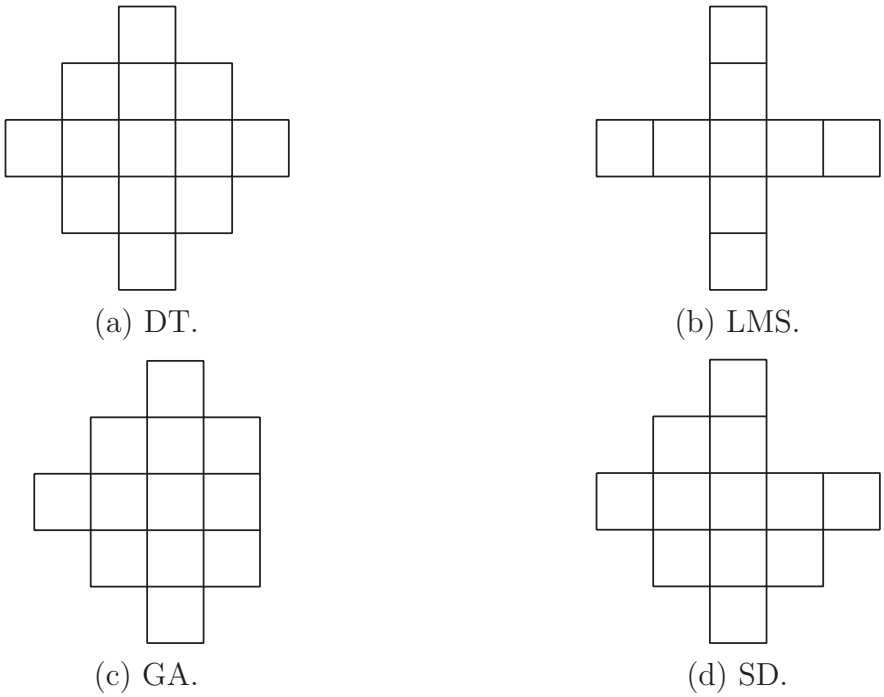
It is clear that time complexity is $O(n^2)$.

In this stage, we tested the same set of 27 images containing useless components as that used for the previous denoising stage (see Section 4.3). We also list average real computing time in Table 5.2 and show the SEs chosen by different methods in Figure 5.10. The final results processed by other methods are almost the same, so Figure 5.11 only shows our result, where DT, LMS, GA and SD denote the methods based on distance transform, the least mean square, genetic algorithm and standard deviation, respectively. Time means the CPU time. Error means the number of useless components not eliminated.

Method	Time (sec)	Error (num)
DT	0.063	0
LMS [29, 30, 31]	0.34	1
GA [2, 59, 58]	1.02	0
SD [52]	0.374	0

Table 5.2: The average real computing time among four methods to eliminate useless components.

The results show that our method (DT) is much faster than other three and works well. The least mean square method needs more time due to its training phase, and



(a) DT. (b) LMS.

(c) GA. (d) SD.

Figure 5.10: SEs chosen by different methods.

the chosen SE is not big enough, so that a few useless components remain. The genetic algorithm method takes the longest time, although it does find the optimal SE. But this computing time is unacceptable for a real time system. The standard deviation method can find an appropriate SE and run as fast as LMS. However, the length of the linear SE is determined manually, rather than automatically. Our method avoids these drawbacks. It runs fast and produces satisfactory results. Thus, it can be used with significant gains for fingerprint recognition in realtime applications.

5.6 Conclusion

We have developed a combinatorial linear time algorithm to eliminate impulsive noise and useless components from fingerprint images using Euclidean distance transform. There are two contributions. The first one represents binary fingerprint images and SEs by integral image and distance transform values, and reduce the time complexity of GMO from $O(N^2 \times d^2)$ to $O(N^2)$. The results are fairly clean and the fingerprint shapes are less affected. The second contribution is an algorithm for automatically determining an appropriate "circular" SE to eliminate useless components from fingerprint images exploiting the average fingerprint ridge width. We used Euclidean distance transform as a measure of width for determining the radius of SEs. One of the advantages of our method does not require parameters determined using experiences. Ordinary erosion with an appropriate SE only needs $O(l) + N^2$ time to eliminate useless components. So, the entire algorithm for fingerprint denoising is linear time.

(a) Images with useless components.

(b) Eroded images.

(c) Images without useless components.

Figure 5.11: Results of eliminating useless components.

Chapter 6

Minutiae Detection

6.1 Introduction

Most automatic systems for fingerprint comparison are based on minutiae matching; hence, reliable minutiae extraction is an extremely important task and a lot of research has been devoted to this topic. Most of the proposed methods require the fingerprint gray-scale image to be converted into a binary image, and then submit the binary image to a thinning stage which allows for the ridge line thickness to be reduced to one pixel (see Figure 6.1 for illustration).

6.1.1 Thinning Based Methods

A binary thinned fingerprint image is also denoted as *skeletons* of ridges. The skeleton of a binary object in a plane is a tree-form curve. A skeleton is called the medial axis, or more generally, the medial set of an object [66, 67, 83]. The skeleton of an object is a fundamental geometric feature for image and shape analysis. Therefore, skeletonisation has been studied in the field of pattern recognition and computer vision for a long time.

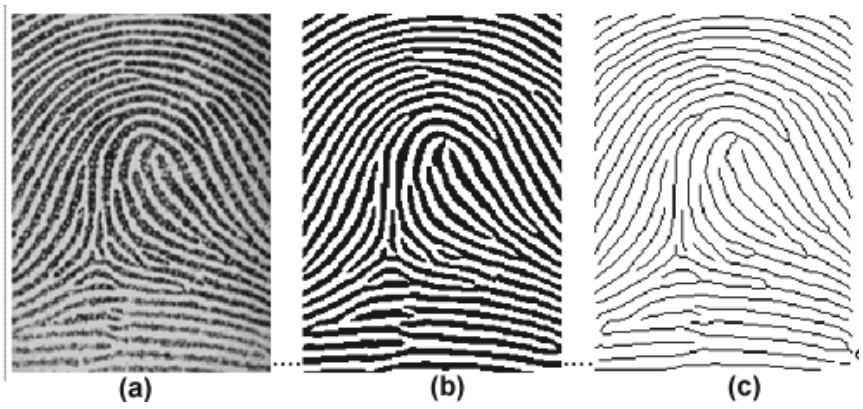


Figure 6.1: (a) shows a fingerprint gray scale image; (b) shows the image obtained after a binarization of the image in (a); (c) shows the image obtained after a thinning process of the image in (b).

Once a binary skeleton has been obtained, a simple image scan allows the pixel corresponding to minutiae to be detected: in fact the pixels corresponding to minutiae are characterized by a *crossing number* different from 2. The crossing number $cn(p)$ of a pixel p in a binary image is defined as half the sum of the differences between pairs of adjacent pixels in the 8 neighborhood of p [4]:

$$cn(p) = \frac{1}{2} \sum_{i=1...8} |val(p_{imod8}) - val(p_{i-1})| \tag{6.1}$$

where $p_0, p_1, \ldots, p_7$ are the pixels belonging to an ordered sequence of pixels defining the 8 neighborhood of p and $val(p) \in \{0, 1\}$ is the pixel value. It is simple to note that a pixel p with $val(p) = 1$:

- is an intermediate ridge point if $cn(p) = 2$;
- corresponds to termination minutia if $cn(p) = 1$;
- defines a more complex minutia (bifurcation, crossover, ect.) if $cn(p) \geq 3$

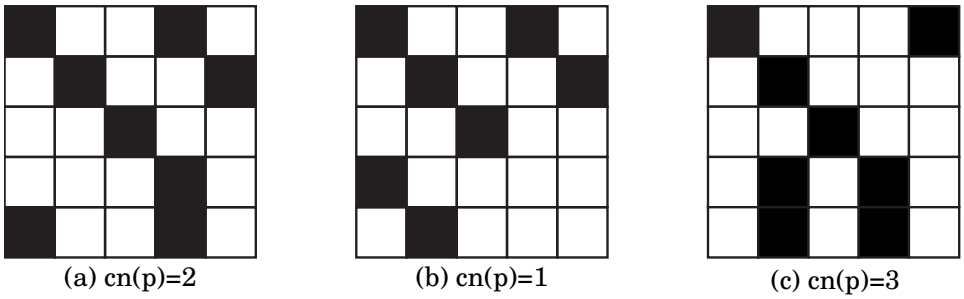


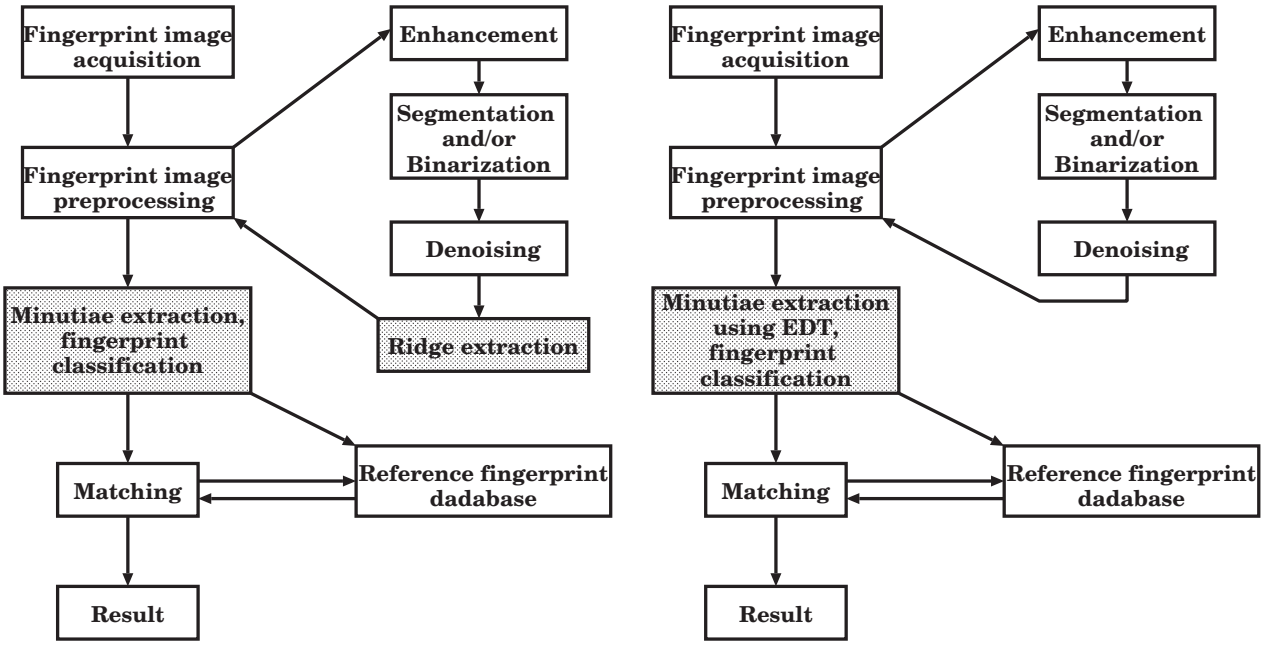
Figure 6.2: (a) intra-ridge pixel; (b) termination minutia; (c) bifurcation minutia.

Classical thinning algorithm for planar objects based on the discrete transform usually transforms T and V shape junctions to Y shape junctions. These changes of junctions affect the final form of thinning process yielding unexpected needles and branches which are not in the original forms. Hilditch's thinning is an algorithm that does not produce unexpected needles and branches. The method contains processes based on the configuration in the neighborhood of each point [38]. A. Imiya, M. Saito and K. Nakamura proposed a thinning algorithm based on curvature flow in a space is defined using configurations of vertices on an isostatic polyhedron derived form the 6-connected boundary. It also removes needles and branches. However, all of these works need more computing time to improve skeleton quality for further matching stage. Therefore, thinning for fingerprint minutiae extraction has two drawbacks: (i) it may introduce a large number of spurious minutiae; (ii) it is time consuming.

6.1.2 Our Work

To reduce preprocessing time of AFIS, we proposed a novel method which extracts minutiae directly from binary image. Most of the minutiae detection algorithms find out the

minutiae from the one-pixel thick ridge image obtained by thinning after binarization (see, for example, [28]). Once our binarization process finishes, we have a matrix with each ridge pixel having a corresponding EDT value. Our goal is now to exploit this information to detect minutiae directly from the thick ridges of the binary image without thinning (see Figure 6.3 for illustration). This will be useful for saving time in real world application. In most cases the location of the minutiae will not be exact the same in different versions of the same fingerprint owing to distortion, so it makes sense to find out a small region of connected pixels in which the minutiae lies; in other words, this region is a set of connected pixels. Our goal, then, is to find these minutiae regions for further matching. The minutiae treated will be bifurcation and termination.



(a) Usual approaches. (b) Our approach.

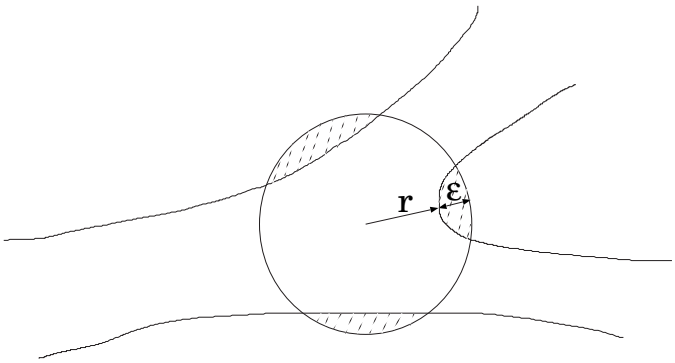
Figure 6.3: Two flowcharts showing the different phases in fingerprint analysis taken by (a) usual approaches and (b) our method. In (b) the highlighted modules show the area of our work.

6.2 Detecting Bifurcation Minutiae

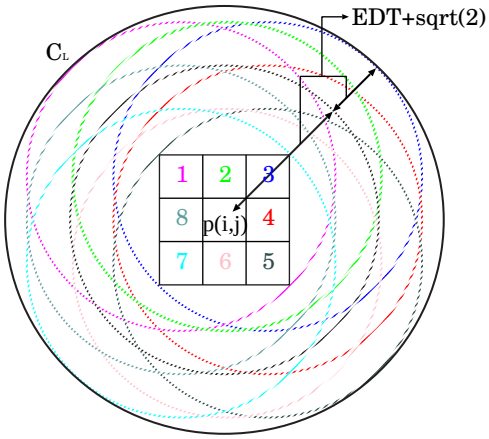
6.2.1 Bifurcation Minutiae in the Continuous Domain

We first show how EDT can be used for finding a bifurcation minutia in the continuous domain in the following observation.

Observation 1 *If we draw a circle C centered at a bifurcation point in the continuous domain with a radius $r + \epsilon$ (where r is the EDT value in the continuous domain and ϵ is an arbitrarily small positive number), then C must contain three background components. See Figure 6.4(a) for an illustration.*



(a) Bifurcation in the continuous domain; the background components are shown shaded.



(b) Circles formed by the EDT value; the circle C_L is the smallest radius circle covering all the 9 EDT circles (shown dotted).

Figure 6.4: Finding bifurcation minutiae.

Loosely speaking, this bifurcation point corresponds to the bifurcation in the medial axis of the continuous structure. Next, follows a discussion on carrying forward Observation 1 in the continuous domain to the discrete domain.

6.2.2 Bifurcation Minutiae in the Discrete Domain

The issue here is to find out an appropriate radius for a circle so that Observation 1 holds in the discrete domain.

Definition 5 *An EDT circle is a circle centered at a grid point (i, j) with a radius r, where r is the EDT value of the grid point (i, j).*

EDT in the discrete domain has a geometric significance as mentioned in the following fact.

Fact 2 *For a ridge pixel p_i having an EDT value r, if we draw a circle C centered at p_i with a radius r, then there will be at least one valley pixel on the circumference of C.*

The radii of two EDT circles whose centers lie in the 8-neighborhood of each other cannot be arbitrary. The following Observation is easy to follow.

Observation 2 *Let the EDT value at a grid point (i, j) and any of its 8-neighbors be r and $r_i, (i = 1, \ldots, 8)$ respectively. Then, surely $r - \sqrt{2} \leq r_i \leq r + \sqrt{2}$.*

The equality is attained when (i, j), any r_i and the pixel that contributes to the EDT value at (i, j) lie on a straight line.

Unlike in a continuous domain, where a point can be chosen as mentioned in Observation 1, it is very difficult to choose a unique pixel as a bifurcation in a discrete digital image. This happens because the medial axis is not uniquely defined for a digital image. So, we mark a set of 9 connected pixels as a region where a bifurcation lies.

Observation 3 *If there is a bifurcation region in a 3×3 window of pixels, there will be three background components of connected pixels inside the union of those 9 EDT circles. See Figure 6.4(b) for the union of the 9 EDT circles.*

This observation is true because in whatever way a bifurcation enters or leaves the 3×3 region, it divides the 3×3 region into three regions and in each region there will be at least one pixel; and the *EDT* circle centered at this pixel will have three different background components. But, finding out the correct union of these 9 circles is somewhat time consuming. So, we will find out a circle C_L with the least radius centered at (i, j) that covers the union of the 9 *EDT* circles centered at (i, j) and its 8 neighbors.

Lemma 3 *The radius of the circle C_L is given by the EDT value plus $\sqrt{2}$.*

The proof follows from Fact 2, Observations 2 and 3.

So, to test for a bifurcation region, we form the circle C_L; and if in this circle there are three separate background components, we mark the pixels in this region as bifurcation.

6.2.3 Avoiding Spurs and Bridges to Find Bifurcation Minutiae

The theory developed earlier for detecting bifurcation fails when there are spurs or bridges, which are deformities in the ridge structure. A *spur* [62] is a small protrusion from the original ridge and gives rise to a false minutia. A *bridge* [62] is a structure that joins two ridges across a valley and gives rise to two false minutiae. See Figure 6.6(a) and (b) for examples. To avoid the false minutiae thus generated, we developed the following heuristic. Note that, in the case of spurs, the circle C_L is such that it will cut the spur and generate a bifurcation. To avoid that, we increase the radius of C_L, to find only two connected components and avoid this false bifurcation. In case of bridges, if the radius of C_L is increased appropriately, then both the false minutiae can be removed by detecting the intersection of two such circles. In massive experiments, statistics show the average width of ridges (valleys) is around $5 \sim 6$ pixels wide in 300×300, 500 dpi image (see Figure 6.5). Note that, from our binarization algorithm, we can estimate the width as average of the EDT values. So, we increase the radius of C_L and make the new radius as $EDT + 3$ $(3 = \lceil (5/2) \rceil = \lceil (6/2) \rceil)$. We now apply the same method of the previous

subsection with a new circle C'_L whose radius is $EDT + 3$. The only modification is that we detect the intersection of C'_Ls to avoid bridges. See Figure 6.6(a) and (b) for an illustration of spur and bridge removal respectively.

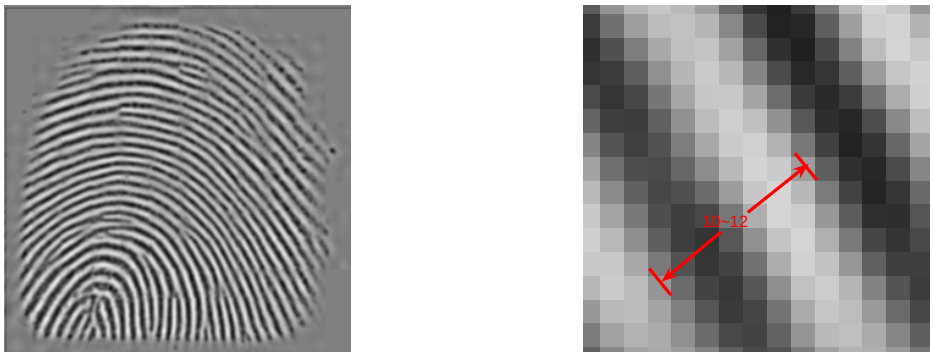
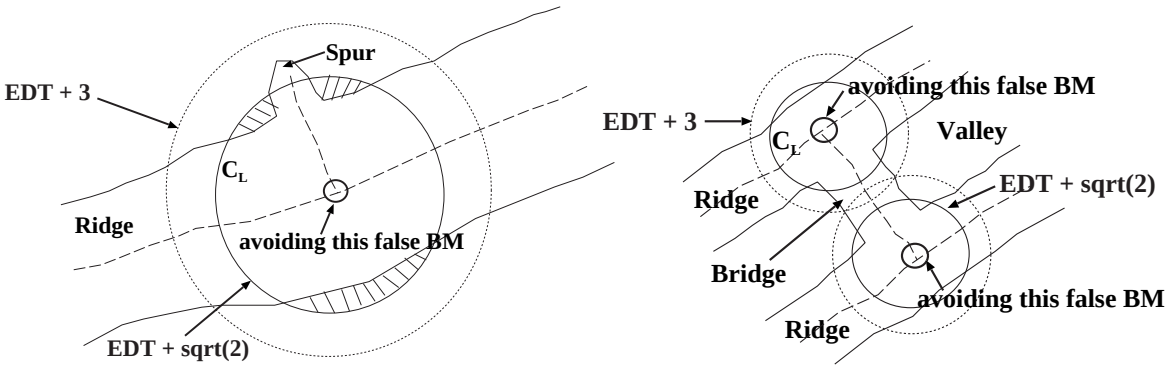


Figure 6.5: The average width of ridges (valleys).



(a) Avoiding detection of spurs as bifurcation.

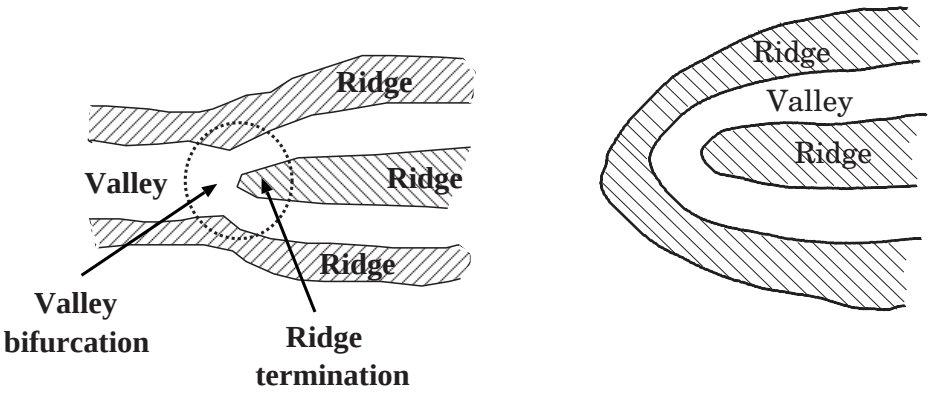
(b) Avoiding detection of bridges as bifurcation.

Figure 6.6: Finding correct bifurcation minutiae(BM) by avoiding spurs and bridges.

6.3 Detecting Termination Minutiae

There is a widely held belief that a ridge termination minutiae is a dual of a valley bifurcation minutiae as pointed out in [62] (see Pg. 85-86 of [62] and Figure 6.7(a)). With this duality principle in place, we thought that the methodology developed for detecting ridge bifurcation could simply be applied to detect valley bifurcation and hence, ridge termination. But, during our experiments on different data sets, we came across a counter example to this duality principle as shown in Figure 6.7(b), where a ridge termination does not correspond to a valley bifurcation. So, we developed the following heuristic method to detect such minutiae. The heuristic method can to some extent differentiate between spurs [62] and real termination minutiae.

Refer to Figure 6.8 for the following part of the discussion. We use the circle C'_L defined for determining bifurcation minutiae to detect the termination minutiae.



(a) The duality relation between ridge termination and valley bifurcation.

(b) A counterexample where duality fails.

Figure 6.7: Misconception of duality relation between ridge termination and valley bifurcation.

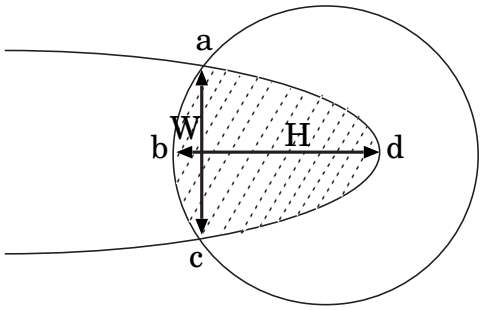


Figure 6.8: Finding termination minutiae.

Observation 4 *A ridge termination minutiae has the following two properties: (i) it is surrounded by a single component of valley pixels inside the union circle and (ii) the length of the elongated part of the ridge termination (estimated by $\overline{bd}$) is greater than the width of the ridge (estimated by $\overline{ac}$) inside the union circle.*

Based on the above observation, our heuristic method for finding a termination is as follows:

Description 1

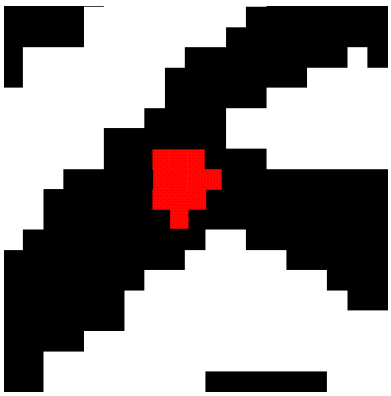
1. *Find the two intersection points of the ridge with the union circle (a and c), and compute the length W of the line segment $\overline{ac}$.*

2. *Find the middle point b of the arc ac of the union circle; and the farthest point d on the ridge boundary from b. Compute the length H of $\overline{bd}$.*

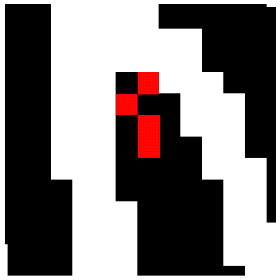
3. *If $H > W$, then the origin of the union circle is a termination minutiae.*

We could remove many spurs in our heuristic for finding terminations because spurs do not satisfy the structural information stated in Observation 4.

Figure 6.9 (a) and (b) are instances of extracted bifurcation and termination minutiae regions using our algorithm.



(a) A magnified view of a bifurcation minutiae region.



(b) A magnified view of a termination minutiae region.

Figure 6.9: A figure showing minutiae regions.

6.4 Minutiae Detection Results

The bifurcation and termination minutiae detected by our algorithm were compared manually with the minutiae present in the enhanced images and the results are given in Table 6.1. It can be seen that even for bad quality fingerprint images, our total error is within acceptable limits. Another interesting point to note is that there is no error for exchanging termination and bifurcation minutiae. Figure 6.10 shows the enhanced, binarized and minutiae detected images of of a representative image from each of the five data sets considered. The time taken for minutiae detection averages 0.177 sec. on an IBM x40, inter pentium M, WinXP, 1.4GHz.

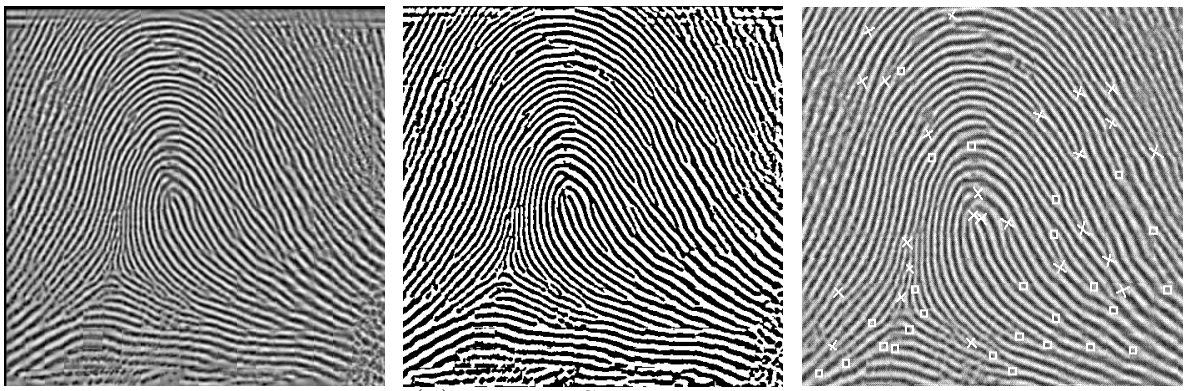
Table 6.1: Minutiae detection results; T: termination, B: bifurcation

Good Quality Fingerprint Image								
	Minutiae		Dropped		False		exchanged	total error
	T	B	T	B	T	B		
Total	48	31	2	0	0	3	0	
Percentage	79		2.53%		3.8%		0%	6.33%
Bad Quality Fingerprint Image								
	Minutiae		Dropped		False		exchanged	total error
	T	B	T	B	T	B		
Total	67	32	3	0	5	7	0	
Percentage	99		3.03%		12.12%		0%	15.15%

(a) Scanned by the FUJITSU Fingerprint Sensor (model: FS-210u).

(b) From FCV2000 database.

(c) From FCV2002 database.



(d) From NIST14 database.

(e) From NIST4 database.

Figure 6.10: Enhanced, binarized and minutiae detected images of a representative image from each of the 5 databases. The cross marks indicate terminations and the boxes indicate bifurcation.

6.5 Conclusion

In this chapter, we focussed on minutiae extraction of fingerprints using a Euclidean distance transform. The particular choice of EDT helped us in developing deterministic algorithms for binarization and denoising. We used the same EDT to find bifurcation and termination minutiae. Working with the original binary image (not the thinned one) has its added advantage: spurs and bridges can be removed in the same phase as minutiae detection. The most notable distinction of our work is that it uses fewest arbitrarily selected parameters.

Chapter 7

Fingerprint Distortion Correction

7.1 Introduction

There is a popular misconception that automatic fingerprint recognition is a fully solved problem, since it was one of the first applications of machine pattern recognition almost fifty years ago. On the contrary, a great number of challenging problems still exist. In particular, all current on-line fingerprint identification techniques require the matching algorithms to become more tolerant with respect to some factors which prevent the false rejection rate (FRR) from decreasing beyond a certain limit.

One of the main difficulties in matching two fingerprint samples of the same finger is to deal with non-linear *distortions*. Distortion arises from the elasticity of finger skin, the pressure and movement of fingers during image capture. It changes the spatial location of minutiae, and then leads to great difficulties in establishing a match among multiple images acquired from a single finger.

Many approaches have been proposed to cope with fingerprint distortion. They can be classified into several categories.

Dorai, Ratha and Bolle [25] proposed an automatic method for detecting the presence of distortion from compressed fingerprint videos of fingerprints, and then rejecting the distorted frames. Unfortunately, such kind of sensor is too expensive to be widely used. Another limitation is that once a print is acquired, nothing can be done about distortion in the data. Distorted prints in large legacy databases cannot benefit from this technique.

Conventional matching techniques use tolerance boxes. In order to tolerate minutiae pairs that are apart because of distortion, and to decrease the false rejection rate (FRR), the tolerance boxes can be enlarged. However, as a side effect, this gives non-matching minutiae pairs with a higher probability to be paired, resulting in a higher false acceptance rate (FAR). Jain et al. [47] and Luo, Tian and Wu [60] define the tolerance boxes in polar coordinates. In their methods, in order to compensate for the effect of distortion, the size of the tolerance boxes is incrementally increased moving from the center toward the borders of the fingerprint area. However, this still increased the probability of higher FAR.

Senior and Boole [77] proposed a canonical model. They believed the most evident effect of distortion is the local compression or stretching of the ridges and valleys. Their

model normalizes a fingerprint image to a canonical form by deriving a fingerprint representation in which all the ridges are equally spaced. This model can actually correct traction deformation very well, but torsion deformation cannot be adequately corrected.

Cappelli, Maio and Maltoni [14] explicitly modeled skin distortion caused by non-orthogonal pressure of the finger against the sensor surface. Their distortion model defines three distinct distortion regions according to different pressure. Experiments showed that this model provides an accurate description of the elastic distortion. However, the parameters of this model should be given by experiments, not automatically. So, this model is not yet used in AFIS.

A.M. Bazen and S.H. Gerez [5] and A. Ross, S.C. Dass and A.K. Jain [72] attempted to find a smoothed mapping between the input and template minutiae. Both of them used a thin-plate spline model to deal with the non-linear distortion. Through an iterative procedure, which starts from the possible matching pairs (minutiae or ridge curve), the minutiae in the input fingerprint are locally moved to better fit the template minutiae.

In this chapter, we propose a combined RBF model for correcting fingerprint elastic distortion (see Figure 7.1 for an illustration). The main contributions of this work have two aspects: first, lead RBF model into fingerprint distortion and find the best basis function and parameters for this problem; second, propose a combined RBF model which separately uses rigid and nonrigid transformations based on particularity of fingerprint distortion. Experiments with data of 20 real fingerprints show our combined RBF model can decrease a horrible distortion into around 30% comparing with a rigid transformation.

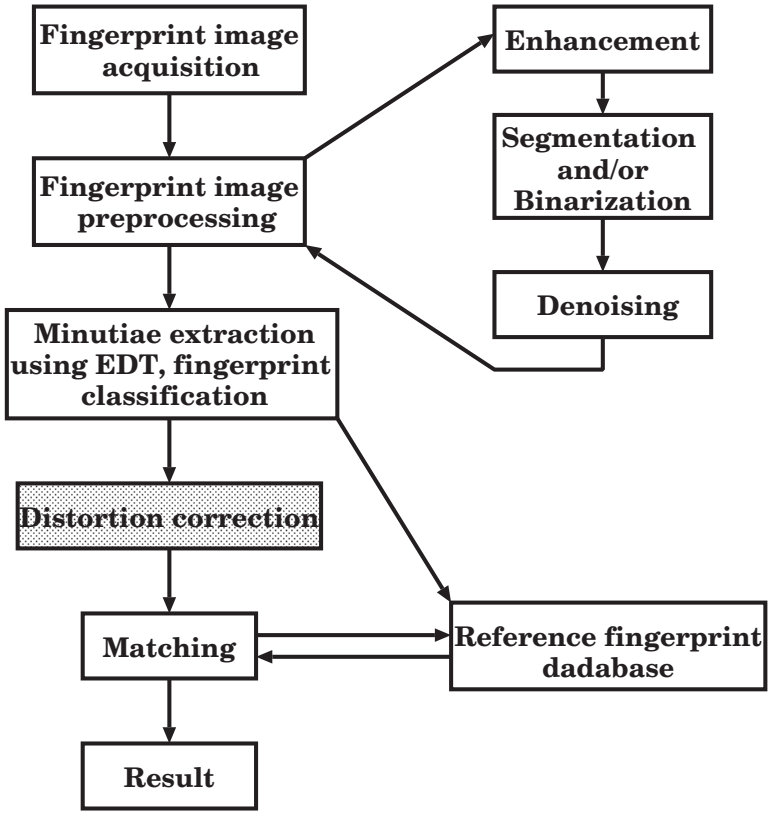


Figure 7.1: A flowchart showing different phases of fingerprint analysis. The highlighted block shows our work in this chapter.

In RBF model, control-points are required to construct the linear and nonlinear components. Process of finding these control-points between input and templet fingerprints is called *indexing* or *alignment*. So we first briefly review the technique of indexing we used in below.

7.2 Indexing Using Delaunay Triangulation

Indexing has received considerable attention in the literature [54, 12, 6] since it does not require considering each template separately, thus, it is less dependent on the database size. During indexing, features which remain unchanged under geometric transformations (invariants) are extracted from groups of template points and used to form indices. The indexed locations are filled with entries containing references to the templates. The templates listed in the indexed entries are collected into a list of candidate templates and the most often indexed templates are selected for further verification.

Although indexing is an attractive approach, very often it becomes less effective due to limited index selectivity. The heart of the problem is the low dimensionality of the invariants used to form the indices. In addition, indexing has high memory requirements. In the case of fingerprints, memory requirements can become much higher since fingerprints contain more features on the average than typical objects. Indexing usually consider every possible group of features (of a specific size) for building the table. There are two main reasons for this: first, it is desirable to build some degree of redundancy in the table so recognition can become more robust and second, there is usually no a-priori knowledge for choosing certain groups over others. Although redundancy can improve robustness, redundancy with limited index selectivity increase false positives, slowing recognition time significantly

One way to deal with the problem of limited index selectivity is by choosing larger size groups. However, this will further increase memory requirements since the number of groups increases exponentially with size [16]. To get around this problem, *grouping* has been suggested in object recognition to identify important groups of features only [46]. Using grouping in fingerprint recognition, however, will not be a good idea since the minutiae have a rather random distribution. Another idea to improve index selectivity is by adding new invariants to the index, thus, increasing its dimensionality. The FLASH algorithm is based on this idea [12]. In [35], the FLASH algorithm was used for fingerprint identification. FLASH considers triangles of minutiae to compute a 9-dimensional index which includes information about the lengths of the sides of the triangle formed by the triangle, the ridge count between each pair, and angle information. Although the idea of using high-dimensional invariants does improve index selectivity, new issues arise since we need to consider how the high-dimensional invariants will be computed fast and reliably.

We use a new indexing approach. Central to the new approach is the idea of associating a unique topological structure with the minutiae using *Delaunay triangulation*. The minutiae triangles of the Delaunay triangulation are then used for indexing. There are several advantages behind this idea. First of all, we only consider $O(N)$ triangles for indexing, implying lower memory requirements and less redundancy. Second, the minutiae triangles of the Delaunay triangulation have good discrimination power since, among all possible triangles, they are the only ones satisfying the properties of the Delaunay trian-

gulation. The improved index selectivity has less redundancy of the information. Finally, indexing can be implemented in a low-dimensional space

7.2.1 Delaunay Triangulation

Triangulation is a process that takes a region of space and divides it into subregions. The space may be of any dimension, however, a 2D space is considered here since we are dealing with 2D points (minutiae). In this case, the subregions are simply triangles. Triangulation has many applications in finite elements simulation, surface approximation and nearest neighbor identification [82, 7]. Here, however, our goal, is to associate a 2D topological structure with the minutiae.

The *Delaunay triangulation* for a set S of points in the plane is the triangulation $DT(S)$ of S such that no point in S is inside the circum-circle of any triangle in $DT(S)$. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid "sliver" triangles.

Given a set S of points $p_1, p_2, \ldots, p_N$, we can compute the Delaunay triangulation of S by first computing its *Voronoi diagram*. The Voronoi diagram decomposes the 2D space into regions around each point such that all the points in the region around p_i are closer to p_i than they are to any other point in S. Given the Voronoi diagram, the Delaunay triangulation can be formed by connecting the centers of every pair of neighboring Voronoi regions. Figure 7.2(left) shows a set of 2D points, their Voronoi diagram is shown in Figure 7.2(middle) while their Delaunay triangulation is shown in Figure 7.2(right). Delaunay triangulation has certain properties, including: (i) the Delaunay triangulation of a non-degenerate set of points is unique, (ii) a circle through the three points of a Delaunay triangle contains no other points and (iii) the minimum angle across all the angles in all the triangles in a Delaunay triangulation is greater than the minimum angle in any other triangulation of the same points.

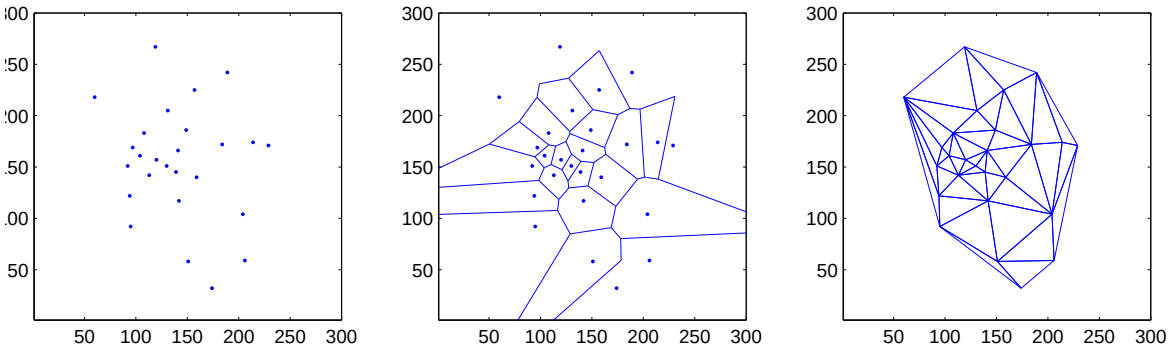


Figure 7.2: (left) A set of points, (middle) its Voronoi diagram, and (right) its Delaunay triangulation.

Property i supports the use the Delaunay triangles for indexing. Property ii implies that the insertion of a new point in a Delaunay triangulation affects only the triangles whose circum circles contain that point. As a result,noise affects the Delaunay triangulation only locally. This is very important in the context of our application. The last

property implies that the triangles obtained are not *skinny*. This is also very desirable in our application since the computation of the geometric transformations between fingerprints is based on corresponding minutiae triangles. Using skinny triangles can lead to instabilities and errors [54]. In a comparison study that involved several well known topological structures [85], the Delaunay triangulation was found to have the best structural stability under random positional perturbations.

The Delaunay triangulation and the Voronoi diagram are very efficient algorithms since the number of edges in both of them is proportional to a small constant times the number of points ($O(N)$). Since each edge belongs to at most two triangles or polygons, then the number of triangles generated by the Delaunay triangulation is also linear to the number of points. The complexity of the algorithm is $O(Nlog(N))$ (N is the number of points).

7.2.2 Building the Index Table

Once the minutiae have been extracted, their Delaunay triangulation is computed. Figure 7.3 demonstrates the Delaunay triangulation of the minutiae extracted from one of the fingerprints.

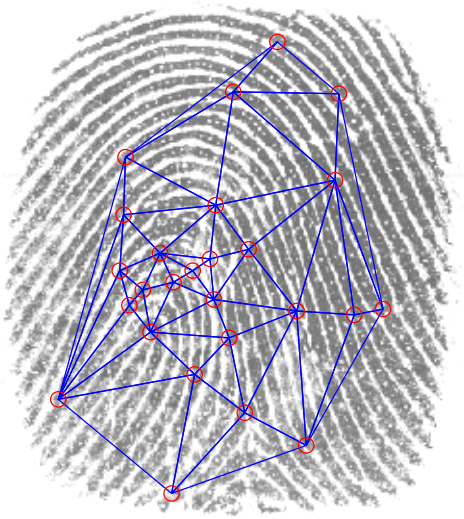


Figure 7.3: The Delaunay triangulation of minutiae.

The index table is built by considering the minutiae triangles formed by the Delaunay triangulation. Then, an index is formed using the invariants and appropriate information is stored in the indexed table location. Without using the Delaunay triangulation, we would have to consider every possible triangle. Assuming N minutiae on the average, the number of possible triangles is $O(N^3)$. In contrast, the Delaunay triangulation yields only $O(N)$ triangles. Since these triangles satisfy the properties of the Delaunay triangulation, they can be found through a well defined procedure and have good discrimination power. Using these triangles for indexing preserves index selectivity and allows for implementing a low dimensional indexing scheme.

Given a minutiae triangle (e.g., see Figure 7.4), we compute invariants which are then used to form a high dimensional index. The invariants are based on the edges and angles of the minutiae and the orientation field:

$$(L,\ \theta_1,\ \theta_2,\ T_1,\ T_2,\ R)$$

where L is the length l_3 of edge AB (here, we use edge AB as an instance); θ_1 and θ_2 are the angles between the edge AB and the orientation field at the point A and B respectively; T_1 and T_2 are minutiae types of minutiae A and B; R is the number of ridges that edge AB crosses over.

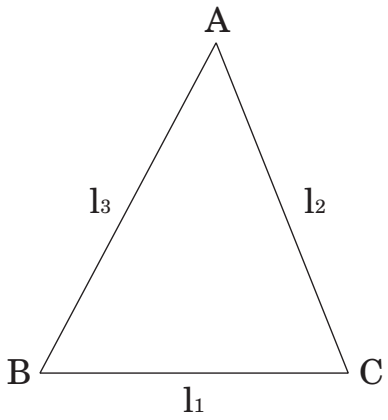


Figure 7.4: Invariants using the minutiae triangle.

As using triangle edge as comparing index has many advantages. We first compute the Delaunay triangulation of minutiae sets of input and template fingerprints. Second, use triangle edge as our comparing index. To compare two edges, $L, \theta_1, \theta_2, T_1, T_2, R$ values are used, all of which invariant of the translation and rotation. We assume two edges match if they satisfy the following set of conditions:

$$\frac{|L_{input} - L_{template}|}{max(L_{input}, L_{template})} < t_1$$

$$|\theta_{1input} - \theta_{1template}| < t_2$$

$$|\theta_{2input} - \theta_{2template}| < t_2$$

$$T_{1input} = T_{1template}$$

$$T_{2input} = T_{2template}$$

$$|R_{input} - R_{template}| < t_3$$

where t_i is a threshold.

If one edge from an input image matches two or more edges from the template image, we need to consider the triangulation to which this triangle edge belongs to and compare the triangle pair. Each index generated by a input fingerprint is used to retrieve all template fingerprints stored in the database under the same index.

Most indexing-based approaches accumulate evidence about a template by casting a vote for every entry stored in the indexed locations and by *histograming* the entries to pick

the ones which have received a high number of votes [54]. The problem with this approach is that it takes into consideration only the number of votes received by a particular entry and does not consider whether these votes are consistent among themselves. To introduce a measure of coherence, Lamiroy and Gros have proposed voting in the transformation space [55]. The key idea behind this approach is to consider transformations which form large clusters in the transformation space. The same idea was also used in [35].

We have also adopted this idea in our work since it is very effective. Specifically, each of the entries retrieved from the index table represents a hypothesized correspondence between minutiae in the input fingerprint and minutiae in the template fingerprint. These correspondent minutiae are the control-points we expect for our combined RBF model.

7.3 Fingerprint Deformation

The most conversional fingerprint matching techniques simply use rigid-Affine transformation to deal with distortion. However, they invariably lead to unsatisfactory matching results since the distortion is basically elastic in nature, owning to the soft tissue of finger. Actually, distortion is a combination of rigid and nonrigid transformations. Below, we briefly discuss the rigid-affine model and the nonrigid model.

7.3.1 Rigid-Affine Model

A spatial transformation is considered *rigid* if the spatial distance between consecutive points is preserved. Rigid transformation can be decomposed into a translation and/or rotation. A translation is a constant displacement over space. A rigid model is a constrained subset of an affine model. So, an affine transformation can be decomposed into a linear transformation and a translation. In 2D, a simple affine transformation can be expressed by a linear polynomial of the following equation:

$$f_k(\vec{x}) = a_{1k} + a_{2k}x + a_{3k}y \qquad k = 1, 2 \tag{7.1}$$

where $\vec{x} = (x, y)$.

With a rigid-affine model, fingerprint structures retain their shape and form during matching. This limits their practical application. To address the issue of deformable behavior, we need an elastic model in which a structure may not necessarily retain its shape or form during transformation.

7.3.2 Nonrigid Model

A nonrigid transformation is opposite to a rigid one, in which the spatial distance between consecutive points cannot be preserved owing to various pressures on finger surface. In this case, prints are compressed on themselves or stretched. Such behavior is synonymous with the characteristics of a higher degree polynomial:

$$f_k(\vec{x}) = \sum_{i=0}^{n}(a_{1k}x^i + a_{2k}y^i) \qquad k = 1, 2;\ n \geq 3 \tag{7.2}$$

7.3.3 Radial Basis Function

Scattered data interpolation is fundamental in deriving a smooth spatial transformation from the correspondence of minutiae between a pair of prints. The advantage of using scattered data interpolation methods to model deformation is that they need fewer control-points, or the control-points are sparsely distributed. The problem of scattered data interpolation can be formulated as follows:

Problem 1 *Given n pairs of data points $(\vec{x}_i, \vec{u}_i)$, where $\vec{x}_i, \vec{u}_i \in R^d$ and $i = 1, \ldots, n$, derive a continuous function $f : R^d \mapsto R^d$ with $f(\vec{x}_i) = \vec{u}_i$.*

Control-points in input and template prints can be extracted by an indexing algorithm using Delaunay triangulation. One way of approaching scattered data interpolation is to use *Radial Basis Function*, abbreviated RBF. The RBF [73, 3] offers several advantages. First of all the geometry of the control-points is by no means restricted. This implies that the distribution of control-points can be both sparse and irregular. Secondly, the RBF provides easily controllable behavior that can be tailored to meet specific requirements. An RBF may be purely deformable, or it may contain some form of linear component, allowing both local and global deformations.

RBF Interpolation

Generally, a RBF spatial transformation in d dimensions, denoted $T(\vec{x})$, is composed of k mapping functions $k = 1, \ldots, d$ such that:

$$T(\vec{x}) = [f_1(\vec{x}), \ldots, f_k(\vec{x}), \ldots, f_d(\vec{x}), \ldots,] \tag{7.3}$$

where $f_1(\vec{x})$ represents the mapping function in the first dimension etc. Each of the mapping functions can be decomposed into a global component and a local component. Although the two components are distinct, they are evaluated almost simultaneously, giving rise to a single transformation. This decomposition enables a family of transformations to be defined where, if desired, the influence of each control-point can be controlled. Given n corresponding control-point pairs, each of the k mapping functions of the RBF has the following general form:

$$f_k(\vec{x}) = P_{mk}(\vec{x}) + \sum_{i=1}^{n} A_{ik} g(r_i) \tag{7.4}$$

The first component $P_{mk}(\vec{x})$ is the global linear transformation denoted by a polynomial of degree m. In 2 dimensions, the general form of a linear polynomial (see formula 7.1, $m = 1$) is used, making the global component an affine transformation. The latter component is the sum of a weighted elastic or nonlinear basis function $g(r_i)$, where r_i denotes the Euclidean norm, such that:

$$r_i = [\vec{x} - \vec{x}_i]^{\frac{1}{2}} \quad or \quad r_i = \|\vec{x} - \vec{x}_i\| \tag{7.5}$$

The coefficients of the function $f_k(\vec{x})$ are determined by following conditions:

$$f_1(\vec{x}_j) = u_j \quad and \quad f_2(\vec{x}_j) = v_j \quad for \ j = 1, \ldots, n \tag{7.6}$$

giving n linear equations together with the additional compatibility constraints:

$$\sum_{i=1}^{n} A_{ik} = \sum_{i=1}^{n} A_{ik}x_i = \sum_{i=1}^{n} A_{ik}y_i = 0 \tag{7.7}$$

These conditions guarantee that the RBF is affine reducible. The coefficients of the basis function and the polynomial can now be found by solving the linear system:

$$W = L^{-1}Y \tag{7.8}$$

where

$$L = \begin{bmatrix} G & P \\ P^T & 0 \end{bmatrix} \tag{7.9}$$

$$G = \begin{bmatrix} g(r_{11}) & g(r_{12}) & \cdots & g(r_{1n}) \\ g(r_{21}) & g(r_{22}) & \cdots & g(r_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ g(r_{n1}) & g(r_{n2}) & \cdots & g(r_{nn}) \end{bmatrix} \tag{7.10}$$

$$P^T = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n) \end{bmatrix} \tag{7.11}$$

$$W^T = \begin{bmatrix} A_{11} & A_{21} & \cdots & A_{n1} & a_{11} & a_{21} & a_{31} \\ A_{12} & A_{22} & \cdots & A_{n2} & a_{12} & a_{22} & a_{32} \end{bmatrix} \tag{7.12}$$

$$Y^T = \begin{bmatrix} u_1 & u_2 & \cdots & u_n & 0 & 0 & 0 \\ v_1 & v_2 & \cdots & v_n & 0 & 0 & 0 \end{bmatrix} \tag{7.13}$$

Basis Functions

A number of candidate 2D basis functions are given in Table7.1. The choice of a basis function is determined by the interpolation conditions and the desired properties of the interpolation. Radial basis functions can be tailored to many specific needs, and their range of influence can be controlled by adjusting the parameters of the RBF.

As Table 7.1 illustrates, both the TPS and multiquadric basis functions monotonically increase with distance from the center. In contrast, the Wendland, Gaussian and Inverse MQ basis functions monotonically decrease with distance from the center. Since fingerprint distortion is elastic rather than viscous (like brain tissue for example), the influence of control-points on the corrected results increases with distance at a certain range from the center. On the other hand, fingerprint distortion is not globally uniform. For instance, if one side of a traction deformation center is compressed, the opposite side

Table 7.1: Basis Functions

Basis Function	$g(r_i)$	**Parameters**
Linear	r_i	-
Thin-Plate Spline(TPS)	$r_i^2 \log r_i$	-
Wendland	$(1-r_i)_+^4(4r_i+1)$	-
Multiquadric(MQ)	$(r_i^2+\delta)^{+\mu}$	$\delta > 0, 0 < \mu$
Inverse Multiquadric	$(r_i^2+\delta)^{-\mu}$	$\delta > 0, 0 < \mu$
Gaussian	$e^{(-r_i^2/\delta)}$	$\delta > 0$

must be dilated. Therefore, MQ suits elastic deformation. Gaussian and Inverse MQ basis functions, especially Wendland's Compactly Supported Function [32] are more suitable for viscous deformation problems. As TPS and Wendland have no parameters, clearly they cannot adapt themselves to a particular deformation problem. They may not be a good choice for our problem.

In the following section, we discuss the interpolation effects by means of these five basis functions with different parameters, and find out the best basis function, meanwhile, choose the best parameter.

7.4 Experimental Comparison Among Various Basis Functions

To show how the nonrigid deformation model can deal with distortion by means of different RBF basis functions, a numerical evaluation measure employs an average distance of all minutiae pairs between input print and template print:

$$\bar{r} = \frac{\sum_{i=1}^{n}[(u_i' - u_i)^2 + (v_i' - v_i)^2]^{\frac{1}{2}}}{n} \tag{7.14}$$

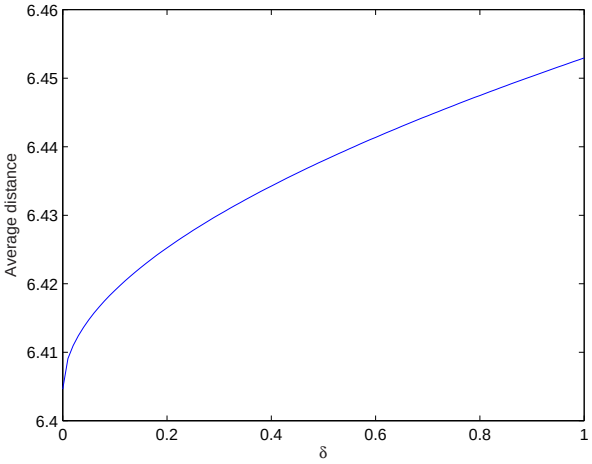
where $f_1(\vec{x}_i) = u_i'$ and $f_2(\vec{x}_i) = v_i'$ for $i = 1, \ldots, n$, and point (x_i, y_i) is a minutia in input print, similarly, point (u_i, v_i) is the corresponding minutia in template print.

7.4.1 Evaluation Using Different Parameter Values

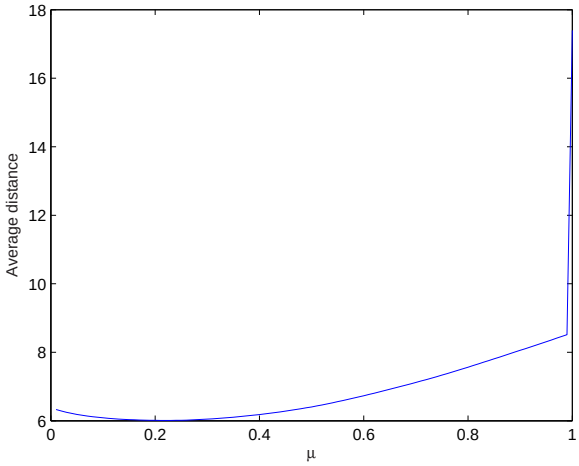
As an illustration of Table 7.1, MQ, inverse MQ, and Gaussian basis functions have at least one parameter, so we evaluate these functions through a reasonable range of each parameter. On the contrary, TPS and Wendland basis functions produce only one value for each of them.

Our strategy is follows: First, fix μ' by a value common in literatures and then find out the δ^* which locally minimizes the $\bar{r}$. Finally, find out the μ^* making $\bar{r}$ minimal using δ^* (see Figure 7.5 for an illustration).

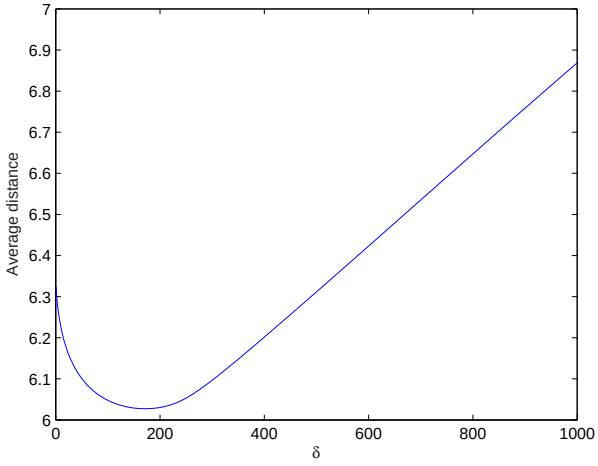
We tested 20 pairs of prints from different fingers scanned by the FUJITSU Fingerprint Sensor (model: FS-210u). Size and resolution are 300×300 and 500dpi respectively. These fingerprints contain horrible distortions than the prints in the FVC2000 database



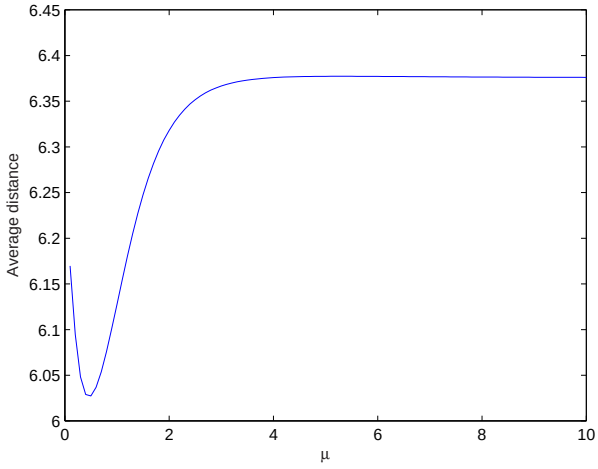
(a) MQ $(0 \leq \delta \leq 1; \mu = 0.5)$



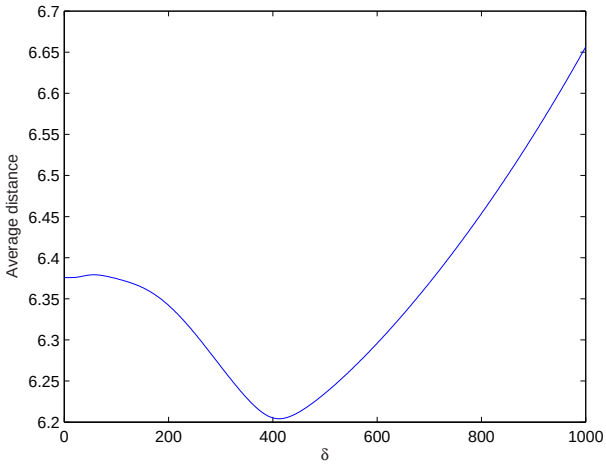
(b) MQ $(\delta^* = 0; 0 \leq \mu \leq 1)$



(c) IMQ $(0 \leq \delta \leq 1000; \mu = 0.5)$



(d) IMQ $(\delta^* = 171; 0 \leq \mu \leq 10)$



(e) Gaussian $(0 \leq \delta \leq 1000)$

Figure 7.5: (a) and (b) show $\mu^* = 0.2$, $\delta^* = 0$ can make a normal RBF model using MQ basis function locally minimal; (c) and (d) are similar to (a) and (b), $\mu^* = 0.46$, $\delta^* = 171$; (e) show $\delta^* = 412$ can make a normal RBF model using Gaussian basis function locally minimal.

[?]. Figure 7.5 shows the average searching results of the MQ, inverse MQ, and Gaussian basis functions through a reasonable parameter range. The parameters found and the minimal average distance $\bar{r}$ are listed in Table 7.2. We also list the average distance $\bar{r}_{rigid}$ of minutiae pairs with rigid transformation.

Table 7.2: Evaluation of various basis functions.

Basis Function	$min\,\bar{r}$	**Parameters**
Thin-Plate Spline(TPS)	8.5643	-
Wendland	6.3475	-
Multiquadric(MQ)	6.0104	$\delta^* = 0, \mu^* = 0.2$
Inverse Multiquadric	6.0264	$\delta^* = 171, \mu^* = 0.46$
Gaussian	6.2041	$\delta^* = 412$
Rigid Transformation	$\bar{r}_{rigid} = 17.5852$	

The results show that the MQ basis function with $\delta = 0, \mu = 0.2$ obtains the best correction. It closely meets the analysis in subsection 7.3.3. However, the parameters generated by the above strategy are not optimal, nor are they adaptive. Our goal is to further improve the accuracy of the nonrigid model. To this end, we employed locality parameters.

7.4.2 Evaluation Using Locality Parameters

As the property of fingerprint distortion, we would like to control the area of influence of each basis function. *Locality parameters*, which can control the range of influence of a basis function, can perform that task. They give less weight to distant control-points and more weight to neighboring ones. For this reason, locality parameters induce a better smoothness of the interpolation at the given control-points. The selection of such parameters is critical for improving fingerprint matching.

An adaptive locality parameter was suggested by D. Ruprecht and H. Müler [73]. The value of δ is extended to use unique values for each control-point, calculated from the distance to the nearest neighboring control-point. This essentially allows the distortion to be softer where control-points are widely spaced and stronger where they are closer together. This may more effectively work on variably sparse or dense control-point set, such as fingerprint minutiae. From the evaluation described in the previous subsection, the MQ basis function gets the best score. Using this adaptive locality parameter, it is modified as:

$$g(r_i) = (r_i^2 + \delta_i)^{\pm\mu} \tag{7.15}$$

where $\delta_i = min_{i \neq j}(r_{ij})\ \ (j = 1, \ldots, n)$. Here, we simply apply Delaunay triangulation [7] on these control-points, then the δ_i can be easily obtained.

The results are listed in Table 7.3. It can be seen that the locality parameter does indeed improve interpolation of MQ, but not much, because RBF depends heavily on control-points selection. If the control-points are uniformly distributed on fingerprint, our RBF distortion model works very well. Unfortunately, our alignment algorithm cannot

guarantee this severe requirement. Therefore, we introduced a more precise distortion model.

7.5 A Combined RBF Model

When the finger tip is orthogonally pressed against the plain surface of a fingerprint sensor, not whole finger surface is orthogonal to the sensor surface due to the convexity and soft tissue of the finger. Therefore, the pressures on the fingerprint are not uniform, but monotonically decrease from the center (see Figure 7.6). We roughly separate a fingerprint image scanned by planar sensor into two regions [14]:

I *Rigid region.* This is the closest contact region with the highest pressure, which normally does not allow skin slippage. In our method, the radii of region **I** is 1/3 of radius of whole fingerprint region.

II *Nonrigid region.* Pressure monotonically decreases from the boundary of region **I** to the external boundary of region **II**. The main elastic distortion is located in this region.

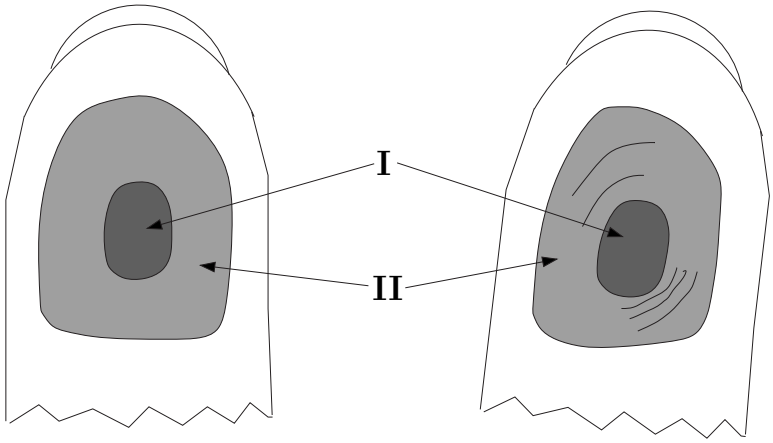


Figure 7.6: Bottom views of a finger without and with distortion, two pressure touching regions.

With this observation, we are ready to modify the method in previous section. From section 7.3.3, we know RBF consists of a linear transformation and a nonlinear transformation (see equation 1,4). All coefficients of both components are calculated simultaneously in the previous RBF model. Therefore, the control-points in nonrigid region **II** must affect the accuracy of the linear polynomial. For distortion correction, it is advantageous to more precisely compute the linear polynomial. According to the definition of rigid region **I**, the minutiae in this region are almost relatively motionless, whether the print contains distortion or not. Hence the control-points in region **I** can contribute to the polynomial of the linear transformation of the whole print. The distortion in nonrigid region **II** is caused by the same linear transformation and the local nonlinear transformation synchronously. Our strategy is as follows: first, estimate the linear transformation of the whole

print by the control-points in region **I**; second, remove the linear transformation effect of the control-points in region **II**, and then construct a RBF distortion model for region **II**; finally, correct distortions in regions **I** and **II**, respectively. According to this idea, we modified the method and equations 8,12,13 as follows:

$$W' = L^{-1}Y', \tag{7.16}$$

where

$$\varphi_{ik} = a_{1k} + a_{2k}x_i + a_{3k}y_i \ , \tag{7.17}$$

$$Y'^T = \begin{bmatrix} u_1 - \varphi_{11} & \cdots & u_n - \varphi_{n1} & 0 & 0 & 0 \\ v_1 - \varphi_{12} & \cdots & v_n - \varphi_{n2} & 0 & 0 & 0 \end{bmatrix}, \tag{7.18}$$

$$W'^T = \begin{bmatrix} A_{11} & A_{21} & \cdots & A_{n1} & 0 & 0 & 0 \\ A_{12} & A_{22} & \cdots & A_{n2} & 0 & 0 & 0 \end{bmatrix}. \tag{7.19}$$

Input: Minutiae X and U from input and template fingerprints respectively.

Output: Minutiae X' in the corrected fingerprint.

1. *Substitute three paired control-points*[1] $\vec{x}_i^c \in X_I$ *and* $\vec{u}_i^c \in U_I$*, which locate in rigid regions* **I***, into equation 1, and then obtain* a_{ik}*.*
2. $\forall \vec{x}_i^c \in X_{II}$ *and* $\forall \vec{u}_i^c \in U_{II}$*, which locate in nonrigid regions* **II***, Orderly compute equations 17, 18 and 16. Obtain* A_{ik}*.*
3. *For all* $\vec{x}_i \in X$

   If $\vec{x}_i \in X_I$*; Then* $\vec{x}'_i \leftarrow$ *equation 1.*

   If $\vec{x}_i \in X_{II}$*; Then* $\vec{x}'_i \leftarrow$ *equation 4.*

   End

In steps 2 and 3, we still employ the RBF model with locality parameters. The visual results are shown in Figure 7.7 and the average distance is listed in Table 7.3. Figure 7.7a and b are prints without and with distortion. Part c and d are correction results using the rigid and combined RBF model, respectively. Circles in these figures denote their own minutiae; crosses denote the mapped minutiae from prints with distortion. Part d clearly shows that most mapped minutiae are very close to the minutiae to be matched. The numerical measure $\bar{r_1}$ of Figure 1d is 5.2478; $\bar{r_2}$ of Figure 2d is 5.2045. Then, these visual results and a better average distance $\bar{r}$ prove that the combined RBF model has more capability to cope with distortion owing to the more precise rigid transformation.

[1]Formula 1 shows that there are six coefficients to be specified. This means at least three paired control-points in region **I** are required. Experiments show that this demand can be easily filled.


1a(left), 2a(right) Regular prints.


(1b, 2b) Prints with distortion.


(1c, 2c) Using rigid transformation.

(1d, 2d) Using RBF combined-distortion model.

Figure 7.7: Minutiae correspondence before and after the application of combined RBF model

Table 7.3: Comparative results among MQ models

Method(MQ)	$\bar{r}$	**Parameters**
Global Parameter	6.0104	$\delta^* = 0, \mu^* = 0.2$
Locality Parameter	5.9454	$\delta_i, \mu^* = 0.2$
combined-distortion model	5.2136	$a_{ik}, \delta_i, \mu^* = 0.2$
Rigid Transformation	$\bar{r}_{rigid} = 17.5852$	

7.6 Conclusion

This chapter introduces a combined RBF model for correcting fingerprint elastic distortion. We first studied the properties of RBF and proposed a normal RBF model with an appropriate basis function and locality parameters, rather than the parameters common in literature. However, this improvement is not great. According to the particularity of fingerprint distortion, we further propose an improved combined RBF model, which separately builds rigid and nonrigid transformations, to attack the distortion problem. Combined RBF model provides more accurate mapping function between a possible matched-pair. Experiments show our combined RBF model can decrease a horrible distortion into around 30% compared with a rigid transformation.

Our future efforts will target a more in-depth study of the properties of fingerprint distortion and the design of an effective and efficient optimization basis function.

Chapter 8

Fingerprint Matching

8.1 Introduction

A fingerprint matching algorithm compares a input fingerprint with a templet fingerprint stored in database; and returns either a degree of similarity (without loss of generality, a score between 0 and 1) or a binary decision (*mated/non-mated*). Most matching algorithms require an intermediate fingerprint representation derived through a feature extraction stage. Among them, minutiae matching is certainly the most well-known and widely used method for fingerprint matching, thanks to its strict analogy with the way that forensic experts compare fingerprints and its acceptance as a proof of identity in the courts of law in almost all countries.

Let T and I be the representation of a template and input fingerprint, respectively. Each minutia may be described by a number of attributes, including its location in the fingerprint image, orientation, type (e.g., ridge termination or ridge bifurcation), a weight based on the quality of the fingerprint image in the neighborhood of the minutia, and so on. Most common minutiae matching algorithms consider each minutia as a triplet $m = \{x, y, \theta\}$ that indicates the (x, y) minutia location coordinates and the minutia angle θ:

$$\begin{array}{llll} T = \{m_1, m_2, \ldots, m_m\}, & m_i = \{x_i, y_i, \theta_i\}, & i = 1, \ldots, m \\ I = \{m'_1, m'_2, \ldots, m'_n\}, & m'_j = \{x'_j, y'_j, \theta'_j\}, & j = 1, \ldots, n \end{array}$$

where m and n denote the number of minutiae in T and I, respectively.

A minutia m'_j in I and a minutia m_i in T are considered "matching", if the *spatial distance sd* between them is smaller than a given tolerance r_0 and the *direction difference dd* between them is smaller than an angular tolerance θ_0:

$$\begin{array}{l} sd(m'_j, m_i) = \sqrt{(x'_j - x_i)^2 + (y'_j - y_j)^2} \leq r_0 \\ dd(m'_j, m_i) = \min(|\theta'_j - \theta_i|, 360^\circ - |\theta'_j - \theta_i|) \leq \theta_0 \end{array} \tag{8.1}$$

Equation 8.1 takes the minimum of $|\theta'_j - \theta_i|$ and $360^\circ - |\theta'_j - \theta_i|$ because of the circularity of angles. The tolerance boxes (or hyper-spheres) defined by r_0 and θ_0 are necessary to

compensate for the unavoidable errors made by feature extraction algorithms and to account for the small distortions that cause the minutiae positions to change.

Let $mm(.)$ be an indicator function (*consistency measure*) that returns 1 in the case where the minutiae m'_j and m_i match according to Equation 8.1:

$$mm(m'_j, m_i) = \begin{cases} 1 & sd(m'_j, m_i) \leq r_0 \text{ and } dd(m'_j, m_i) \leq \theta_0 \\ 0 & otherwise \end{cases} \tag{8.2}$$

Then, the fingerprint minutiae matching problem can be formulated as follows:

$$\max \min \sum_{i=1}^{m} mm(m'_{PD(i)}, m_i) \tag{8.3}$$

where $PD(i)$ is the alignment algorithm in section 7.2.2, which determines pairing between I and T minutiae; in particular, each minutia has either exactly one mate in the other fingerprint or has no mate at all.

To achieve consistency measure $mm(.)$ in geometry aspect, the classical way is based on tolerance box (e.g. circle, square or rectangle) [42, 43]. Jain, Hong and Holle. [47] and Luo, Tian and Wu [60] proposed an adaptive tolerance box with respect to distance in polar coordinates.

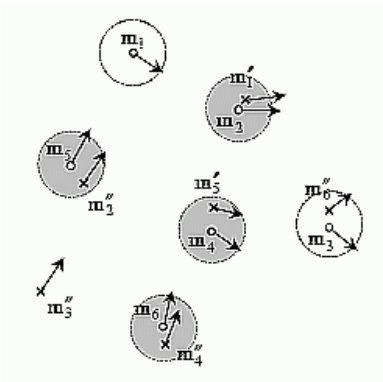


Figure 8.1: Minutiae of T are denoted by $\circ$, whereas I minutiae are denoted by $\times$. Pairing is performed according to the minimum distance. The circles indicate the tolerance box (the maximum spatial distance). The gray circles denote successfully mated minutiae; minutia m_1 of T and minutia m'_3 of I have no mates, minutiae m_3 and m'_6 cannot be mated due to their large direction difference.

Although, all of above matching methods employ some geometry information of minutiae (e.g. type, coordinate and orientation). However, other particular information of minutiae (especially for bifurcation) were lost, such as the shape of bifurcation. To include these information, precise calculating is naive and time consuming. In this chapter, we design a simple polygon that include all previous information for bifurcation (and termination) minutia. Hence, based on a deterministic polygon matching algorithm, a new fingerprint matching method is naturally proposed (see Figure 8.2 for an illustration). Its time complexity of one simple polygon is $(m+n)\log(m+n)$.

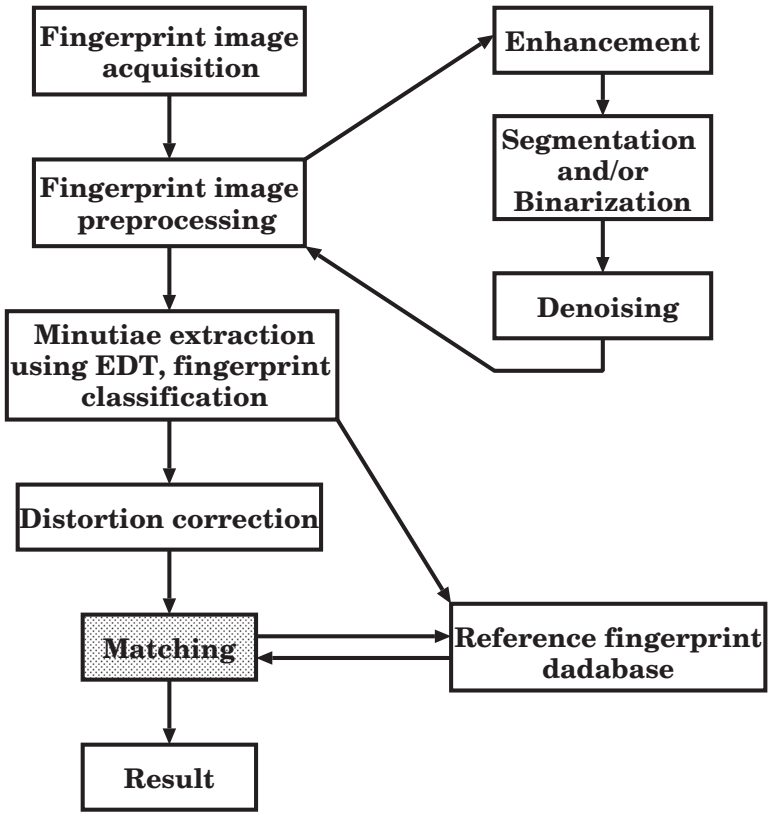


Figure 8.2: A flowchart showing different phases of fingerprint analysis. The highlighted block shows our work in this chapter.

8.2 Minutiae Polygons

8.2.1 Polygons of Bifurcation Minutiae

Let p be a bifurcation minutia with three ridges incident upon it, namely, where r is the ridge before bifurcation, r_1 and r_2 are the two ridges after bifurcation. For each of r, r_1 and r_2, consider a line segment, which has length λ and is tangent to the corresponding ridge at p. Let these three line segments be $\overline{b_1p}, \overline{b_2p}$ and $\overline{ap}$ corresponding to r_1, r_2 and r, respectively (see Figure 8.3 for an illustration). Let θ be the angle made by $\overline{ap}$, measured in counterclockwise direction with regard to x-axis. We call the group of the three line segments $\overline{b_2p}, \overline{b_1p}$ and $\overline{ap}$ as the *bifurcation detail* $\mathcal{B}(p, a, b_1, b_2)$ for minutia p.

Now consider a square $ABCD = \mathcal{S}(p, \theta, \epsilon)$ having side length ϵ, centered about the minutia p, and having one of its sides perpendicular to $\overline{ap}$, as shown in Figure 8.3. As we go on sliding the minutia p with its three line segments within the ϵ-square, we get a bifurcation polygon $\mathcal{M}$ (see Figure 8.4).

The polygon $\mathcal{M}$ can have different shapes, depending on the mutual orientations of $\overline{b_1p}, \overline{b_2p}$ and $\overline{ap}$. When p coincides with A, the region around p can be divided into 4 quadrants, which are named as S_{LL} (lower left region), S_{UL} (upper left region), S_{UR} (upper right region), and S_{LR} (lower right region), as shown in Figure 8.4. Each of b_1 and b_2 can lie in any one of these 4 regions, thereby making $4 \times 4 = 16$ possibilities. Out of these 16 possibilities, however, there will be 10 cases having distinct mutual positions of

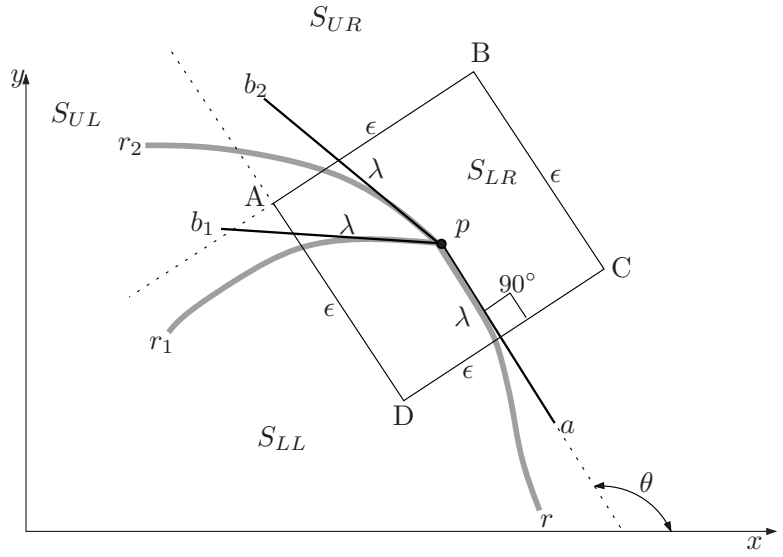


Figure 8.3: Minutia detail and defining ϵ-square box with regard to a bifurcation minutia at p.

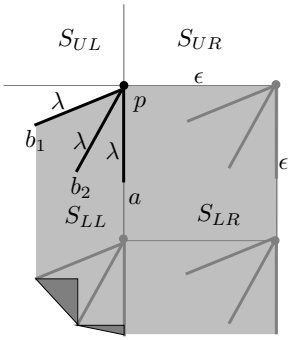


Figure 8.4: Bifurcation polygon (case 1).

b_1 and b_2, considering the inter-changeability of b_1 and b_2. That is, for example, the case of $b_1 \in S_{LR}$ and $b_2 \in S_{UR}$, and the case of $b_2 \in S_{LR}$ and $b_1 \in R_{UR}$, which are 2 different cases in 16 possibilities, are the same in the later 10 cases. These 10 cases are enumerated in Table 8.1. Cases 2, 6 and 9 can have two subcases each, depending on the relative x-axis (or, y-axis) coordinates of b_1 and b_2, which will have differently shaped polygons of $\mathcal{M}$. Therefore, we get $7 + 6 = 13$ different polygons, which are shown in Figures 8.4 and 8.5.

For a valid bifurcation minutia p, the angle $\angle(b_1, p, b_2)$ should be less than both $\angle(b_1, p, a)$ and $\angle(b_2, p, a)$, which helps us to distinguish r form r_1 and r_2. Keeping this point in consideration, out of the above 10 cases, cases 3, 4 and 7 are not possible for a valid bifurcation minutia. Hence we have only $13 - 3 = 10$ possible differently shaped polygons.

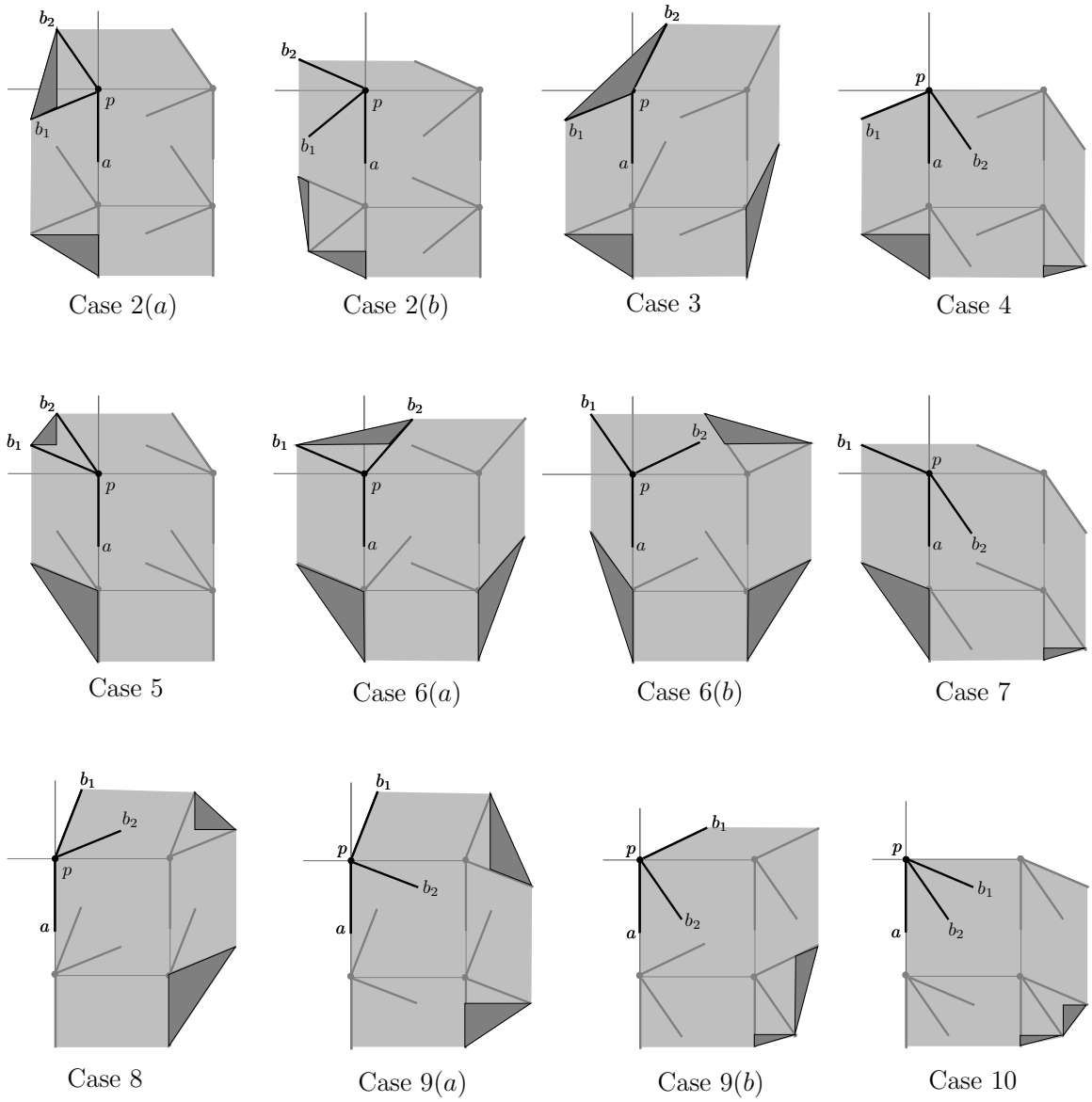


Figure 8.5: Bifurcation polygons (case 2 – 10).

Table 8.1: Different cases for polygon formation

Case	$b_1 \in$	$b_2 \in$	Condition
1	S_{LL}	S_{LL}	
2 (a)	S_{LL}	S_{UL}	$x[b_1] \leq x[b_2]$
2 (b)	S_{LL}	S_{UL}	$x[b_1] > x[b_2]$
3	S_{LL}	S_{UR}	
4	S_{LL}	S_{LR}	
5	S_{UL}	S_{UL}	
6 (a)	S_{UL}	S_{UR}	$y[b_1] \leq y[b_2]$
6 (b)	S_{LL}	S_{UR}	$y[b_1] > y[b_2]$
7	S_{UL}	S_{LR}	
8	S_{UR}	S_{UR}	
9 (a)	S_{UR}	S_{LR}	$x[b_1] \leq x[b_2]$
9 (a)	S_{UR}	S_{LR}	$x[b_1] > x[b_2]$
10	S_{LR}	S_{LR}	

8.2.2 Polygon of Termination Minutiae

Let p also be a termination minutia. As there is only one ridge r incident upon it. A termination detail $\mathcal{T}(p, a)$ has just one line segment $\overline{ap}$ corresponding to r, which is defined as same as bifurcation detail. So, only the line segment $\overline{ap}$ contributes to termination polygon. Analogously slid the minutia p with line segment $\overline{ap}$ within square $\mathcal{S}(p, \theta, \epsilon)$, we get a termination polygon $\mathcal{M}$, i.e. a rectangle (see Figure 8.6). Certainly, the shape of termination polygon is unique.

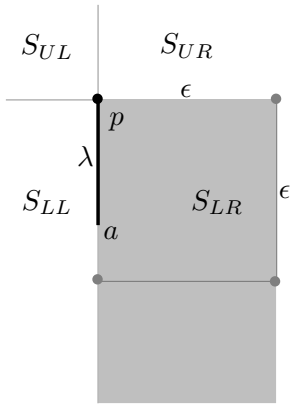


Figure 8.6: Termination polygon.

8.3 Polygon Generation

To generate the polygons for minutiae, those three line segments $\overline{b_2p}, \overline{b_1p}$ and $\overline{ap}$ are required. In discrete image, however, obtaining such above line segment is a non-trivial task. Difficulty mainly focuses on the fingerprint ridge are a thick line object (i.e. the

width of ridge are always greater than one pixel). On the contrary, the line segment for generating polygon is a thin object (i.e. one pixel wide). How to determine the line segment from a thick ridge has arisen.

In image processing, skeleton is often used to represent the structure of an object. Essentially, the above line segment is another representation of a skeleton of fingerprint ridge. Therefore, skeleton of minutiae ridge can be employed for minutiae polygon generation. There are a lot of skeleton algorithms were described in literature [93, 10, 36, 8]. Owing to our system uses Euclidean distance transform through preprocessing of AFIS for reducing processing time, in this phase, we also employ a skeleton extraction algorithm using Euclidean distance transform [79].

8.3.1 Skeleton Extraction

The distance function of pixel $p \in Ob$ is defined as the smallest distance of p from all the background pixels $q \in Bg$ (please refer Equation 3.2). That is

$$I_d(p) = \min_{q \in Bg}\{d(p,q)\}$$

where $d(p,q)$ is the Euclidean distance between p and q. The skeleton can be easily derived that if and only if a point p belongs to the skeleton of a ridge and a maximal disk with the radius $I_d(p)$ hits the contour of Bg at least two places. We give out a definition as below:

Definition 6 *If the set $A(p)$ contains more than one element, then p is a skeleton point.*

$$A(p) = \{q | d(p,q) = I_d(p), q \in Bg\}$$

Prior to giving out the skeleton extraction algorithm, two basic definitions are illustrated below.

1. *Start point*: The start point is one on the boundary of minutiae region (please refer Figure 6.9), whose non-minutia neighbor has a *9 EDT circle* covering two background components. The start points are considered as sources or elementary cells of the skeleton which grows up emitting from it. Start point is colored red in Figures 8.I and 8.II. Its non-minutia neighbor, which has a *9 EDT circle* covering two background components, is colored green.

2. *Branch generation*: The set of points $\{p_i\}$ is called the *directional-neighborhood* of p, denoted by D_p, if they are in the 8-neighborhood of p and located within $\pm 45^\circ$ slope changes from the current medial axis orientation of p. For example, using the 8-neighbors labeled $p_1, p_2, \ldots, p_8$ counterclockwise from the positive x axis of p, if p_7 and p are the skeleton points, the points p_2, p_3, p_4 are the directional neighbors of p, i.e. $D_p = \{p_2, p_3, p_4\}$. Several cases are illustrated below:

$$\begin{array}{ccc} p_4 & p_3 & p_2 \\ \cdot & p & \cdot \\ \cdot & p_7 & \cdot \end{array} \qquad \begin{array}{ccc} \cdot & \cdot & p_2 \\ p_5 & p & p_1 \\ \cdot & \cdot & p_8 \end{array} \qquad \begin{array}{ccc} p_4 & p_3 & \cdot \\ p_5 & p & \cdot \\ \cdot & \cdot & p_8 \end{array} \qquad \begin{array}{ccc} \cdot & p_3 & p_2 \\ \cdot & p & p_1 \\ p_6 & \cdot & \cdot \end{array}$$

Note that the set of directional-neighborhood contains always three elements. The branch generation is to add the point which is the maximum of p's directional-neighborhood (see the blue points in Figures 8.I and 8.II). That is to say:

$$p_{next} = \max_{p_i \in D_p} \{p_i\} \tag{8.4}$$

3. *Minutia point*: The minutia point is the one being the maxima (i.e. the largest EDT value locally) in extracted minutiae region. Occasionally, there would exist more than one maxima points in one minutiae region. In this case, the midpoint among them is chosen as the minutia point.

The algorithm of skeleton extraction for minutiae is described as follows:

Input: Fingerprint image I_{EDT} represented by EDT values and minutiae regions.

Output: Short skeleton incident upon the minutia.

1. *Pick up a start point from minutiae region as initial skeleton point.*
2. *Choose its non-minutia neighbor that has a 9 EDT circle covering two background components as next skeleton point. If number of eligible neighbors is more than one, choose the midpoint of them.*
3. *Starting from start points and its chosen neighbor, the branch generation is used to add more skeleton points. For each branch, the branch generation halts at the fifth skeleton point.*
4. *Connect start points with minutia point to complete the skeleton.*

Following shows the skeleton of fingerprint minutia.

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\sqrt{2}$	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
$\sqrt{5}$	2	2	2	2	2	2	$\sqrt{2}$	1	1	1	0	0	0	0	0
$\sqrt{5}$	$\sqrt{8}$	3	3	3	3	$\sqrt{8}$	$\sqrt{5}$	2	2	$\sqrt{2}$	1	1	0	0	0
$\sqrt{2}$	2	$\sqrt{5}$	2	2	2	$\sqrt{5}$	$\sqrt{8}$	$\sqrt{8}$	$\sqrt{5}$	2	2	$\sqrt{2}$	1	0	0
1	1	$\sqrt{2}$	1	1	1	$\sqrt{2}$	$\sqrt{5}$	$\sqrt{5}$	$\sqrt{2}$	1	1	$\sqrt{2}$	$\sqrt{2}$	1	1
0	0	1	0	0	0	1	$\sqrt{2}$	$\sqrt{2}$	1	0	0	1	1	$\sqrt{2}$	$\sqrt{2}$
0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	$\sqrt{2}$
0	0	0	0	0	0	0	1	$\sqrt{2}$	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1	$\sqrt{2}$	1	1	0	0	0	0
$\sqrt{2}$	0	0	0	0	0	0	0	0	1	$\sqrt{2}$	$\sqrt{2}$	1	0	0	0
$\sqrt{5}$	0	0	0	0	0	0	0	0	0	1	2	$\sqrt{2}$	1	0	0

Figure 8.I: Skeleton of bifurcation.

1	0	0	0	0	0	0	0	0	0
$\sqrt{2}$	1	0	0	0	0	0	0	0	0
$\sqrt{5}$	$\sqrt{2}$	1	1	1	0	0	0	0	0
$\sqrt{8}$	$\sqrt{5}$	2	2	1	0	0	0	0	0
$\sqrt{13}$	$\sqrt{10}$	3	2	1	0	0	0	0	0
$\sqrt{8}$	$\sqrt{13}$	$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	0	0	0	0
$\sqrt{5}$	$\sqrt{10}$	$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	$\sqrt{2}$	1	0	0	0
2	3	4	$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	$\sqrt{2}$	1	0	0
2	$\sqrt{8}$	3	3	3	$\sqrt{8}$	$\sqrt{5}$	$\sqrt{2}$	1	0
$\sqrt{2}$	2	2	2	2	2	$\sqrt{5}$	2	1	0
1	1	1	1	1	1	$\sqrt{2}$	2	1	0
0	0	0	0	0	0	1	$\sqrt{2}$	1	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0

Figure 8.II: Skeleton of termination

8.3.2 Polygon Generation

The theoretical definition of line segments $\overline{b_1p}, \overline{b_2p}$ and $\overline{ap}$ defines that they are tangent to the corresponding ridges at minutia point p. But in discrete domain, it is a bit complicate in practice. An alternative approach simply connects the fifth skeleton point with minutia point on each branch. Therefore, they have almost 6 pixels length (please recall Section 6.2.3). Such line segments can generate similar minutia polygon without losing matching accuracy.

With the line segments $\overline{b_1p}, \overline{b_2p}$ and $\overline{ap}$ ready, we do *Minkowski sum* [51] of the minutia detail $\mathcal{B}(p, a, b_1, b_2)$ (or $\mathcal{T}(p, a)$) and the square $\mathcal{S}(p, \theta, \epsilon)$.

$$\mathcal{M} = \mathcal{B}(p, a, b_1, b_2) \cup \mathcal{S}(p, \theta, \epsilon) \quad \text{or} \quad \mathcal{T}(p, a) \cup \mathcal{S}(p, \theta, \epsilon) \tag{8.5}$$

Then ten different bifurcation polygons and one termination polygon are obtained.

8.4 Polygon Matching

The polygon matching is a kind of computation of the dissimilarity between the two polygons. In other words, given two shapes P and Q, how much do they resemble each other? (or: Are they identical up to some tolerance $\delta > 0$?) P will be the input polygon and Q is a template one. Here we will assume that P and Q are simple polygons in the plane. Geometrically, the problem above can be formulated as follows:

Problem 2 *Given P, Q, find the distance $d(P, Q)$ between P and Q.*

The simplest way to compute $d(P, Q)$ is measuring how much P and Q overlapped. Formally:

$$d(P,Q) = 1 - \frac{2P \bigcap Q}{P+Q} \tag{8.6}$$

From above equation, it is clear that the less $d(P,Q)$ denotes P and Q are closer and more resemblant. However, if we fix the spatial distance between P and Q, this $d(P,Q)$ cannot represent how different they are. Therefore, more reliable distance metric is necessary.

8.4.1 Hausdorff Distance

In many applications, not all points from P need to have a corresponding point in Q, due to occlusion. Typically, the two point sets are of different size, so that no one-to-one correspondence exists between all points. In that case, a dissimilarity measure that is often used is the *Hausdorff distance*. Given two finite points sets $P = \{p_1, \ldots, p_i\}$ and $Q = \{q_1 \ldots, q_j\}$, the Hausdorff distance is defined as:

$$H(P,Q) = \max(h(P,Q), h(Q,P)) \tag{8.7}$$

where

$$h(P,Q) = \max_{p \in P} \min_{q \in Q} \parallel p - q \parallel$$

and $\parallel \cdot \parallel$ is some underlying norm on the points of P and Q (e.g., the L_2 or Euclidean norm).

The function $h(P,Q)$ is called the *directed* Hausdorff distance from P to Q (this function is not symmetric and thus is not a true distance). It identifies the point $p \in P$ that is farthest from any point of Q, and measures the distance from p to its nearest neighbor in Q (using the given norm $\parallel \cdot \parallel$). That is, $h(P,Q)$ in effect ranks each point of P based on its distance to the nearest point of Q, and then uses the largest ranked such point as the distance (the most mismatched point of P). Intuitively, if $h(P,Q) = d$, then every point of P must be within a distance d of some point of Q, and there also is some point of P that is exactly distance d from the nearest point of Q (the most mismatched point) and vice versa.

The Hausdorff distance, $H(P,Q)$, is the maximum of $h(P,Q)$ and $h(Q,P)$. Thus it measures the degree of mismatch between two sets, by measuring the distance of the point of p that is farthest from any point of Q and vice versa. Intuitively, if the Hausdorff distance is d, then every point of P must be within a distance d of some point of Q and vice versa. Thus the notion of resemblance encoded by this distance is that each member of P be near some member of Q and vice versa. Unlike most methods of comparing shapes, there is no explicit pairing of points of P with points of Q (for example many points of P may be close to the same point of Q).

8.4.2 An Algorithm for the Hausdorff Distance of Two Polygons

Let P, Q be two polygons with p, q vertices, respectively. To determine the Hausdorff distance between P and Q, we consider the *Voronoi diagram* of P, $\mathrm{Vor}(P)$.

Vor(P) assigns to each edge and each vertex of P its Voronoi-cell, i.e. the set of points in the plane which are closer to this element (i.e. edge or vertex) than to any other one. The edges of Vor(P) are either line segments (if they separate the cells of two edges or two vertices of P) or parabolic segments (if they separate the cell of a vertex from the cell of an edge). Vor(P) has $O(p)$ edges and vertices and can be constructed in time $O(p \log p)$. In order to obtain a finite problem we observe the following:

Consider the intersection of a Voronoi-cell Vor(P) with Q. Suppose that we move monotonically on an edge of Q within this Voronoi-cell Vor(P). As easily can be seen the distance to the corresponding element of P defining cell Vor(P) first decreases and then increases monotonically (or is just monotone increasing or just monotone decreasing). It follows that the maximal distance of a point of Q on this edge to P must be assumed at the end points of the edge or at the intersection point with some Voronoi edge bounding cell Vor(P) (see Figure 8.7 for an illustration).

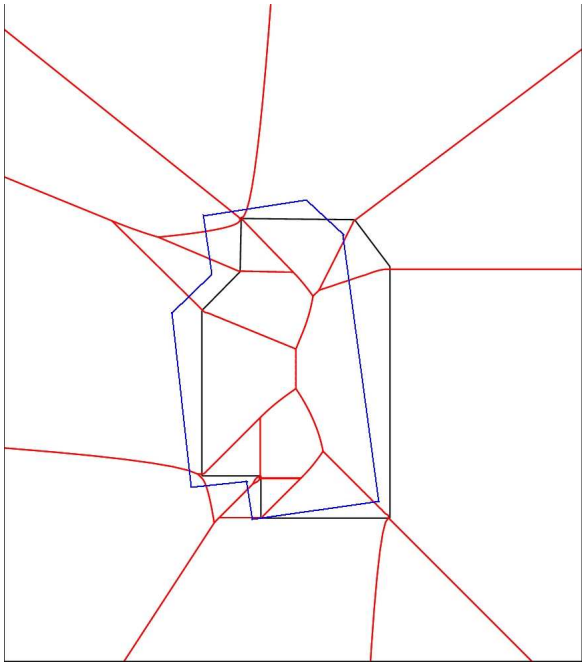


Figure 8.7: Intersection of Voronoi-cell Vor(P) with Q. P is colored black; its Voronoi edges are red. Q is colored blue.

It follows that the distance $h(Q, P)$ must be assumed at a vertex of Q or an intersection point of an edge of Q with a Voronoi-edge of P. Furthermore if we move monotonically on a Voronoi-edge e of P, the distance to the elements whose cells are separated by this edge is described before. Summarizing we have

Lemma 4 *The distance of Q to P, $h(Q, P)$ is assumed either at some vertex of Q or at some intersection point of Q with some Voronoi-edge e of P having either the smallest or largest x-coordinate among the intersection points of Q with e.*

Notice that the number of points in Lemma 4 is $O(p + q)$. It remains to show how to find these points and their nearest neighbors on P, that is we have to determine the cells of Vor(P) containing the vertices of Q and the elements of P closest to the critical intersection points.

We do this by a plane sweep across the arrangement obtained by the possibly split edges of Vor(P) and Q. In order to obtain only the extreme intersection points of each edge e of Vor(P), we delete e from the data structure (e.g. 2-3 tree) as soon as the first intersection point with Q has been found. Two such sweeps, one from left to right and one from right to left are necessary. Since there are $O(p+q)$ event points we obtain an $O((p+g)\log(p+q))$ algorithm for determining all candidates in the sense of Lemma 4. By determining their distance to P and taking their maximum, we get $h(Q,P)$. Analogously, $h(P,Q)$ and thus $H(P,Q)$ can be determined.

8.5 Minutiae Polygons Matching

The purpose of fingerprint minutiae polygons matching is to find the minimum *average Hausdorff distance* among all possible matched fingerprint pairs. After fingerprint distortion correction, all minutiae and minutiae polygons of input fingerprint are mapped to a new position on the template fingerprint. If two polygons intersect with same minutia polygon case and similar orientation, they are defined as *paired*. For each paired polygons, calculate the Hausdorff distance $H_k(P_i, Q_j)$. We give out the matching score:

$$M_1 = \frac{2m_{paired}}{n_P + n_Q}, \qquad M_2 = \frac{\sum_{k=1}^{m_{paired}} H_k(P_i, Q_j)}{m_{paired}} \qquad i = 1, \ldots, n; j = 1, \ldots, m. \quad (8.8)$$

where m_{paired}, n_P and n_Q denote the number of paired minutiae polygons, minutiae in the overlap area of input and template fingerprints, respectively.

8.6 experimental Results

We evaluated our method by testing it on Database FVC2000 which consists of 880 fingerprints, 8 prints of each of 110 distinct fingers. In addition, we also scanned 80 fingerprints with heavy distortion from FUJITSU Fingerprint Sensor (model: FS-210u), 4 prints of each of 20 distinct fingers.

Due to the lack of benchmark of minutiae matching performance, we compared our matching method with the TPS-based algorithm in [5] on two aspects: computing time and accuracy of identification. In our method, $\epsilon = 3$ and $\lambda = 5$ were used to generate minutia polygons; $\mu = 0.2$ was employed for MQ basis function. For TPS-based algorithm, a tolerance box with radius $r = 5$ was used. Since TPS-based algorithm uses three local structures for each minutia in alignment step, iteratively registers minutiae between two fingerprints, they lead to matching time consuming and a higher risk of matching two similar fingerprints coming from distinct fingers. On the contrary, our method uses triangle edges for alignment, number of comparison is less. Moreover, parameters tailor MQ basis function more suitable for elastic distortion and minutiae polygon works as tolerance box with higher capability of tolerating distortion. These two advantages make our matching does not need iteration on registering minutiae. Therefore, our method costs $49ms$ for one matching rather than $107ms$ of TPS-based algorithm on average. So, computing time is faster around one time. The matching performance (ROC) curves plotting the false reject

rate (FRR) against the false accept rate (FAR) at various thresholds is presented in Fig. 8.8. For the data in FVC2000, two methods has almost same accuracy owing to these fingerprints with slightly elastic distortion. But for the scanned fingerprints with heavy distortion, our method is more accurate. We believe that this is due to minutiae polygon has a higher capability of tolerating distortion and MQ basis function can be tailored by parameters for this particular problem. To sum up, our matching method using minutiae polygons not only provides computational efficiency, but also leads to better performance in matching.

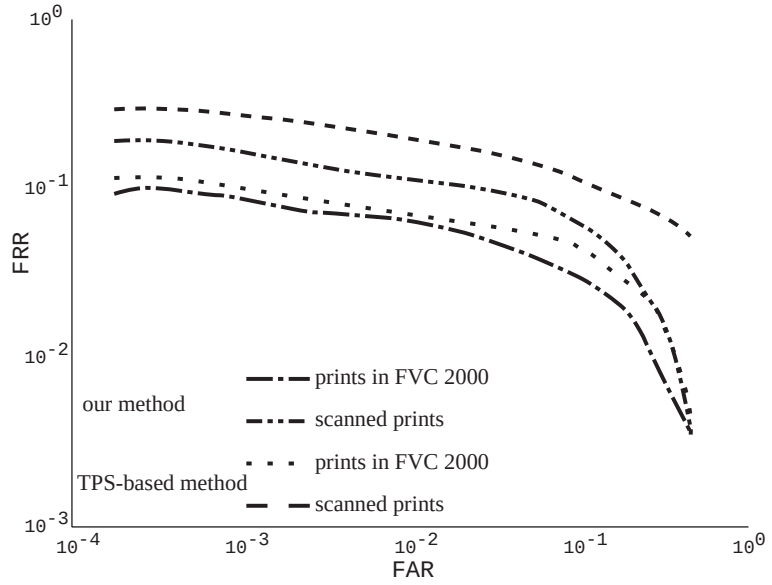


Figure 8.8: ROC curves on FVC2000 DB and scanned fingerprints obtained with the proposed method and the algorithm in [5].

8.7 Conclusion

We proposed a novel minutiae matching algorithm which includes the detail of minutiae information not only the type and orientation. Conventional minutiae matching algorithms define the tolerance box simply and experientially. Hence, matching result always heavily depends on the pre-defined size and shape of tolerance box. Our approach gathers not only type, orientation but angle between two branches (for bifurcation) into a simple minutiae polygon. An efficient algorithm is also designed aiming at polygon matching. One significant advantage of minutia polygon is that it is adaptive and does not depend on the size in an appropriate range. In other words, minutia polygon can be enlarged or shrunk by a reasonable scale for matching without losing accuracy. Experiment result shows that fingerprints can be well matched using our algorithm based on minutia polygon.

Chapter 9

Conclusions

We finish the dissertation with a summary of contributions of our work and some discussions for future research.

9.1 Contributions

In this thesis, we have developed new combinatorial techniques using computational geometric algorithms to improve Automatic Fingerprint Identification Systems. Two important issues, which obstruct AFIS becoming more efficient, have been addressed. They are speeding up preprocessing of AFIS and coping with fingerprint distortion.

It is known that the preprocessing of AFIS consumes almost $90 \sim 95\%$ of the total time. Because most proposed methods only concentrate on a certain step of AFIS, therefore, a whole AFIS have to apply various techniques in each step. No more helpful information, which can accelerate system, is inherited from previous step. It causes identification procedure, especially the preprocessing, to take a long time.

To address speeding up processing, a crucial idea of our work is a linear time Euclidean distance transform (EDT). The same feature of Euclidean distance transform can be used for binarization, denoising, minutiae extraction and matching. An EDT matrix is obtained in binarization. From then on, each step of preprocessing employs the same EDT matrix as basis of current step's method. Thanks to the EDT used as a measure of fingerprint ridge/valley width, preprocessing steps can directly realize their own methods on it. This strategy in real application can save a lot of time. Experiment results prove our methods can save $20 \sim 30\%$ of computing time; and they are also adequately good for further processing.

As using EDT as a basis technique through preprocessing of AFIS, we designed novel algorithms for each step. They differ greatly from conventional methods and hold their own advantages summarized as follows.

- In Chapter 4, we proposed a combinatorial algorithm for binarization of fingerprint images using a linear time EDT algorithm. Most of the previous algorithms are heuristics in that they do not start with a definition of an optimal threshold. In contrast, we define a condition for an optimal threshold based on equal widths of

ridges and valleys. Another significant advantage of our algorithm is that binarization and fingerprint area segmentation can be done simultaneously.

- In Chapter 5, we first proposed a simple and efficient (*linear time*) mathematical morphology method to eliminate impulsive noise, which can be restricted to a minimum number of pixels. Besides, we provided an automatic approach without any experiential parameter for choosing appropriate structuring elements to eliminate useless components by using EDT.

- Most of the minutiae detection algorithms find out the minutiae from the one-pixel thick ridge image obtained from thinning. In Chapter 6, we developed a new way for detecting minutiae straightaway from the thick ridges of the binary image without thinning! The most notable distinction of our work is that it uses fewest arbitrarily selected parameters.

To cope with fingerprint distortion, we in-depth investigated property of finger tips; and then proposed a powerful distortion model and a novel minutiae polygon matching algorithm. Matching results show our method can correct around 70% of a horrible distortion compared with a common method. They prove this model has a high capability to cope with fingerprint distortion, even can catch up with some manual distortion model. Following are their advantages:

- In Chapter 7, we proposed an combined radial basis function model, which separately builds rigid and nonrigid transformations, for attacking fingerprint distortion problem. This model improves around 16% of accuracy than a normal RBF model.

- In Chapter 8, we designed a novel fingerprint minutiae polygon matching algorithm. Minutiae polygon includes more information of minutia than conventional tolerance boxes. Moreover, it is adaptive and does not depend on the size in an appropriate range.

9.2 Future Research Diections

We achieved a lot of work on reducing preprocessing time and improving accuracy of distorted fingerprints matching. To make system "intelligently" identify distorted fingerprints using fewest passible parameters, following issues will be challenging areas of future research.

- The definition of the optimal threshold used for binarization algorithm has a drawback in realistic terms. During the acquisition of fingerprints, ridges, being the elevated structures on the finger, exert more pressure on the device making the acquisition. And as such, the widths of the ridges should be greater than the width of the valley for a more realistic model. But, still the lemma 1 will hold and the algorithm instead of trying to find the crossover point of sum total of $SumDT_{1,0}$ and $SumDT_{0,1}$ will terminate when $SumDT_{1,0}$ is greater than $SumDT_{0,1}$ by a certain ϵ. Determining this ϵ from real fingerprint images is a future problem we would like to address.

- Our alignment algorithm for fingerprint distortion cannot guarantee control points are uniformly distributed in fingerprint image. This problem, occasionally, causes our combined RBF model correct local distortions unsatisfactorily. Designing an efficient alignment algorithm with more invariants could contribute to a better distortion correction. We are going to include the minutiae polygon to achieve above purpose.

- Time complexity of computing Hausdorff distance between two minutiae polygons matching is $O(n \log n)$ owing to minutiae polygon is a simple polygon. In practice, it is a bit long for a real time system. If a convex polygon can also represent minutiae detail, time complexity of our algorithm will decrease to linear. Our next step will investigate the possibility of this idea.

- In addition, as iris recognition problem has some common characteristics with fingerprint recognition, we are trying to extend our proposed methods to iris recognition.

Bibliography

[1] G. Agam and Its'hak Dinstein, "Generalized Morphological Operators Applied to Map-Analysis", Proc. of SSPE'96, pp. 60-69 (1996).

[2] G. Anelli, A. Broggi and G. Destri : "Decomposition of Arbitrarily Shaped Binary Morphological Structuring Elements Using Genetic Algorithms", IEEE Trans. on PAMI, vol. 20, no. 2, pp. 217–224 (1998).

[3] N. Arad and D. Reisfeld : "Image warping using few anchor points and radial functions", Computer Graphics Forum, vol. 14, no. 1, pp. 35–46 (1995).

[4] C. Arcelli and G.S.D Baja : "A Width Independent Fast Thinning Algorithm", IEEE Trans. on PAMI, vol. 4, no. 7, pp. 463–474 (1984).

[5] A.M. Bazen and S.H. Gerez : "Elastic mniutiae matching by means of thin-plate spline models", Proc. of ICPR, vol. 2, pp. 985–988 (2002).

[6] G. Bebis, M. Georgiopoulos, M. Shah and N. La VitoriaLobo : "Indexing Based on Algebraic Functions of Views",Computer Vision and Image Understanding, vol. 72, no.3, pp. 360–378 (1998).

[7] M.de Berg, M.van Kreveld, M.Overmars and O.Schwarzkopf : "Computational Geometry (Algorithms and Application)", Springer-Verlag Berlin Heidelbarg, Chapter 9 (1997).

[8] I. Bitter, A.E. Kaufman and M. Sato : "Penalized-distance volumetric skeleton algorithm", IEEE Trans. on Visualization and Computer Graphics, vol. 7, no. 3, pp. 195–206 (2001)

[9] J.L. Blue, G.T. Candela,P.J. Grother, R. Chellappa, C.L. Wilson and J.D. Blue, : "Evaluation of Pattern Classifiers for Fingerprint and OCR Application", Pattern Recognition, vol. 27, no. 4, pp. 485–501 (1994).

[10] J.W. Brandt and V.R. Algazi : "Continuous skeleton computation by Voronoi diagram", CVGIP: Image Understanding, vol. 55, no. 3, pp. 329–338 (1992).

[11] H. Breu, J. Gil, D. Kirkpatrick and M. Werman : "Linear Time Euclidean Distance Transform Algorithms", IEEE Trans. on PAMI, vol. 17, no. 5, pp. 529–533 (1995).

[12] A. Califano and R. Mohan : "Multidimensional Indexing for Recognizing Visual Shapes", IEEE Trans. on PAMI, vol. 16, no. 4, pp. 373–392 (1994).

[13] G.T. Candela, P.J. Grother, C.I. Watson, R.A. Wilkinson and C.L. Wilson : "PCASYS - A Pattern-Level Classification Automation System for Fingerprints", NISTIR 5647, National Institute of Standards and Technology (Aug. 1995).

[14] R. Cappelli, D. Maio and D. Maltoni : "Modelling plastic distortion in fingerprint images", Proc. of CAPR, pp. 369–376 (2001).

[15] J.H. Chang and K.C. Fan : "Fingerprint Ridge Allocation in Direct Gray-Scale Domain", Pattern Recognition, vol. 34, no. 10, pp. 1907–1925 (2001).

[16] D. Clemens and D. Jacobs : "Space and Time Bounds on Indexing 3D Models from 2D Images", IEEE Trans. on PAMI, vol. 13, no. 10, pp.1007–1017 (1991).

[17] L. Coetzee and E.C. Botha : "Fingerprint Recognition in Low Quality Images", Pattern Recognition, vol. 26, no. 10, pp. 1441–1460 (1993).

[18] M.W. Colins : "Realizing the Full Value of Latent Prints", California Identification Digest (1992).

[19] R. Creutzburg and J. Takala : "Optimising Euclidean Distance Transform Values by Number Theoretic Methods", IEEE Nordic Signal Processing Symposium, pp. 199–203 (2000).

[20] F.Crow : "Summed-area tables for texture mapping", Proc. of SIGGRAPH, 18(3), pp. 207–212 (1984).

[21] O. Cuisenaire and B. Macq : "Fast Euclidean Distance Transformation by Propagation using Multiple Neighbourhoods", Computer Vision and Image Understanding, vol. 76, pp. 163–172 (1999).

[22] O. Cuisenaire and B. Macq : "Fast Euclidean morphological operators using local distance transformation by propagation, and applications", Internat. Symp. Pattern Recog., (1999).

[23] P.E. Danielsson : "Euclidean Distance Mapping", Computer Graphics and Image Processing, vol. 14, pp. 227–248 (1980).

[24] "Automated Classification System Reader Project (ACS)", Technical Report, DeLaRue Printrak Inc. (Feb. 1985).

[25] C. Dorai, N.K. Ratha and R.M. Bolle : "Detectng dynamic behavior in compressed fingerprint videos: Distortion", Proc. of CVPR, , vol.2, pp. 320–326 (2000).

[26] D.C. Douglas Hung : "Enhancement and Feature Purification of Fingerprint Images", Pattern Recognition, vol. 26, no. 11, pp. 1661–1771 (1993).

[27] H. Eggers : "Two Fast Euclidean Distance Transformations in Z2 Based on Sufficient Propagation", Computer Vision and Image Understanding, vol. 69, pp. 106–116 (1998).

[28] A. Farina, Z.M. Kovács-Vajna and A. Leone : "Fingerprint Minutiae Extraction from Skeletonized Binary Images", Pattern Recognition, vol. 32, pp. 877–889 (1999).

[29] S. Fejes, F. Vajda : "Simplified Adaptive Approach to Efficient Morphological Image Analysis", Proc. Int. Conf. on Pattern Recogntion D: Parallel Computing, vol. 94, pp. 257–261 (1994).

[30] S. Fejes, F. Vajda : "Efficient implementation technique of adaptive morphological operations", Mathematical Morphology and its Applications to Signal Processing II, ed. J.Serra and P.Soille, pp. 273–280, Kluwer Academic Publishers, The Netherlands (1994).

[31] S. Fejes, F. Vajda : "A data-driven algorithm and systolic architecture for image morphology", Proc. Int. Conf. on Image Processing, vol. 2, pp. 550–554 (1994).

[32] M. Fornefett, K. Rohr and H.S. Stiehl : "Elastic registration of medical images using radial basis functions with compact support", Proc. of CVPR, vol. 1, pp. 402–407 (1999).

[33] Fingerprint Verification Competition (2000), http://bias.csr.unibo.it/fvc2000/download.asp.

[34] F. Galton : "Fingerprints", London: Macmillan (1892).

[35] R. Germain, A. Califano and S. Colville : "Fingerprintmatching Using Transformation Parameter Clustering", IEEE Computational Science and Engineering, pp. 42-49 (Oct.-Nov. 1997).

[36] C. Gold and J. Snoeyink : "A One-Step Crust and Skeleton Extraction Algorithm", Algorithmica vol. 30, no. 2, pp. 144–163 (2001).

[37] L. O'Gorman and J.V. Nickerson : "An Approach to Fingerprint Filter Design", Pattern Recognition, vol. 22, pp. 29–38 (1989).

[38] C.J. Hilditch : Linear skeletons from square cupboards. Machine Intelligence, 4, (Meltzer B., Mitchie D., Eds.), Edinburgh University Press, pp. 403–420 (1969).

[39] T. Hirata and T. Katoh : "An Algorithm for Euclidean distance transformation", SIGAL Technical Report of IPS of Japan, 94-AL-41-4, pp. 25–31, (Sept. 1994).

[40] J. Hollingum : "Automated Fingerprint Analysis Offers Fast Verification", Sensor Review, vol. 12, no. 13, pp. 12–15 (1992).

[41] C.T. Huang and O.R. Mitchell : "A Euclidean Distance Transform Using Grayscale Morphology Decomposition", IEEE Trans. on PAMI, vol. 16, no. 4, pp. 443–448 (1994).

[42] S. Huvanandana, C. Kim and J.N. Hwang : "Reliable and Fast Fingerprint Identification for Security Applications ", Proc. Int. Conf. on Image Processing, vol. 2, pp. 503–506 (2000).

[43] S. Huvanandana, S, Malisuwan, J, Santiyanon and J.N. Hwang : "A Hybrid System for Automatic Fingerprint Identification", Proc. Int. Sym. on Circuits and Systems, vol. 2, pp. 952–955 (2003).

[44] A. Imiya, M. Saito and K. Nakamura : "Thinning by Curvature Flow", Proc. of IWCIA 2004, pp. 432–442 (2004).

[45] N. Ikeda et al. : "Fingerprint Image Enhancement by Pixel-Parallel Processing", Proc. of Int. Conf. on pattern recognition (16th), vol. 3, pp. 752–755 (2002).

[46] D. Jacobs : "Robust and Efficient Detection of Convex Groups", IEEE Trans. on PAMI, vol. 18, no. 1, pp. 23–37 (1996).

[47] A.K. Jain, L. Hong and R.M. Bolle : "On-line fingerprint verification", IEEE Trans. on PAMI, vol. 19, no. 4, pp. 302–313 (1997).

[48] A.K. Jain, L. Hong, S. Pankanti and R. Bolle : "An Identity-Authentication System Using Fingerprints", Proc. of IEEE, vol. 85, no. 9, pp. 1365–1388 (1997).

[49] A.K. Jain and S. Pankanti : "Automated Fingerprint Identification and Imaging Systems", Advances in Fingerprint Technology, 2nd Edition, Elsevier Science (2001).

[50] S. Jung, R. Thewes, T. Scheiter, KF Goser and W. Weber : "A low-power and high-performance CMOS fingerprint sensing and encoding architecture", IEEE Journal Solid-State Circuits, vol.34, no. 7, pp. 978–984 (1999).

[51] A. Kaul, M.A. O'Connor and V. Srinivasan : "Computing minkowski sums of regular polygons", Proc. of the 3rd Canad. Conf. Comput. Geom., pp. 74–77 (1991).

[52] T. Kikuchi, S. Murakami : "Application of Fuzzy Mathematical Morphology with Adaptive Structuring Elements to Seal Defect Testing", Journal of Advanced Computational Intelligence, vol. 6, no. 1, pp. 62–69 (2002).

[53] Z.M. Kovács-Vajna : "A Fingerprint Verification System Based on Triangular Matching and Dynamic Time Warping", IEEE Trans. on PAMI, vol. 22, no. 11, pp. 1266–1276 (2000).

[54] Y. Lamdan, J. Schwartz and H. Wolfson : "Affine Invariant Model-based Object Recognition", IEEE Trans. on Robotics and Automation, vol. 6, no. 5, pp. 578–589 (Oct. 1990).

[55] B. Lamiroy and P. Gros : "Rapid object indexing and recog-nition using enhanced geometric hashing", EuropeanConference on Computer Vision (ECCV), pp. 59–70 (1996).

[56] H.C. Lee and R.E. Gaensslen : "Advances in Fingerprint Technology", 2nd edition, Elsevier, New York (2001).

[57] G. Levi and F. Sirovich : "Structure Description of Fingerprint Images", Information Sciences, pp.327–355 (1972).

[58] Z. Liposcak, S. Loncaric : "Face Recognition from Profiles Using Morphological Signature Transform", Proc. of The 21st Int. Conf. on Information Technology Interfaces, pp. 93–98 (1999).

[59] S. Loncaric, A.P. Dhawan : "Optimal Shape Description using Morphological Signature Transform via Genetic Algorithm", SPIE Proceedings 2030: Image Algebra and Morphological Image Processing, pp. 121–127 (1993).

[60] X. Luo, J. Tian and Y. Wu : "A minutia matching algorithm in fingerprint verification", Proc. of ICPR, vol. 4, pp. 833–836 (2000).

[61] D. Maio and D. Maltoni : "Direct Gray-Scale Minutiae Detection In Fingerprints", IEEE Trans. on PAMI, vol. 19, no. 1, pp. 27–39 (1997).

[62] D. Maltoni, D. Maio, A.K. Jain and S. Prabhakar : "Handbook of Fingerprint Recognition", Springer-Verlag, New York (2003).

[63] B.M. Mehtre and B. Chatterjee : "Segmentation of Fingerprint Images - A Composite Method", Pattern Recognition, vol. 22, pp. 381–385 (1989).

[64] B. Moayer and K. Fu : "A Tree System Approach for Fingerprint Pattern Recognition", IEEE Trans. on PAMI, vol. 8, no. 3, pp. 376–388 (1986).

[65] A. Moenssens : "Fingerprint Tchniques", Chilton, London (1971).

[66] I. Nystrom, G.D. Sanniti and S. Svensson : "Curve Skeletonization by Junction Detection", Lecture Notes in Computer Science, 2059, pp. 229–238 (2001).

[67] K. Qian, S. Cao, P. Bhattacharya : "Gray Image Skeletonization with Hollow Preprocessing Using Distamce Transformation", International Journal of Pattern Recognition and Artificial Intelligence, vol. 13 no. 6, pp. 881–892 (1999).

[68] I. Rangelmam : "The Euclidean Distance Transformation in Arbitrary Dimensions", Pattern Recognition Letters, vol. 14, pp. 883–888 (1993).

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

[70] H.T.F. Rhodes : "Alphonse Bertillon: Father of Scientific Detection", Abelard-Schuman, New York (1956).

[71] A. Rosenfeld and J. Pfaltz, "Sequential Operations in Digital Picture Processing", Journal of the ACM, vol. 13, no. 4, pp. 471–494 (1966).

[72] A. Ross, S.C. Dass and A.K. Jain : "Estimation fingerprint deformation", Proc. of ICBA, vol.2, pp. 249–255 (2004).

[73] D. Ruprecht and H. Müller : "Image warping with scattered data interpolation", IEEE Computer Graphics and Applications, vol. 15, no. 2, pp. 37–43 (1995).

[74] J.C. Russ, "Image Processing Handbook", 2nd edition, CRC Press, Boca Raton, Florida (1995).

[75] T. Saito and J.I. Toriwaki : "New Algorithms for Euclidean Distance Transformations of an N-dimensional Digitised Picture with Applications", Pattern Recognition, vol. 27, pp. 1551–1565 (1994).

[76] W. Scott : "Fingerprint Mechanics – A Handbook", C. Thomas, Springfield, IL (1951).

[77] A. Senior and R. Bolle : "Improved fingerprint matching by distortion removal", IEICE Trans. on INF. & SYST., vol. E84-D, no. 7, pp. 825–832 (2001).

[78] J. Serra and P. Soille : "Mathematical Morphology and its Applications to Image Processing", Kluwer Academic Publishers, The Netherlands, 1994.

[79] F.Y. Shih and C.C. Pu : "A Skeletonization Algorithm by Maxima Tracking on Euclidean Distance Transform", Pattern Recognition, vol. 28, no. 3, pp. 331–341 (Mar. 1995).

[80] F.Y. Shih and Y.T. Wu : "Fast Euclidean Distance Transformation in 2 Scans Using a 3x3 Neighborhood", Computer Vision and Image Understanding, vol. 93, pp. 109-205 (2004).

[81] S. Shigematsu, H. Morimura, Y. Tanabe and K. Machida : "A Single-Chip Fingerprint Sensor and Identifier", IEEE Journal of Solid-State Circuits, vol. 34, no. 12, pp. 1852–1859 (1999).

[82] S. Skiena : "The Algorithm Design Manual", Springer-Verlag, NY, (1998)

[83] S. Svensson, G. Borgefors and I. Nyström : "On reversible skeletonization using anchor-points from distance transforms". Journal on Visual Communication and Image Representation, vol. 10, Mo. 4, pp. 379–397 (1999).

[84] O. Trier and A.K. Jain : "Goal-Directed Evaluation of Binarization Methods", IEEE Trans. on PAMI, vol. 17, no. 12, pp. 1191–1201 (1995).

[85] M. Tuceryan and T. Chorzempa : "Relative Sensitivity of a Family of Closest-point Graphs in Computer Vision Applications", Pattern Recognition, vol. 24, no. 5, pp. 361–373 (1991).

[86] L. Vincent : "Exact Euclidean distance function by chain propagations", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 520–525 (1991).

[87] P. Viola and M. Jones : "Robust Real-time Face Dectection", International Journal of Computer Vision, vol. 57(2), pp. 137–154 (2004).

[88] A. Wahab, S.H. Chin and E.C. Tan : "Novel Approach to Automated Fingerprint Recognition", IEEE Trans. on PAMI, vol. 145, no. 3, pp. 160–166 (1998).

[89] F.M. Waltz and H.H. Garnaoui : "Fast Computation of the Grassfire Transform Using SKIPSM", SPIE Conf on Machine Vision Applications, Architectures and System Integration III, vol. 2347, pp. 396–407 (1994).

[90] C.I. Watson : "Fingerprint Database, National Institute of Standards and Technology", Special Database 4, FPDB (Apr. 1992).

[91] J.H. Wegstein : "An Automated Fingerprint Identification System", US Government Publication, Washington (1982).

[92] Q. Xiao and H. Raafat : "Fingerprint Image Post-Processing: A Combined Statistical and Structural Approach", Pattern Recognition, vol. 24, no. 10, pp. 985–992 (1991).

[93] T.Y. Zhang and C.Y. Suen : "A fast parallel algorithm for thinning digital patterns", Communications of the ACM, vol. 29, no. 3, pp. 239–242 (1986).

Publications

[1] X.F. Liang, T. Asano : "A Linear Time Algorithm for Binary Fingerprint Image Denoising using Distance Transform", To appear in IEICE Trans. on INF. & SYST..

[2] H. Zhang, T.B. Ho, M.S. Lin, and X.F. Liang : "Feature Extraction for Time Series Classification Using Discriminating Wavelet Coefficients", Accepted by Third International Symposium on Neural Networks (ISNN2006), Chengdu, China (May 2006).

[3] X.F. Liang, K. Kotani, T. Asano : "Automatically Choosing Appropriately-Sized Structuring Elements to Eliminate Useless Components in Fingerprint Image", Proc. Visual Communications and Image Processing 2005 (Proc. Of SPIE, Vol. 5960), pp. 284-293, Beijing, China, (Jul. 2005).

[4] X.F. Liang, A. Bishnu, T. Asano : "A near-linear time algorithm for binarization of fingerprint images using distance transforms", Proc. 10th Intl. Workshop, IWCIA 2004, December, Auckland, Lecture Notes in Computer Science LNCS 3322, edited by R. Klette and J. Zunic, "Combinatorial Image Analysis", pp.197-208 (Dec. 2004).

[5] X.F. Liang, T. Asano : "A fast denoising method for binary fingerprint image", Proc. IASTED Conf. on Visualization, Imaging, and Image Processing, pp. 309-313, Marbella, Spain (Sep. 2004).