Fingerprint Identification - Feature Extraction, Matching, and Database Search

Asker M. Bazen

Final version: August 19, 2002

Contents

Voorwoord

Gedurende mijn promotie heb ik een een aantal jaar kunnen werken aan een volledig zelf vormgegeven onderzoek. In deze periode heb ik erg veel geleerd, zowel vakinhoudelijk als ook daar buiten, maar daarnaast heb ik ook een geweldige tijd gehad. Hierbij wil ik graag van de gelegenheid gebruik maken om de mensen te bedanken die aan het succes van mijn promotie hebben bijgedragen.

Allereerst wil ik mijn promotoren Kees Slump en Otto Herrmann, die deze opdracht mogelijk hebben gemaakt, bedanken. Daarnaast mijn begeleider en assistent-promotor Sabih Gerez. Met hem heb ik altijd lange discussies gevoerd, die er voor hebben gezorgd dat nieuwe ideeën van alle kanten kritisch werden bekeken en nog net iets scherper werden geformuleerd. Verder natuurlijk alle medewerkers en studenten van de leerstoel SAS-NT, die de perfecte (werk-)sfeer hebben gecreëerd.

Ook wil ik de studenten bedanken die door middel van een opdracht hebben meegewerkt aan mijn promotieonderzoek. Gerben Verwaaijen heeft gewerkt aan een correlatie-gebaseerd vingerafdruk herkenningssysteem, Pieter van der Meulen heeft een genetisch programmeer-omgeving ontworpen die Han Schipper heeft toegepast op minutiae extraction, Johan de Boer en Eelke blok hebben gewerkt aan het indexeren van databases, Bart Blaauwendraad aan enhancement, Samuael Jonathan aan het gebruik van commerciele SDKs en Stefan Klein heeft onderzoek gedaan naar segmentatie.

Verder wil ik Martijn van Otterlo bedanken voor de samenwerking op het gebied van re-inforcement learning, Marc Schrijver voor de talloze discussies over allerlei onderwerpen, Raymond Veldhuis voor de ideeën en samenwerking op het gebied van segmentatie en like-lihood ratios, en Mannes Poel en Leo Veelenturf voor hun inbreng bij de toepassing van CI technieken. Daarnaast wil ik Anton Kuip en Harry Kip van NEDAP bedanken voor het delen van hun ervaringen met de toepassing van vingerafdruk herkenning in praktijk situaties, en de praktische problemen die zich in zulk soort situaties voor doen.

Ten slotte wil ik natuurlijk Anja en Elbert bedanken voor hun fantastische steun tijdens deze periode.

Asker Bazen
Enschede, 19 augustus 2002

Chapter 1

Introduction

Recognition of persons on the basis of biometric features is an emerging phenomenon in our society [Jai99b, Zha02]. It has received increasing attention in recent years due to the need for security in a wide range of applications, such as replacement of the *personal identification number* (PIN) in banking and retail business, security of transactions across computer networks, high-security wireless access, televoting, and admission to restricted areas. More examples of applications are given in Section 1.2.

Traditional systems to verify a person's identity are based on knowledge (secret code) or possession (ID card). However, codes can be forgotten or overheard, and ID cards can be lost or stolen, giving impostors the possibility to pass the identity test. The use of features inseparable from a person's body significantly decreases the possibility of fraud. Furthermore biometry can offer user-convenience in many situations, as it replaces cards, keys, and codes.

Many such biometric features can be distinguished: fingerprint, iris, face, voice, hand geometry, retina, handwriting, gait, and more. For several reasons, the fingerprint is considered one of the most practical features. Fingerprints are easily accessible, recognition requires minimal effort on the part of the user, it does not capture information other than strictly necessary for the recognition process (such as race, health, etc.), and provides relatively good performance. Another reason for its popularity is the relatively low price of fingerprint sensors. PC keyboards and smart cards with built-in fingerprint sensors are already available on the market, and the sensors can be integrated easily in wireless hardware.

Even though many academic and commercial systems for fingerprint recognition exist, the large number of publications on this subject shows the need for further research on the subject so as to improve the reliability and performance of the systems. As this chapter will clarify, techniques to process fingerprints for recognition purposes are far from mature in spite of the extensive research already done in this field. The first *Fingerprint Verification Competition* (FVC2000) [Mai02] has shown that many factors may decrease the recognition performance. Noise in the captured fingerprint image, elastic distortion of the skin when pressing the sensor, the partial image of a finger, large fingerprint databases: all these factors make it difficult for fingerprint recognition algorithms to achieve high performance.

For wide application and user-acceptance of fingerprint recognition, improvement of the recognition performance is still necessary. New algorithms may reduce the error rates to levels that are acceptable for application of biometric authentication, and enable the use of low-cost sensors that can be integrated easily in wireless hardware or smart cards. Next, users will accept biometrics as a part of modern society if they have experienced the benefits of reliable and high-quality biometric systems.

This chapter is organized as follows. First, Section 1.1 discusses some basic issues and definitions in fingerprint recognition. Section 1.2 then presents a number of applications and summarizes the experience gained in live situations. Finally, Section 1.3 lists various challenges in fingerprint recognition, and Section 1.4 presents an overview of this thesis.

1.1 Definitions

This section provides an overview of the relevant issues and definitions related to fingerprint recognition. It provides background knowledge for understanding the applications and overview that are presented in the rest of this chapter.

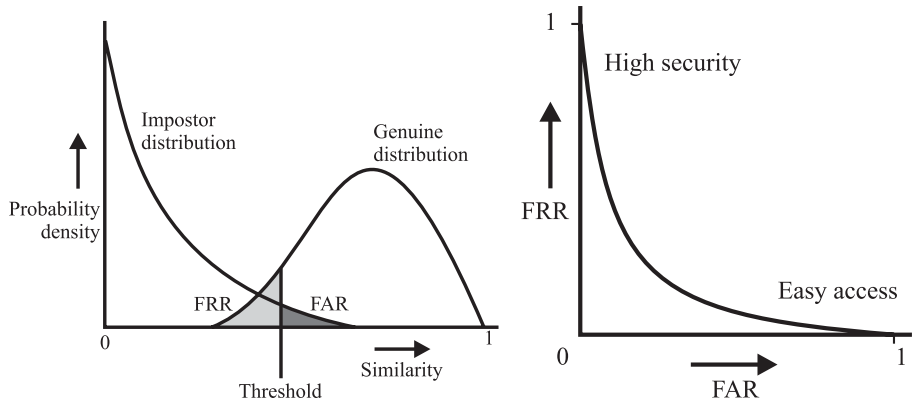
1.1.1 Verification, Identification, and Classification

Several problems, with their associated algorithms and systems, can be defined in the context of fingerprint recognition, being verification, identification, and classification. The term *recognition* is used in a general sense and encompasses all three kinds of tasks. Although these definitions may conflict with the definitions that are used in other research areas, in this thesis I will use the terms that are commonly used in fingerprint recognition.

Verification (or *authentication*) systems use fingerprint technology to verify the claimed identity of a person. Such systems receive two inputs: the claimed identity of the person requesting authentication (usually a username or smart card) and the live-scanned fingerprint of that person. The claimed identity is used to retrieve a *reference* fingerprint stored in a database and is matched (compared) against the currently offered fingerprint (the *test* fingerprint). This results in a measure of similarity, on which the verification decision is based.

Identification systems identify a person based on a fingerprint. Such systems receive only one input, namely the live-scanned *query* fingerprint. A database is searched for a matching fingerprint, which is also referred to as one-to-many matching. A person is identified if a matching fingerprint is found in the database. The system assigns the identity that corresponds to the matching fingerprint to the person that requests identification. On the other hand, if no matching fingerprint is found in the database, the person is rejected. For both verification and identification systems, *enrollment* is an important step. This is the process of taking reference fingerprints of all users and storing these in the database for comparison.

The task of a *Classification* system is to determine which class (or group) the input fingerprint belongs to. These systems also receive only a single fingerprint as input. A well-known set of categories is formed by the *Henry* classes [Hen00], which are discussed in Chapter 10.



(a) Distributions (b) ROC

Figure 1.1: Match and non-match distributions and ROC.

These consist of five classes related to global fingerprint patterns. Classification can be an initial step in an identification task as it reduces the number of database entries to be searched.

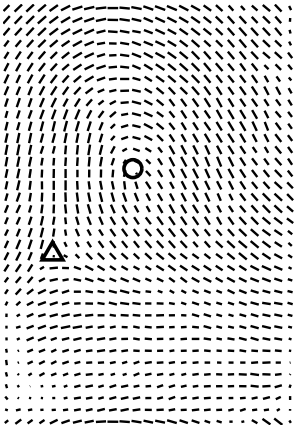
1.1.2 Error Measures

In verification and identification systems, *matching* is an important step. This is the comparison of one particular fingerprint to another one. The result is either a 'match' or a 'non-match'. This is often achieved by assigning a numeric value, corresponding to a measure of similarity, to the result of the comparison, followed by the use of a decision function or threshold to convert this value to a match/non-match decision.

In general, the distributions of the similarity values of genuine attempts (matching fingerprints) and impostor attempts (non-matching fingerprints) cannot be separated completely by a threshold. Instead, the distributions overlap to some extent, resulting in matching errors. This is illustrated in Figure 1.1(a). Two fingerprints are deemed to match if their similarity score exceeds the threshold, while they are deemed to originate from a different finger if their score is below the threshold.

The matching performance of fingerprint verification systems is measured by two error measures [Way99, Mai02]. The *false acceptance rate* (FAR) is the probability that the system outputs 'match' for fingerprints that are not from the same finger, as shown by the dark- gray area in Figure 1.1(a). The *false rejection rate* (FRR) is the probability that the system outputs 'non-match' for fingerprints that originate from the same finger. This is shown by the light-gray area in the figure. Currently available commercial systems perform at approximately $FAR = 10^{-4}$ and $FRR = 10^{-2}$ on high-quality databases. However, for more realistic fingerprint databases, the performance easily drops below $FAR = 10^{-2}$ and $FRR = 5 \cdot 10^{-2}$.

In most systems, the performance can be controlled by a parameter such as the threshold



(a) DF and SPs. The core is indicated by the circle and the delta by the cross

(b) Minutiae

Figure 1.2: Fingerprint with directional field, singular points, and minutiae.

mentioned above. The variation in performance for different parameter settings can be visualized by plotting FAR against FRR. This plot, shown in Figure 1.1(b), is called the *receiver operating curve* (ROC). The threshold can be tuned to meet the requirements of the application for which the system is used. Some systems may require very high security (a very low FAR), while other systems may need to provide easy access (a low FRR). The *equal error rate* (EER) is given by the specific point on the ROC where FAR and FRR are equal.

Achieving low error rates in an identification system is much harder than it is in a verification system. Consider the identification of a fingerprint in a database of $n = 10,000$ entries and an identification system that matches the query fingerprint to all entries in the database. The high-performance matching algorithm with $FAR = 10^{-4}$ will find on average $n \cdot FAR = 1$ false match in this database. Furthermore, the probability of false acceptance over the entire database can be computed as: $FAR_{1:n} = 1 - (1 - FAR)^n = 0.63$. Obviously, this is unacceptable for any identification system.

1.1.3 Fingerprint Features

A fingerprint is a pattern of curving line structures called *ridges*, where the skin has a higher profile than its surroundings, which are called the *valleys*. In most fingerprint images, the ridges are black and the valleys are white.

Due to all kinds of noise and distortions, fingerprints cannot be matched simply by taking the cross-correlation or the Euclidean distance of the gray scale images. This is solved to some extent by extracting features from the fingerprints that are more robust to the distortions. Commonly used features are:

- The *directional field* (DF) is defined as the local orientation of the ridge-valley structures. It describes the coarse structure, or basic shape, of a fingerprint and is calculated on a regular grid in the fingerprint. An example of a DF is given in Figure 1.2(a).
- The *singular points* (SPs) are the discontinuities in the directional field. Two types of SP exist. According to Henry [Hen00], a *core* is the uppermost point of the innermost curving ridge, and a *delta* is a point where three ridge flows meet. In some fingerprints, the SPs fall outside the image area. The SPs are indicated in the DF of Figure 1.2(a).
- The *minutiae* provide the details of the ridge-valley structures. Automatic fingerprint recognition systems use the two elementary types of minutiae that exist, being ridge-endings and bifurcations. Sometimes composite types of minutiae such as lakes or short ridges are also used. In Figure 1.2(b), the minutiae are indicated with small circles.

In most fingerprint recognition systems, the directional field is used for enhancement of the fingerprint and, together with the singular points, for classification, while the minutiae are used for matching.

1.2 Applications

This section presents a number of applications of fingerprint recognition. Since I have not been involved in these applications myself, but discussed them with the actual implementers, these examples are somewhat anecdotal. They are included to give an overview of possible application of fingerprint recognition. Physical access control, computer login and keyless lockers have been implemented by NEDAP[1], while the now defunct company Interstrat performed experiments with biometric smart cards. Various specific aspects and practical experience with fingerprint recognition systems are discussed in this section as well. More applications and implementation issues can for instance be found in [Ash00].

1.2.1 Physical Access Control

Physical access control to buildings or restricted areas is one of the earliest applications of biometric techniques. The goal of this traditional application is to provide high security access control. We will discuss two situations in which fingerprints are used for physical access control.

The first situation is the more traditional setting. A bank in France has introduced fingerprint verification for access by its employees at remote locations. This particular situation involves a small number of employees who use the technology frequently, namely for each time that they require access. The users are highly motivated. They agree with the need for security measurements (the fingerprint recognition system replaces secret access codes), and they accept that secure access may take some more time. Since only a small number of people use the system, considerable time can be spent in training and enrollment.

[1] `http://www.nedap.com/`

The second application is situated in a city in the Netherlands, where the local authorities have decided to use fingerprint verification for access to specific departments of the city hall. In this case, the goal is to provide a secure environment for the maintenance of various kinds of information and files. The motivation is that citizens expect from their government that their information is safe.

For most departments, fingerprints are only required for access outside of working hours. During working hours, there are many employees, and strangers will not be able to obtain access without being noticed. Therefore, an ID card is enough to obtain access during the day. Within the computer department, fingerprint access applies around the clock, since access to the computer systems is much more critical.

Initially during this Dutch city hall experiment, the performance of the fingerprint verification system was somewhat disappointing. Access was refused for reasons that were unclear to the users, and employees did not receive enough help to resolve these problems. However, improved communication and training of operators and users has led to an acceptable situation.

1.2.2 Login to Computer Network

Since access to information requires not only physical presence but also access to computer systems, the city decided to secure its computer network by means of fingerprint login. All computers were equipped with a smart card reader and a fingerprint sensor. For login from different locations at a central network, both an ID card and a fingerprint are required. Furthermore, access via the Internet is also supported so as to enable teleworking at home. Computer access represents a situation that is much more controlled than access to a building. Furthermore, feedback of the image acquisition can be given on the computer screen. Therefore, the performance level of the system is much higher than in the large-scale physical access control experiment.

1.2.3 Key-less Lockers

Many swimming pools have lockers where swimmers can leave their valuables. Traditional lockers require the swimmer to carry the key during swimming. As this is quite inconvenient, the use of a fingerprint to replace the key is an attractive alternative. Furthermore, it would simplify the management of the locker system. Key-less lockers have been tested in several pilot projects. The lockers are installed in a swimming pool as an alternative to key-operated lockers. In a typical setting, a group of 100 lockers are jointly controlled by a computer that is connected to a fingerprint sensor. Figure 1.3 shows a photo of part of such a system.

This application presents several special challenges. First, the recognition of wet fingers gives rise to various problems. Fingerprint images from wet fingers lead to decreased image quality. This problem is enlarged by the fact that the layer of fat at the surface of the skin is reduced by an extended stay in the (chloride-enhanced) water. Various fingerprint sensors were used, but none of them yielded a satisfactory image quality. Also, water causes the skin to fit less tightly around the finger, resulting in above-average elastic distortions of the

Figure 1.3: Key-less lockers.

fingerprint, thereby decreasing the matching performance.

The second challenge is that users should be able to operate the lockers without the constant direct supervision of an operator. After all, correct use of the system is not the main concern of the users: they come to swim and do not want to be bothered with all kinds of procedures. Therefore, the enrollment of the fingerprint is unsupervised and involves no training. The system should give the users feedback on the correct placement of their fingers. This leads to significantly lower image quality than in a situation that involves supervised enrollment.

The final challenge is that the lockers have to be operated by the users without the use of an ID card or other token. This is an identification problem where a database of 100 fingerprints has to be searched reliably. Since no ID cards are required, it is important that the locker system is able to prevent impostor access. The goal is to keep the FAR below 1% when an impostor tries all ten fingers. After a specified number of successive impostor attempts, the system can be blocked and the operator will be alarmed for additional security.

Mainly because of the low image quality and the unsupervised enrollment, the system performance was considerably lower than in experiments involving an office situation. However, an experiment in a swimming pool with a fixed group of users that were enrolled with one-time supervision did yield satisfactory performance.

1.2.4 Biometric Smart Cards

A new application of fingerprint recognition is the biometric smart card that is used in nightclubs to offer visitors more security during their stay. Biometric smart cards have been tested

in a large-scale experiment in 15 nightclubs in the Netherlands. Visitors of the clubs had to purchase a membership card which stores a photograph and a fingerprint. In order to obtain access to the clubs, their face and fingerprint were verified.

A person causing trouble at a nightclub is removed. To offer security to the other visitors, such a person is added to a blacklist and will not have access to the nightclub for a specified period of time. Since all participating nightclubs have a joint blacklist, the person is also denied access to the other clubs.

New users have to be verified against the blacklist. Since suspended users may try to purchase a membership card under a false name, a fingerprint identification system is used. The identification problem is especially difficult since suspended users do not want to be recognized and may try to cause elastic deformation of their fingerprints. To deal with this issue, the FRR must be kept very low.

1.2.5 Practical Experience

This section summarizes the experience that was gained by the application of fingerprint recognition in live situations. Fingerprint recognition is harder to install than face, hand geometry, or iris recognition. The application of such methods is far less sensitive to handling by untrained users and other interfering factors. However, fingerprint recognition can be applied with the very small and cheap sensors that are on the market for this technology.

The performance of a fingerprint verification system highly depends on the situation in which it is used. In the high-security bank situation, employees are well-trained and accept the fact that they have to use the system conscientiously. The system thus performs well. Computer login is also an application in a controlled situation. The login procedure takes some time anyway, and employees are relatively patient as they are sitting at their desk. Therefore, this application does not cause many problems either. However, for physical access control, people may be in a hurry, or they may be cold, wet, or sweaty because of weather conditions. As a result, they may be too impatient for careful acquisition of their fingerprints. On top of that, it takes much more time and organization to train large groups of people. In such situations, the employer must be committed to make the system a success and willing to spend time to explain the system, to train the users, and to offer continuous support.

Enrollment is another critical issue. If the enrolled fingerprints are not of high quality, system performance decreases significantly. Therefore, enough time has to be taken for the enrollment. Especially when the enrollment is unsupervised, feedback of the acquisition process is important. Users need to know whether they have to press harder or to place their fingers differently on the sensor.

Most verification systems use an ID card to store the claimed identity of a user. In this situation, achievement of an extremely low FAR is not a critical issue. Impostor attempts occur only sporadically, since the impostor first has to get access to a valid ID card. If an attempt is made with a stolen ID card that has not been reported as missing yet, the fingerprint verification serves as an additional barrier. In such a case, an FAR level of 1% is satisfactory, while most algorithms offer an FAR of 10^{-3} or better.

Achievement of a low FRR is much bigger problem. If the FRR is too high, many users will not obtain access without knowing why. Although false rejection can also occur with high-quality fingerprints, the main cause of rejection of a fingerprint is its poor image quality (see Figure 5.2 on page 61). The poor-quality fingerprint is rejected before matching with the template, and those cases are not included in the performance figures of the software application. However, users experience such rejection as unjust.

It is generally recognized that a small percentage of the human population has very poor quality fingerprints (typically 1% to 5%) [Pra01]. If those fingerprints are included in the matching performance figures, the distribution of the genuine matching score, which is shown in Figure 1.1(a), will become bimodal. It will show an additional peak near zero, caused by the poor-quality fingerprints. This will result in a heightened FRR, which cannot be resolved by means of threshold settings. The only solution is to increase the image quality by means of better sensors or enhancement algorithms.

A final remark on live biometric systems is that a backup system has to be constructed. A backup system is needed not only when the fingerprint recognition system is out of order, but also to apply to users who are denied access by the system. Backup possibilities may consist of a code that can be used as alternative to the fingerprints, or the physical presence of operator or security staff to provide correct user access.

1.3 Challenges

Analysis of the shortcomings and error types of current fingerprint recognition systems identifies three principal algorithmic challenges in fingerprint recognition that will be addressed in this thesis:

- robust feature extraction from low-quality fingerprints,
- matching fingerprints that are affected by elastic distortions,
- classification methods for efficient search of fingerprints in a database.

Since this thesis focusses on algorithmic aspects of fingerprint recognition, some important problems that fall outside the scope of this thesis. Examples of the issues that will not be addressed are:

- sensor technology,
- detection of faked fingerprints,
- user acceptance.

1.4 Outline

This section presents the general structure of a fingerprint recognition system, shown in the block diagram of Figure 1.4. This thesis presents an exploration of methods and techniques that are encountered in fingerprint recognition. By discussing the various phases of a fingerprint recognition system, an overview of the rest of this thesis is presented. This section ends with a presentation of the benchmarks that are used throughout this theses.

A fingerprint recognition system involves several phases. First, in the *acquisition* phase, the fingerprint is scanned using a fingerprint sensor. Then there is the *feature extraction* phase, which involves calculation of the directional field, enhancement and segmentation of the fingerprint, and extraction of the minutiae. In the *database search* phase, a template fingerprint is retrieved from the database for comparison. In a verification system, a claimed ID is available for retrieval of the template, while in an identification system, the system has to actually search the database, using some form of classification to reduce the search space. Finally, in the *matching* phase, the features of the fingerprint are compared to a template that is found in the database.

1.4.1 Acquisition

The first phase in a fingerprint recognition system is the acquisition of a fingerprint. In the past, fingerprints were obtained by rolling an inked finger from nail to nail on a sheet of paper. Nowadays, however, many sensors are available that capture a fingerprint based on principles in the optical, capacitive, pressure, thermal, or ultrasound domain. They produce a digital image of the fingerprint, typically consisting of 8-bit gray-scale values, scanned at 500 dpi.

Sensors have made the capturing process much more user-friendly since they require only a simple touch of the finger on the sensor and since no ink is involved anymore. However, the task of a fingerprint identification algorithm has become more complicated since the plain touch images (also called dab images) contain a much smaller part of the entire fingerprint. Therefore, fewer minutiae are present, SPs may lie outside of the image area, and two images may overlap for only a very small part. Furthermore, a large amount of elastic deformation may exist between two dab images if force is applied during the acquisition. Finally, algorithms have to be tuned to the specific sensor that is used since different sensors provide images with different characteristics (see Figure 1.5).

Fingerprint mosaicking is a method that artificially constructs an approximation of a rolled image from a series of dab images. Two applications of fingerprint mosaicking are proposed. In [Rat98, Zho01], a finger is rolled over a fingerprint sensor that captures a sequence of images. These images are easily combined into a single larger image. In [Jai02], a method is proposed to construct a composite fingerprint image from multiple impressions that are taken at different moments. In this case, the registration is much more difficult since the registration parameters are not known beforehand, and elastic deformations between the impressions may exist.

The quality and characteristics of the fingerprint image are highly dependent on the exact type of fingerprint sensor that is used. Therefore, the choice of the sensor directly affects the

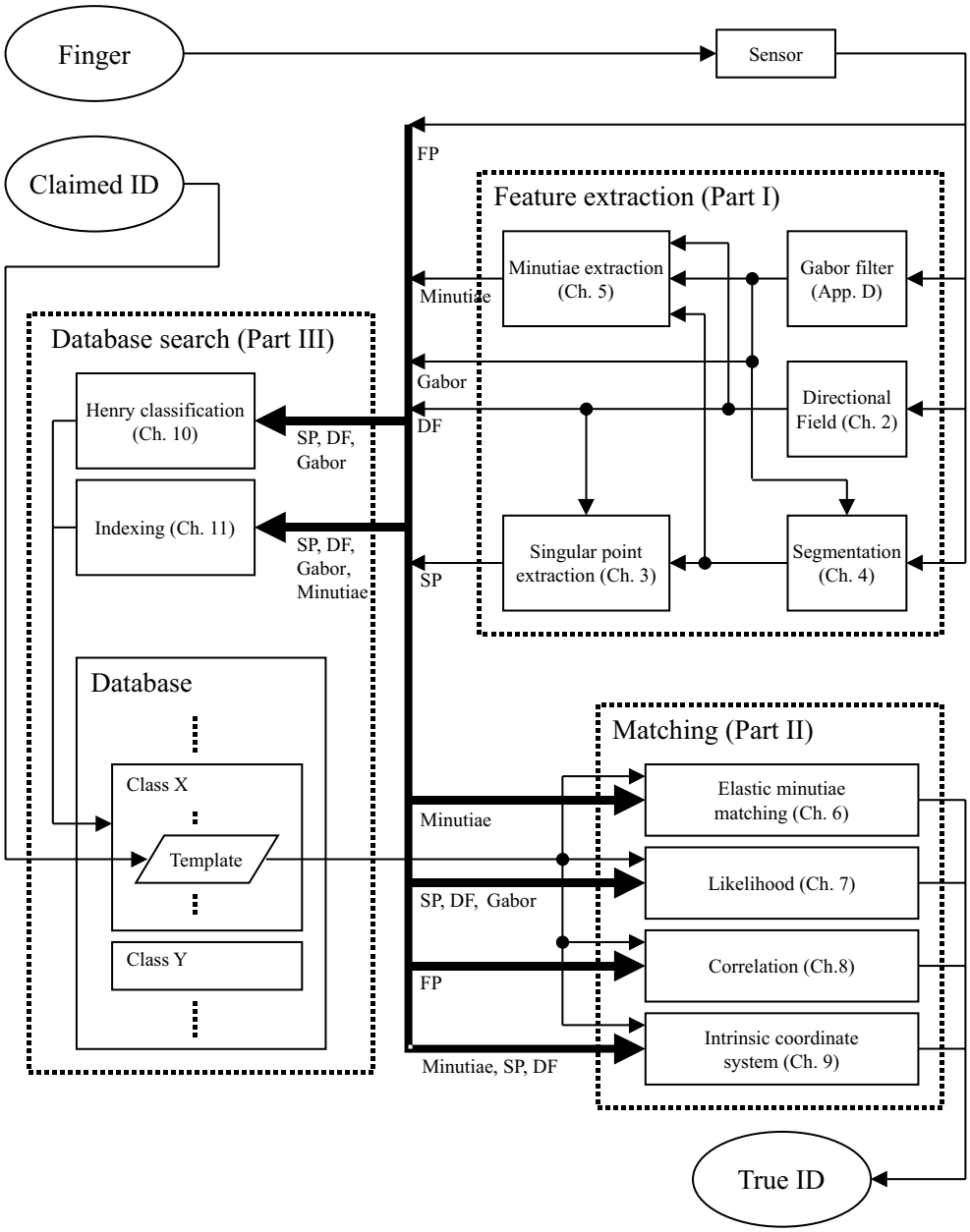


Figure 1.4: Block diagram of a fingerprint recognition system.

(a) Inked (b) Capacitive

(c) Optical (d) Optical

Figure 1.5: Fingerprint images that are acquired by different sensors.

recognition performance, and the recognition algorithms have to be adapted to the specific fingerprint sensor that is used. Fingerprint sensor technology itself is not addressed in this thesis.

1.4.2 Feature Extraction

The next phase in a fingerprint recognition system is the feature extraction from fingerprint images, which is discussed in Part I of this thesis. First, Chapter 2 discusses the estimation of the directional field from fingerprint images and Chapter 3 deals with a method for the extraction of singular points from the directional field. Next, Chapter 4 discusses segmentation of fingerprint images, which is the partitioning of the image in a foreground area that can be used for recognition purposes, possibly a low quality region, and a noisy background area. Finally, Chapter 5 presents minutiae extraction algorithms. Traditional image processing based minutiae extraction is discussed and two alternative learning agent based methods are presented. Each of the subsequent stages in the other parts makes use of some of the features that are presented in Part I.

1.4.3 Matching

Part II deals with matching fingerprints. Four different matching algorithms are discussed, and in practice, one of them has to be chosen. First, Chapter 6 presents an elastic minutiae matching algorithm. This algorithm is able to estimate and reverse elastic deformations in fingerprints by means of thin-plate splines. Next, Chapter 7 proposes a likelihood ratio-based algorithm for matching the directional field and Gabor response (see Appendix D) of fingerprints. Then, Chapter 8 provides an algorithm for matching fingerprints by means of segments from the original gray-scale fingerprint, without any feature extraction. Finally, Chapter 9 presents an intrinsic coordinate system that provides a shape and elastic distortion invariant representation of the minutia locations, but for which a matching algorithm has not been developed yet. Because of performance considerations, only the matching approaches of Chapters 6 and 7 will be used in practice.

1.4.4 Database Search

Part III discusses methods for searching fingerprint databases efficiently. The techniques that are developed in this part are only used in identification system. First, Chapter 10 discusses Henry classification. It is shown that this provides no sufficient solution for searching large databases. Next, Chapter 11 presents indexing methods that provide a much better alternative for this task.

1.4.5 Benchmarks

We have tried as much as possible to use the same databases for benchmarking the performance of all algorithms throughout this thesis. For this task, we have chosen Database 2 of

the fingerprint verification competition 2000 (FVC2000) [Mai00, Mai02].

Before the summer of 2000, there was no common benchmark for assessing the performance of fingerprint recognition systems. Therefore, it was very hard to compare the results that were presented in various papers. Most databases that were used then consisted of scanned images of inked fingerprints. However, the characteristics of these fingerprint images is quite different from the images that are acquired by the digital sensors that are used in automatic fingerprint recognition system. After the summer of 2000, more and more authors switched to the FVC2000 database as benchmark for their algorithms. In August 2002, the FVC2002 database will released, acquired by new state-of-the-art fingerprint sensors.

The FVC2000 database consists of 4 different databases. Three of those are acquired by different fingerprint sensors (1 capacitive and 2 optic sensors) and one is generated synthetically [Cap00a]. Each database contains 880 fingerprints, originating from 110 different fingers of untrained volunteers (they did not receive extensive instructions how to place their fingers etc. to simulate a situation that is encountered in practice) who have all provided 8 prints of the same finger. The database that we use most of the time (Database 2) is captured by a capacitive sensor at 500 dpi, providing 8-bit gray-scale images of 364 by 256 pixels.

At the time of the actual competition, 80 fingerprints (8 prints of 10 fingers) were provided for training purposes before submitting the matching algorithms. The other 800 prints were used for testing. Now, all 880 prints can be used to test a fingerprint recognition algorithm. For one-to-one matching, each fingerprint can be matched against 7 other prints. After removing the double matches, 3080 valid matching experiments remain. For testing the impostor characteristics, usually only the first print of each ID is used. Each fingerprint of the 110 that are available can be matched to 109 other fingerprints, which results in 5995 non-matching experiments after removal of the double matches.

Part I

Feature Extraction

Chapter 2

Directional Field Estimation

Abstract

This chapter discusses the estimation of a high resolution directional field of fingerprints. Traditional methods are compared to a new method, based on principal component analysis. The method is adapted to compute the directional field and its coherence in each pixel. By combining a non-uniform window with efficient filter techniques, the method can also be used to estimate a block directional field with higher accuracy, without being computational much more complex. Parts of this chapter have been published in [Baz00a] and [Baz02d].

(a) Fingerprint (b) Directional field

Figure 2.1: Examples of a fingerprint, and its directional field. The orientation of the small lines represents the estimated direction, while the length of the lines gives an indication of the certainty.

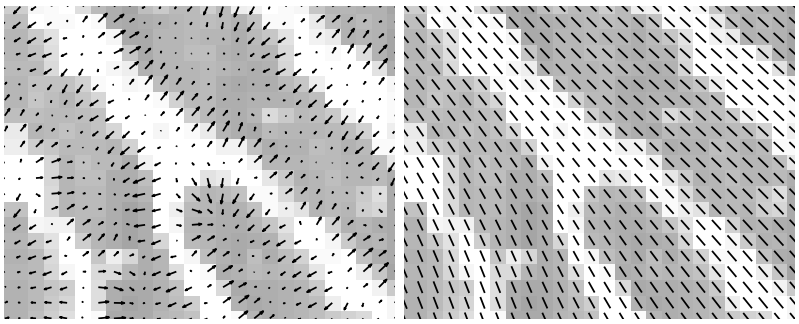
2.1 Introduction

In Figure 2.1(a), a fingerprint is depicted. The information carrying features in a fingerprint are the line structures, called *ridges* and *valleys*. In this figure, the ridges are black and the valleys are white. It is possible to identify two levels of detail in a fingerprint. The *directional field* (DF), shown in Figure 2.1(b), describes the coarse structure, or basic shape, of a fingerprint. It is defined as the local orientation of the ridge-valley structures. The *minutiae* provide the details of the ridge-valley structures, like ridge-endings and bifurcations.

This chapter focuses on the directional field (DF) of fingerprints and matters directly related to the DF. The DF is, in principle, perpendicular to the gradients. However, the gradients are orientations at pixel scale, while the DF describes the orientation of the ridge-valley structures, which is a much coarser scale. Therefore, the DF can be derived from the gradients by performing some *averaging* operation on the gradients, involving pixels in some neighborhood [Lin94]. This is illustrated in Figure 2.2(a), which shows the gradients in a part of a fingerprint (indicated by the small arrows), and Figure 2.2(b), which shows the averaged directional field. While the gradients are not all parallel in this area because of the presence of the endpoint, the directional field is constant in the entire area due to the averaging operator. The averaging of gradients in order to obtain the DF is the topic of this chapter.

The estimation method that is described in this chapter, enables the application of DF-related tasks that require very high resolution and accurate DFs. Examples of these demanding techniques are for instance the accurate “extraction of singular points” as discussed in Chapter 3 and “high performance classification”. Together with the DF, the *coherence* can be estimated. The coherence is a measure that indicates how well the gradients are pointing in the same direction. An example of its use is high resolution segmentation as presented in Chapter 4.

This chapter is organized as follows. First, in Section 2.2, the estimation of the DF is discussed. In Section 2.2.1, the traditional method of averaging squared gradients is discussed,



(a) Gradients (b) Directional field

Figure 2.2: Detailed area in a fingerprint: (a) the gradients, indicated by the small arrows, and (b) the averaged directional field.

while in Section 2.2.2, a new method based on *principal component analysis* (PCA) is proposed. In Appendices A and B, a proof is given that both methods are exactly equivalent and it is shown that the coherence, which is a measure for the local strength of the directional field, can be elegantly expressed in the two eigenvalues that are computed for the PCA. In Section 2.3, some computational aspects of DF estimation are discussed. Furthermore, it is shown that the high-resolution DF can be used to obtain more accurate block-DF estimates. Finally, in Section 2.4, experiments are presented where the theory is applied to fingerprints contained in one of the databases used for the *Fingerprint Verification Competition 2000* [Mai00]. In that section, some practical aspects of the algorithms are discussed as well. Experiments that evaluate the DF extraction by the number of false and missed singular points are presented in Chapter 3.

2.2 Directional Field Estimation

Various methods to estimate the DF from a fingerprint image are known from literature. They include matched-filter approaches [Dre99, Wil94, Kar96], methods based on the high-frequency power in three dimensions [O'G89], 2-dimensional spectral estimation methods [Wil94] and micropatterns that can be considered binary gradients [Kaw84]. These approaches do not provide as much accuracy as gradient-based methods, mainly because of the limited number of fixed possible orientations. This is especially important in case the DF is used for tasks like tracing flow lines. The gradient-based method was introduced in [Kas87] and adopted by many researchers (see e.g. [Rao92, Rat95, Jai97b, Per98]).

The elementary orientations in the image are given by the gradient vector $[G_x(x, y)\ G_y(x, y)]^T$, which is defined as:

$$\begin{bmatrix} G_x(x,y) \\ G_y(x,y) \end{bmatrix} = \text{sign}(G_x)\nabla I(x,y) = \text{sign}(\frac{\partial I(x,y)}{\partial x}) \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix} \tag{2.1}$$

where $I(x,y)$ represents the gray-scale image, and the gradient is calculated by means of a convolution with the derivative of a Gaussian window. The first element of the gradient vector has been chosen to be always positive. The reason for this choice is that in the DF, which is perpendicular to the gradient, opposite directions indicate equivalent orientations. It is illustrated in Figure 2.2 that some averaging operation has to be performed on the gradients, which are distributed over a large range of directions as indicated by the small arrows, in order to obtain the smooth DF.

2.2.1 Averaging Squared Gradients

This section discusses the problems that are encountered with averaging gradients and the traditional solution of averaging squared gradients. First, the general idea behind averaging squared gradients is presented and then, an analysis of the results of this method is given. Apart from the estimation of the DF, this section also discusses the coherence, which provides a measure for the strength or certainty of the estimated orientation.

Intuitive Analysis

Gradients cannot directly be averaged in a local neighborhood, since opposite gradient vectors will then cancel each other, although they indicate the same ridge-valley orientation. This is caused by the fact that local ridge-valley structures remain unchanged when rotated over 180 degrees [Per98]. Since the gradient orientations are distributed in a cyclic space ranging from 0 to π, and the average orientation has to be found, another formulation of this problem is that the 'π-periodic cyclic mean' has to be computed.

In [Kas87], a solution to this problem is proposed by doubling the angles of the gradient vectors before averaging. After doubling the angles, opposite gradient vectors will point in the same direction and therefore will reinforce each other, while perpendicular gradients will cancel. After averaging, the gradient vectors have to be converted back to their single-angle representation. The ridge-valley orientation is then perpendicular to the direction of the average gradient vector.

In the algorithm version discussed in this chapter, not only the angle of the gradients is doubled, but also the length of the gradient vectors is squared. This can also be expressed by considering the gradient vectors as complex numbers that are squared. This has the effect that strong orientations have a higher vote in the average orientation than weaker orientations, equivalent to the L_2 norm. Furthermore, this approach results in the least complex expressions. However, other choices, like for instance setting all lengths to unity [Per98], are found in the literature as well.

In [Kas87], also a method is proposed to use the squared gradients for computation of the strength of the orientation. This measure, which is called the coherence, measures how well

all squared gradient vectors share the same orientation. If they are all parallel to each other, the coherence is 1 and if they are equally distributed over all directions, the coherence is 0.

Formal Analysis

In this section, the intuitive analysis that was given in the previous section is formalized. The gradient vectors are first estimated using Cartesian coordinates, in which a gradient vector is given by $[G_x\ G_y]^T$. For doubling the angle and squaring the length, the gradient vector is converted to "polar" coordinates, in which it is given by $[G_\rho\ G_\varphi]^T$. This conversion is given by:

$$\begin{bmatrix} G_\rho \\ G_\varphi \end{bmatrix} = \begin{bmatrix} \sqrt{G_x^2 + G_y^2} \\ \tan^{-1} G_y/G_x \end{bmatrix} \tag{2.2}$$

Note that $-\frac{1}{2}\pi < G_\varphi \leq \frac{1}{2}\pi$ is a direct consequence of the fact that G_x is always positive. The gradient vector is converted back to its Cartesian representation by:

$$\begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} G_\rho \cos G_\varphi \\ G_\rho \sin G_\varphi \end{bmatrix} \tag{2.3}$$

Using trigonometric identities, an expression for the squared gradient vectors $[G_{s,x}\ G_{s,y}]^T$ that does not refer to G_ρ and G_φ, is found:

$$\begin{bmatrix} G_{s,x} \\ G_{s,y} \end{bmatrix} = \begin{bmatrix} G_\rho^2 \cos 2G_\varphi \\ G_\rho^2 \sin 2G_\varphi \end{bmatrix} = \begin{bmatrix} G_\rho^2(\cos^2 G_\varphi - \sin^2 G_\varphi) \\ G_\rho^2(2 \sin G_\varphi \cos G_\varphi) \end{bmatrix} = \begin{bmatrix} G_x^2 - G_y^2 \\ 2G_xG_y \end{bmatrix} \tag{2.4}$$

This result can also be obtained directly by using the equivalence of 'doubling the angle and squaring the length of a vector' to 'squaring a complex number':

$$G_{s,x} + j \cdot G_{s,y} = (G_x + j \cdot G_y)^2 = (G_x^2 - G_y^2) + j \cdot (2G_xG_y) \tag{2.5}$$

Next, the average squared gradient $[\overline{G_{s,x}}\ \overline{G_{s,y}}]^T$ can be calculated. It is averaged in a local neighborhood, using a not necessary uniform window W:

$$\begin{bmatrix} \overline{G_{s,x}} \\ \overline{G_{s,y}} \end{bmatrix} = \begin{bmatrix} \sum_W G_{s,x} \\ \sum_W G_{s,y} \end{bmatrix} = \begin{bmatrix} \sum_W G_x^2 - G_y^2 \\ \sum_W 2G_xG_y \end{bmatrix} = \begin{bmatrix} G_{xx} - G_{yy} \\ 2G_{xy} \end{bmatrix} \tag{2.6}$$

In this expression,

$$G_{xx} = \sum_W G_x^2 \tag{2.7}$$

$$G_{yy} = \sum_W G_y^2 \tag{2.8}$$

$$G_{xy} = \sum_W G_x G_y \tag{2.9}$$

are estimates for the variances and crosscovariance of G_x and G_y, averaged over the window W. Now, the average gradient direction Φ, with $-\frac{1}{2}\pi < \Phi \leq \frac{1}{2}\pi$, is given by:

$$\Phi = \frac{1}{2}\angle\left(G_{xx} - G_{yy}, 2G_{xy}\right) \tag{2.10}$$

where $\angle(x, y)$ is defined as:

$$\angle(x, y) = \begin{cases} \tan^{-1}(y/x) & x \geq 0 \\ \tan^{-1}(y/x) + \pi \quad \text{for} & x < 0 \wedge y \geq 0 \\ \tan^{-1}(y/x) - \pi & x < 0 \wedge y < 0 \end{cases} \tag{2.11}$$

and the average ridge-valley direction θ, with $-\frac{1}{2}\pi < \theta \leq \frac{1}{2}\pi$, is perpendicular to Φ:

$$\theta = \begin{cases} \Phi + \frac{1}{2}\pi \quad \text{for} & \Phi \leq 0 \\ \Phi - \frac{1}{2}\pi & \Phi > 0 \end{cases} \tag{2.12}$$

An expression for the gradients after squared averaging is given in Appendix A.

The coherence of the squared gradients can also be expressed using the same notations. The coherence *Coh* is given by [Kas87]:

$$Coh = \frac{\left|\sum_W [G_{s,x}\ G_{s,y}]^T\right|}{\sum_W \left|[G_{s,x}\ G_{s,y}]^T\right|} = \frac{\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}}{G_{xx} + G_{yy}} \tag{2.13}$$

as will be shown in Appendix B.

If all squared gradient vectors are pointing in exactly the same direction, the sum of the moduli of the vectors equals the modulus of the sum of the vectors, resulting in a coherence value of 1. On the other hand, if the squared gradient vectors are equally distributed in all directions, the length of the sum of the vectors will equal 0, resulting in a coherence value of 0. In between these two extreme situations, the coherence will vary between 0 and 1, thus providing the required measure.

2.2.2 Principal Component Analysis

This chapter proposes a second method to estimate the directional field from the gradients, which is based on *principal component analysis* (PCA). PCA computes a new orthogonal base given a multi-dimensional data set such that the variance of the projection on one of the axes of this new base is maximal, while the projection on the other one is minimal. It turns out that the base is formed by the eigenvectors of the autocovariance matrix of this data set [The92].

When applying PCA to the autocovariance matrix of the $[G_x\ G_y]^T$ gradient vectors, it provides the 2-dimensional Gaussian joint probability density function of these vectors. The main direction of the gradients can be calculated from this function.

The estimate of the autocovariance matrix **C** of the gradient vector pairs is given by:

$$\mathbf{C} = \begin{bmatrix} G_{xx} & G_{xy} \\ G_{xy} & G_{yy} \end{bmatrix} = \sum_W \begin{bmatrix} G_x^2 & G_x G_y \\ G_x G_y & G_y^2 \end{bmatrix} \tag{2.14}$$

In this estimate, the assumption is made that the gradient vectors are zero-mean, i.e.

$$E[G_x] = E[G_y] = 0 \tag{2.15}$$

in a window W in the given fingerprint. This is true in any window in which the fingerprint has a constant mean gray value. Then, the gradient is defined as the difference of two values that have the same expectation. Therefore, the expectation of the gradient is zero. The requirement of constant mean is reasonable in windows that contain a small number of ridge-valley transitions.

The longest axis $\mathbf{v}_1$ of the 2-dimensional joint probability density function is given by the eigenvector of the autocovariance matrix that belongs to the largest eigenvalue λ_1. This axis corresponds to the direction in which the variance of the gradients is largest, and so to the 'average' gradient orientation. The ridge-valley orientations are perpendicular to this axis, and therefore given by the shortest axis $\mathbf{v}_2$. This is the direction of the eigenvector that belongs to the smallest eigenvalue λ_2. The average ridge-valley orientation θ is given by:

$$\theta = \angle \mathbf{v}_2 \tag{2.16}$$

An expression for $\mathbf{v}_2$ is given in Appendix A.

The "strength" *Str* of the orientation can be defined as a simple function of the two eigenvalues. In order to limit the strength between 0 and 1, it is defined by:

$$Str = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \tag{2.17}$$

Again, if all gradients are pointing in the same direction, $\lambda_2 = 0$ and $Str = 1$, while in case of a uniform distribution over all angles, $\lambda_1 = \lambda_2$ and $Str = 0$.

Both methods appear to be exactly equivalent. In Appendix A a proof is given that the DF estimates are equal by showing that the gradient that is obtained by squared averaging is an eigenvector of autocovariance matrix $\mathbf{C}$. In Appendix B, it is shown that the coherence *Coh*, calculated using the squared gradient method (see Equation 2.13) and the strength *Str* (see Equation 2.17) are exactly equal as well. A similar result has been presented in [Big91].

2.3 Computational Aspects

For efficient calculation of the DF and the coherence in each pixel in the fingerprint image, one should not use either of the two basic methods. Instead, first Equations 2.7 to 2.9 are used for estimation of G_{xx}, G_{yy} and G_{xy}, and subsequently Equations A.22 and B.5 are used for calculation of the DF and the coherence. If those are calculated for all pixels in the image, the summations over W reduce to linear filter operations, which can be implemented very efficiently. On a 500 MHz Pentium III computer, an efficient C++ implementation for calculation of the DF and the coherence takes approximately 300 ms of processing time for a fingerprint of 300 by 300 pixels.

For most DF-related tasks, a high resolution estimate that estimates the DF and coherence in each pixel is not needed. Instead, a simple block-directional field (BDF) with blocks of for instance 8×8 pixels provides sufficient spatial resolution. The classical way to estimate a BDF is to partition the image into blocks and estimate G_{xx}, G_{yy} and G_{xy} as the average of the block. Sometimes, overlapping blocks are used for additional noise suppression.

However, this processing method causes artefacts and noise in the DF estimate, which in turn may create false singular points. This problem can be understood better by considering it as a decimation (subsampling) problem, which is known from multi-rate signal processing [Pro92]. Averaging within a block with a uniform window W does not suppress the high-frequency noise that is present in the gradients sufficiently. This results in aliasing artefacts the BDF estimate. There are two causes for the problem: the shape of the averaging filter is uniform and its length is equal to the decimation rate. This can be solved by decoupling the size and the shape of the averaging filter W from the subsampling rate.

We propose the use of an alternative BDF calculation method that is based on the high-resolution DF. In each block, G_{xx}, G_{yy} and G_{xy} are estimated by means of decimation of the high-resolution DF. Scale-space theory tells that averaging with a Gaussian window W minimizes the amount of artefacts that are introduced by subsampling [Lin94]. This will improve the quality of the DF and reduce the number of false singular points considerably.

From multi-rate signal processing, it is known that the filtering and decimation steps can be implemented very efficiently using polyphase filters by interchanging the order of decimation and filtering [Pro92]. The computational complexity of the filtering step in various DF estimation methods is summarized in Table 2.1. Using the optimal calculation scheme, which first separates the filter and then applies a polyphase implementation, the calculation of an 8×8 BDF is predicted to take only 10 ms on a 500 MHz Pentium III. Compared

Table 2.1: Complexity of the filtering step in DF estimation. The size of the fingerprint is N by M pixels, the decimation rate is d, the standard deviation of the Gaussian window is σ and the region of support of the filter ranges from $-n\sigma$ to $n\sigma$.

Method	Complexity
Traditional BDF	NM
Naive HR DF	$4n^2\sigma^2 NM$
Separable HR DF	$4n\sigma NM$
HR BDF (first polyphase, then separation)	$4n\sigma d^{-1} NM$
HR BDF (first separation, then polyphase)	$2n\sigma d^{-1} NM$

to the traditional BDF estimation algorithm, the computational complexity increases with a factor $2n\sigma/d$, where σ is the standard deviation of the Gaussian window, n is the number of standard deviations that is used as support for the window, and d is the decimation rate. For the parameter values that we use ($\sigma = 6$, $n = 3$, and $d = 8$), the cost of the more accurate estimate is a factor 4.5 in computational complexity, independent of the size of the fingerprint.

2.4 Experimental Results

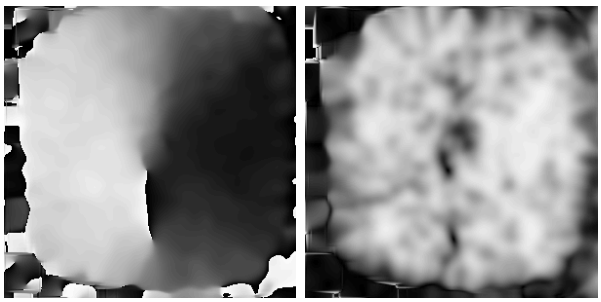
In this section, experiments are presented in which the previously derived results are applied to a number of fingerprints. It will be shown that application of these methods enables the estimation of very accurate and high resolution DFs.

Since there exists no clear ground truth for the DF of fingerprints, objective error measures cannot be constructed. Therefore, it is difficult to evaluate the quality of a DF estimate quantitatively. Alternatively, the quality of a DF estimate can be measured indirectly. This is done in Chapter 3 by counting the number of false and missed singular points.

Most authors process fingerprints blockwise [Kar96, Jai97b]. This means that the directional field is not calculated for all pixels individually. Instead, the average DF is calculated in blocks of, for instance, 16 by 16 pixels. In this section, the processing is carried out pixelwise, leading to a high resolution and accurate DF estimate.

The first experiment considers the fingerprint of Figure 2.1. Although the DF is only shown at discrete steps in Figure 2.1(b), it is estimated for each pixel. This is illustrated in the gray-scale coded Figure 2.3(a). In that figure, the angles in the range of $-\frac{1}{2}\pi$ to $\frac{1}{2}\pi$ have uniformly been mapped to the gray levels from black to white. The figure is somewhat chaotic at the borders, since those are areas that consist of noise. However, as shown in Figure 2.3(b), the coherence is very low in these noisy areas [Baz00a]. In this figure, black indicates $Coh = 0$, while white indicates $Coh = 1$.

Next, an experiment is carried out to illustrate the effects of the choice of the window W. We have chosen a Gaussian window, in accordance with the scale-space theory [Lin94]. In Figure 2.4, the DF in a small segment of 25×20 pixels is shown. This segment contains a broken ridge that is almost horizontal. In this experiment, σ is chosen in the range from



(a) Directional field (b) Coherence

Figure 2.3: Gray-scale coded directional field and coherence. The DF has been mapped linearly from $-\frac{1}{2}\pi$ (black) to $\frac{1}{2}\pi$ (white), while the coherence has been mapped linearly from 0 (black) to 1 (white).

$\sigma = 1$ to $\sigma = 5$. It can be seen that the DF is very erratic for small values of σ. For higher values of σ, the DF becomes more uniform, and the lines get longer, indicating higher coherence values.

From the figure, a window with $\sigma = 5$ seems a good choice, but in more heavily damaged parts of the fingerprints, a window with $\sigma = 6$ is better. For this value, the DF around a broken ridge is sufficiently averaged. The window has then an effective region of support of approximately 25 pixels (2σ on each side), which corresponds to approximately 2 to 3 ridge-valley structures.

2.5 Conclusions

In this chapter, a new PCA-based method to estimate directional fields from fingerprints is proposed. Since it is proven that this method provides exactly the same results as the traditional method, the method offers a different view and an increase of insight into the validity of the traditional solution of estimating an 'average' gradient. It is pointed out that the methods that are presented in this chapter can be used either to estimate a high-resolution DF, or to improve the accuracy of block directional fields, without demanding significantly more processing time.

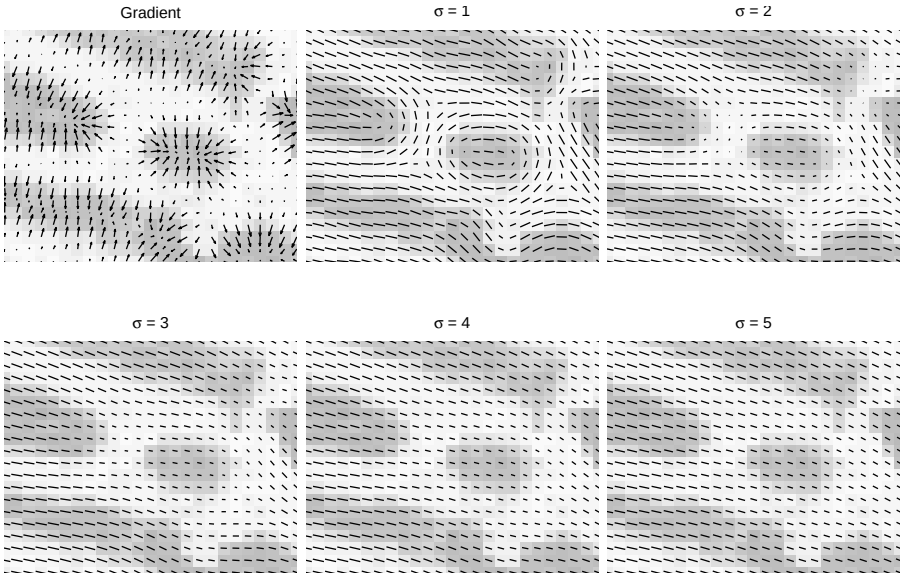


Figure 2.4: Gradients and directional field for various values of σ.

Chapter 3

Singular Point Extraction

Abstract

The subject of this chapter is singular point detection. An efficient algorithm is proposed that extracts singular points from a high-resolution directional field. The algorithm is based on the Poincaré index and provides a consistent binary decision that is not based on post-processing steps like applying a threshold on a continuous resemblance measure for singular points. Furthermore, a method is presented to estimate the orientation of the extracted singular points. The accuracy of the methods is illustrated by experiments on a live-scanned fingerprint database. Parts of this chapter have been published in [Baz01b] and [Baz02d].

(a) Fingerprint (b) Singular points

Figure 3.1: Examples of a fingerprint, and its singular points.

3.1 Introduction

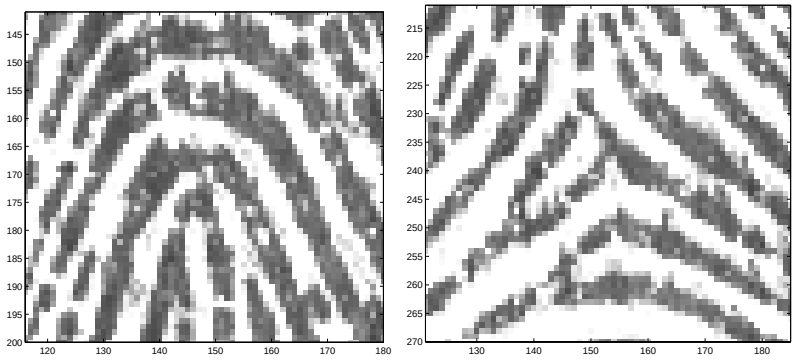
In a fingerprint, *singular points* (SPs) can be identified. The extraction of those singular points is the topic of this chapter. SPs are the points in a fingerprint where the directional field is discontinuous. Henry [Hen00] defined two types of singular points, in terms of the ridge-valley structures. The *core* is the topmost point of the innermost curving ridge, and a *delta* is the center of triangular regions where three different direction flows meet. The locations of the singular points in an example fingerprint are given in Figure 3.1(b). Each fingerprint contains maximal 2 cores and 2 deltas. Apart from its location, a segment of a fingerprint image around an SP has an *orientation;* this chapter also proposes an estimation method for the orientation of SPs.

The most common use of SPs is *registration*, which means that they are used as references to line up two fingerprints (see for instance Chapters 7 and 11. Another example of their use is classification of fingerprints into the Henry classes [Kar96]. The orientation of singular points can be used for more advanced classification methods, or to initialize flow lines in the DF [Kar96, Kaw84, Cho97, Baz01c].

This chapter is organized as follows. In Section 3.2, we propose an efficient implementation of an SP extraction algorithm that is based on the Poincaré index and makes use of small 2-dimensional filters. The algorithm extracts *all* singular points from the DF, including false SPs that are caused by an insufficiently averaged DF. Furthermore, the algorithm determines whether a core or a delta is detected.

Section 3.3 presents an algorithm for estimating the orientation of SPs. As far as we know, there exists only one earlier publication on computing the orientation of SPs [Nak82]. That method examines the DF at a number of fixed positions in a circle around the SP. The position where the DF points best towards the SP is taken as orientation of the SP. The method that is described below uses the entire neighborhood of the SP for the orientation estimate, providing more accurate results.

In Section 3.4, an experiment is presented where the theory is applied to fingerprints



(a) Core (b) Delta

Figure 3.2: Segments of a fingerprint that contain a singular point.

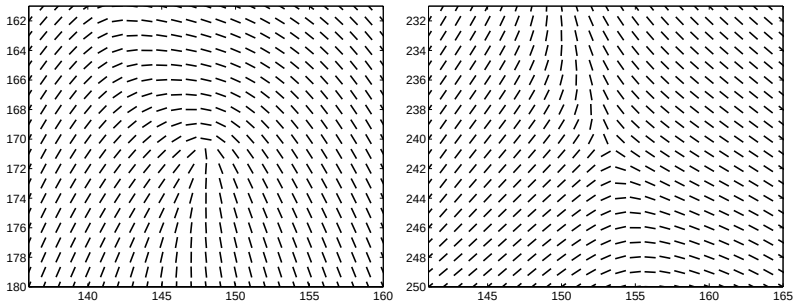
contained in one of the databases used for the *Fingerprint Verification Competition 2000* [Mai00]. In that section, some practical aspects of the algorithms are discussed as well.

3.2 Singular Point Extraction

The subject of this section is extraction of the SPs, which are the points in a fingerprint where the DF is discontinuous. In Figure 3.2, two segments of the fingerprint of Figure 3.1 are shown, one containing a core and one containing a delta. The SPs are somewhere in the center of the segments. However, they cannot be located more accurately than within the width of one ridge-valley structure in the gray-value fingerprint, which is approximately 10 pixels for this example.

In Figure 3.3, the DF of those segments is shown. From this DF, the exact SP location can be determined easily with a resolution of only 1 pixel. Although it seems like a very straightforward task to extract the SPs from the DFs, many different algorithms for SP extraction are known from the literature.

In [Nak82], first areas of high curvature are identified as search areas. Then a feature vector is estimated by taking the difference between the estimated direction and the direction of a double core (whorl) in a number of positions in a circle around a candidate area. This feature vector is classified as being core, delta, whorl or none of these. In [Sri92], first candidate areas of high curvature are selected too. Then, a feature vector is constructed by taking the average directions at four positions around the candidate SP. This feature vector is classified as a core or delta. In [Rao92], some reference models are shifted over the DF, and SPs are detected by a least-squares fit. In [Per98], the local energy of the DF is used as a measure for how much the local DF resembles an SP, and in [Dre99], a neural network is slided over the DF to detect SPs. Finally, in [Jai00b], the ratio of the sines of the DFs in two adjacent regions is used as a measure to detect SPs.



(a) Core (b) Delta

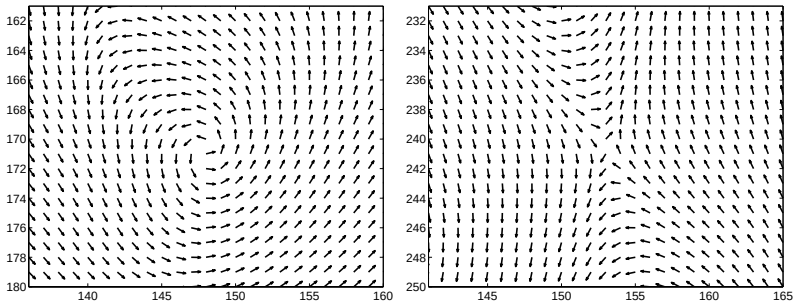
Figure 3.3: Directional fields.

These methods all provide somewhat unsatisfactory results, since they are not capable of consistently extracting the singular points. Instead of providing a Boolean output that indicates whether an SP is present at some location or not, they produce a continuous output that indicates how much the local DF resembles an SP. Postprocessing steps, like thresholds and heuristics, are necessary to interpret the outputs of the algorithms and to make the final decisions.

The method that is presented in this section is based on the Poincaré index, which was first introduced in fingerprint recognition by [Kaw84]. The Poincaré index can be explained using the DFs that are depicted in Figure 3.3. Following a counter-clockwise closed contour around a core in the DF and adding the differences between the subsequent angles results in a cumulative change in the orientation of π and carrying out this procedure around a delta results in $-\pi$. However, when applied to locations that do not contain an SP, the cumulative orientation change will be zero.

Although the Poincaré index provides the means for consistent detection of SPs, the question arises how to calculate this measure. Apart from the problem of how to calculate cumulative orientation changes over contours efficiently, a choice has to be made on the optimal size and shape of the contour. A possible implementation is described in [Hon99]. That paper claims that, in a fingerprint that is scanned at 500 dpi, a square curve with a length of 25 pixels is optimal. A smaller curve results in spurious detections, while a larger curve may ignore core-delta pairs which are close to each other. If the postprocessing step finds a connected area of more than 7 pixels in which the Poincaré index is $\geq \pi$, a core or delta is detected. In case of an area that is larger than 20 connected pixels, 2 cores are detected.

In the implementation that is proposed in this chapter, choices of the size and shape of the contour do not have to be made. Postprocessing steps are not necessary and the cumulative orientation changes over contours are implemented efficiently in small 2-dimensional filters. The method computes for each individual pixel whether it is an SP, and is therefore capable of detecting SPs that are located only a few pixels apart. This property is especially useful for the extraction of SPs from *block-directional fields* (BDFs), which estimate one direction for each $n \times n$ block. Special care has to be taken that high-resolution DFs are sufficiently



(a) Core (b) Delta

Figure 3.4: Squared directional fields.

averaged such that spurious SPs are eliminated beforehand, as the SP-extraction algorithm will detect *all* SPs present in the DF of a given resolution.

The algorithm first takes the squared directional field (SDF). This eliminates the step of π which is encountered in the DF between the orientations $\theta = \frac{1}{2}\pi$ and $\theta = -\frac{1}{2}\pi$. Then, the Poincaré indexes is equal to 2π for a core, -2π for a delta and 0 an area without any SP. The orientation of the SDF, denoted by 2θ, is depicted in Figure 3.4 for the areas around SPs.

Summing the changes in orientation corresponds to summing the gradients of the squared orientation. The gradient vector J can be efficiently precalculated for the entire image by:

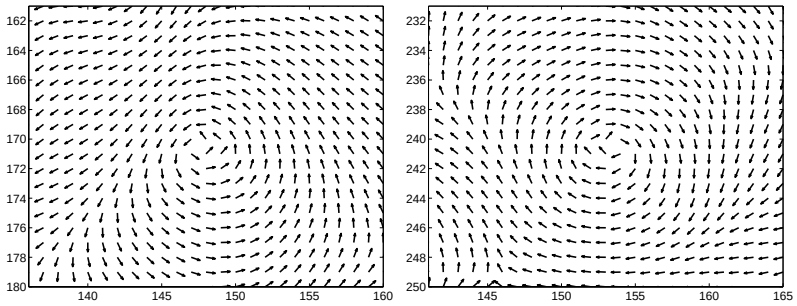
$$\begin{bmatrix} J_x(x,y) \\ J_y(x,y) \end{bmatrix} = \nabla 2\theta(x,y) = \begin{bmatrix} \frac{\partial 2\theta(x,y)}{\partial x} \\ \frac{\partial 2\theta(x,y)}{\partial y} \end{bmatrix} \tag{3.1}$$

In the calculation of the discrete version of this gradient, both components of J should be calculated 'modulo 2π', such that they are always between $-\pi$ and π. This makes the transition from $2\theta = -\pi$ to $2\theta = \pi$ continuous or, in other words, the orientation is considered to be cyclic. The gradient vectors of the squared orientation around both singular points are shown in Figure 3.5.

The next step is the application of Green's Theorem, which states that a closed line-integral over a vector field can be calculated as the surface integral over the rotation of this vector field:

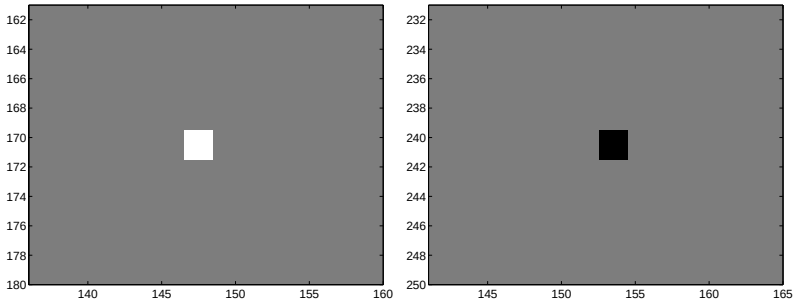
$$\oint_{\partial A} w_x dx + w_y dy = \iint_A \text{rot}[w_x \; w_y]^T dxdy = \iint_A \left(\frac{\partial w_y}{\partial x} - \frac{\partial w_x}{\partial y} \right) dxdy \tag{3.2}$$

where x and y define the coordinate system, A is the area, ∂A is the contour around this area and $[w_x \; w_y]^T$ is the vector field. This theorem is discretized in order to applied it to the summation of the gradients of the squared orientation over the contour:



(a) Core (b) Delta

Figure 3.5: Gradient of squared directional fields.



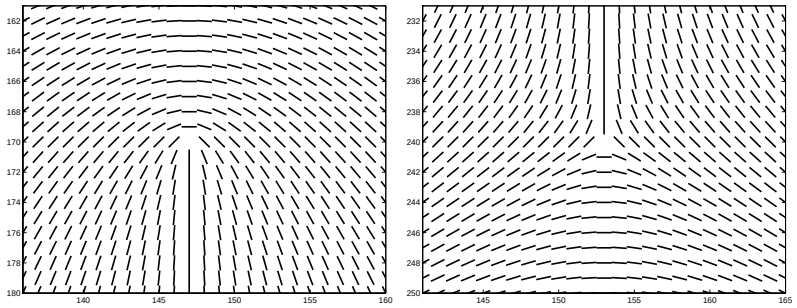
(a) Core (b) Delta

Figure 3.6: Rotation of the gradient of the squared directional fields.

$$Index = \sum_{\Delta x, \Delta y \text{ along } \partial A} (J_x \cdot \Delta x + J_y \cdot \Delta y) = \sum_{A} \text{rot}[J_x \; J_y]^T = \sum_{A} \left(\frac{\partial J_y}{\partial x} - \frac{\partial J_x}{\partial y} \right) \quad (3.3)$$

Since all SPs have to be extracted from the DF, A is taken as a square of 1 pixel. This results in a very efficient method for computation of the Poincaré index, which can be implemented in small 2-dimensional filters. Application of the proposed method will indeed lead to the desired SP locations. Unlike all other SP extraction methods, a core results in a Poincaré index of 2π, a delta in -2π while the index for all other pixels in the image is exactly equal to 0. This is illustrated in Figure 3.6.

The exact locations of the SPs in the DF are just between the pixels. Our method detects an SP in all neighboring pixels of the point, because of the region of support of the gradient operator. This results in SP detections that have a size of 2x2 pixels, as can also be seen in Figure 3.6.



(a) Core (b) Delta

Figure 3.7: Reference models of singular points.

3.3 Orientation of Singular Points

The second subject of this chapter is the estimation of the *orientations* φ of the segments around the extracted SPs. This is the angle over which an image of an SP has been rotated from a standard upright orientation. The method that is described here, makes use of the squared gradient vectors in the neighborhood of an SP, both for the image to be analyzed and for a reference SP. First, reference models of the DFs around standard cores and deltas are constructed. For a core at $(x, y) = (0, 0)$, the reference model that describes the SDF is given by:

$$SDF_{\text{core,ref}} = \frac{(y, -x)}{\sqrt{x^2 + y^2}} \tag{3.4}$$

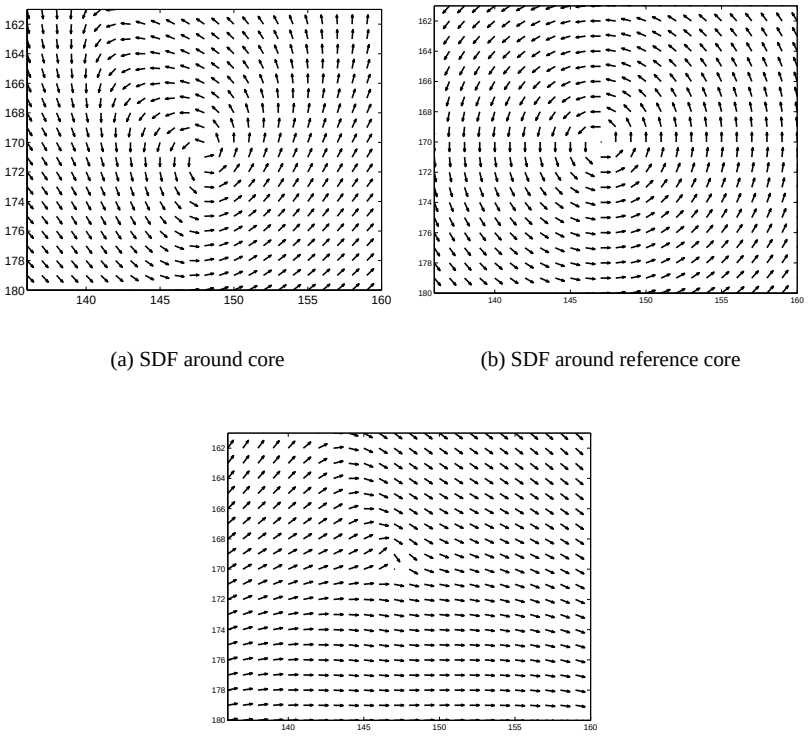
and for a delta at $(x, y) = (0, 0)$, it is given by:

$$SDF_{\text{delta,ref}} = \frac{(-y, -x)}{\sqrt{x^2 + y^2}} \tag{3.5}$$

Note that $|SDF_{\text{core,ref}}| = |SDF_{\text{delta,ref}}| = 1$ for all (x, y). The DFs that are associated with these SDF models are shown in Figure 3.7.

The SDF in the neighborhood of a core, repeated in Figure 3.8(a), ideally looks like the reference model in Figure 3.8(b). The usefulness of the squared gradients is caused by the fact that, when the gray-scale image rotates around the core, all components of the SDF rotate over the same angle, as shown in Appendix C. Therefore, using the complex notation for a two-element vector, the model of a core that has rotated over an angle φ, is given by a reference model with all its components multiplied by $e^{j\varphi}$.

$$SDF_{\text{core},\varphi} = SDF_{\text{core,ref}} \cdot e^{j\varphi} \tag{3.6}$$



(a) SDF around core

(b) SDF around reference core

(c) Orientation estimate

Figure 3.8: Processing steps in the calculation of the orientation of a core.

This property is used for the estimation of the orientation of the core. The orientation of the core with respect to the reference model is found by taking the element-by-element product of the observed squared gradient data $SDF_{\text{core,obs}}(x, y)$ and the complex conjugated of the reference model $SDF_{\text{core,ref}}(x, y)$. This is depicted in Figure 3.8(c). Then, the elements are summed and the sum is divided by the number of matrix elements N, and the angle of the resulting vector is taken.

$$\hat{\varphi}_C = \angle \frac{1}{N} \sum_{x,y} SDF^*_{\text{core,ref}}(x, y) \cdot SDF_{\text{core,obs}}(x, y) \tag{3.7}$$

The relative orientation of a delta with respect to the reference model is given by one third of the angle of the element-by-element product, as also shown in Appendix C:

$$\hat{\varphi}_D = \frac{1}{3} \angle \frac{1}{N} \sum_{x,y} SDF^*_{\text{delta,ref}}(x, y) \cdot SDF_{\text{delta,obs}}(x, y) \tag{3.8}$$

The averaging operator provides an accurate and unbiased estimate for the orientations φ_C and φ_D. If the observed core is exactly a rotated version of the reference core, the orientation estimate gives:

$$\begin{aligned} \hat{\varphi}_C &= \angle \frac{1}{N} \sum_{x,y} SDF^*_{\text{core,ref}}(x,y) \cdot SDF_{\text{core,ref}}(x,y) \cdot e^{j\varphi} \\ &= \angle \frac{1}{N} \sum_{x,y} \left|SDF_{\text{core,ref}}(x,y)\right|^2 \cdot e^{j\varphi} \\ &= \angle e^{j\varphi} = \varphi \end{aligned} \tag{3.9}$$

When applying the orientation estimate to the core of Figure 3.2, it is found to be rotated 4 degrees clockwise with respect to the reference core of Figure 3.7, while the delta of Figure 3.2 is found to be rotated 8 degrees counter-clockwise with respect to the reference delta of Figure 3.7. This corresponds to the estimates that were made by visual inspection.

3.4 Experimental Results

In this section, experiments are presented in which the previously derived results are applied to a large number of fingerprints. It will be shown that application of these methods enables the estimation of very accurate and high resolution DFs, accurate SP locations and correct orientations of the singular points.

We have run our experiments on the second database of the FVC2000 contest [Mai00]. This database contains fingerprint images that are captured by a capacitive sensor with a resolution of 500 pixels per inch. This means that two adjacent ridges are located 8 to 12 pixels apart. In this database, 110 untrained individuals are enrolled, each with 8 prints of the same finger.

The proposed algorithm makes use of small 2-dimensional filters. Because of the efficient algorithm, extraction of the SPs from a 300×300 DF takes 150 ms on a 500 MHz Pentium III machine. It is expected to take less than 5 ms to extract the SPs from a 8×8 BDF of a fingerprint of 300×300 pixels.

In Section 3.4.1, the number of false SPs is used as a measure for the quality of a DF estimate. Then, Section 3.4.2 presents experimental results on the orientation estimation of the SPs.

3.4.1 Singular Point Extraction

In Section 3.2, it has been shown that the SP extraction method correctly extracts SPs from the smooth DF of Figure 3.3. This was also illustrated in Figure 3.1(b) for the fingerprint of Figure 3.1(a). In this section, the question will be answered how well the method performs on a larger set of DFs that are estimated from real fingerprints.

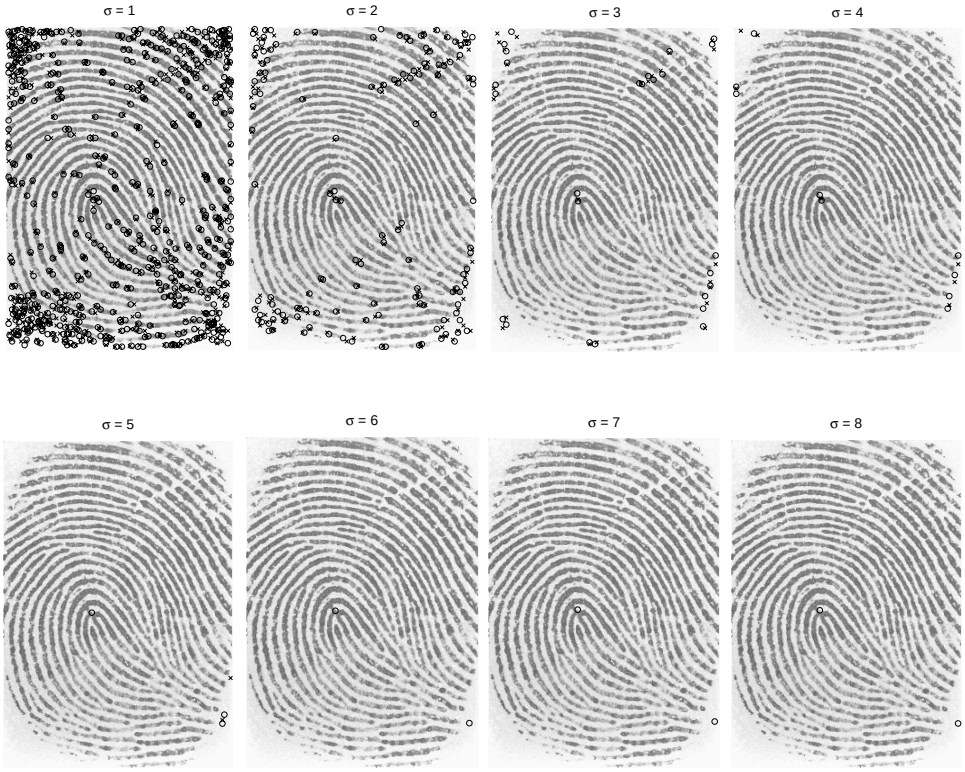


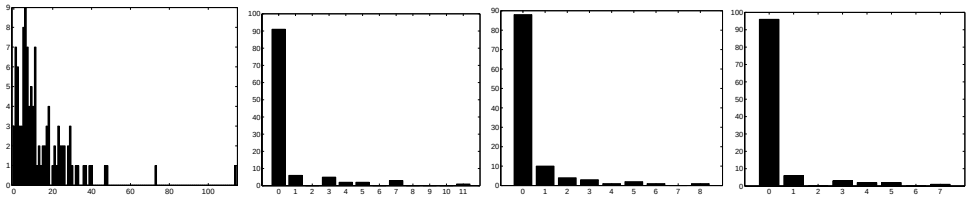
Figure 3.9: Extracted singular points for various values of σ.

As already mentioned in Section 3.2, our method extracts *all* SPs from the DF. In case the directional field is not averaged sufficiently, this may result in many false singular points. A DF that has not been averaged at all, may contain as many as 100 spurious core-delta pairs, especially in noisy regions like the borders of the image. During averaging of the DF, which corresponds to observing the DF from a larger scale, these pairs either merge and disappear or float off the border of the image [Per98]. This is illustrated in Figure 3.9, where the extracted SPs are shown for various values of σ. Another example of this behaviour can be seen from Figure 2.4. For $\sigma = 1$, as many as 5 false cores and 5 false deltas can be identified, which all disappear for $\sigma \geq 3$.

In fingerprint recognition, only those SPs are valid that exist at a scale that is considerable larger than the period of the ridge-valley structures. This means that the SPs have to be extracted from a DF that is estimated at this scale [Lin94]. The coarse-scale directional field can be obtained by averaging it using the algorithms of Chapter 2. Next, the proposed SP extraction method can be applied. In fact, scale and singular point extraction are two different problems. The SP extraction method will only provide satisfactory results if the scale is chosen well by sufficient averaging. Since a fingerprint never contains more than 2 core-delta pairs, this might provide a check whether the right scale has been reached. Experiments have shown that $\sigma = 6$ is a good value for the database that is used in this section.

Table 3.1: Results of SP extraction

Segmentation method	1	2	3	4
Average number of false SPs	15.4	0.8	0.8	0.5
Ratio of fingerprints with false SPs	0.97	0.17	0.2	0.13
Ratio of fingerprints with missed SPs	0	0	0.02	0.05



(a) No segmentation (b) Manual segmentation (c) Segmentation 1 (d) Segmentation 2

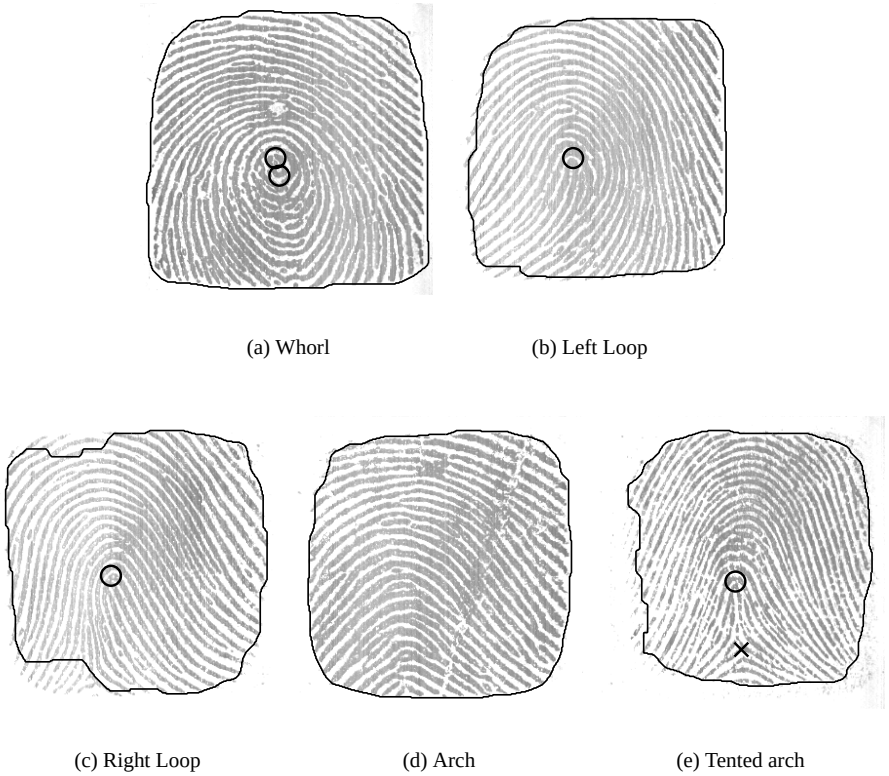
Figure 3.10: Distribution of the number of false singular points for various segmentation methods.

Even when the DF has been averaged sufficiently, the noisy regions outside the fingerprint area may still contain some singular points, as can also be seen from Figures 2.3(a) and 3.9. More averaging in these regions of low coherence does not always solve this problem: some false singular points will remain. This may also be the case in fingerprint regions that are very noisy. A solution is to use *segmentation*, which is the partitioning of the image in a 'foreground' fingerprint area and a 'background' noise area. After segmentation, false SPs in the background are discarded. Details of segmentation are discussed in Chapter 4.

In our experiment, SPs are extracted from the first prints of all fingers of the second FVC2000 database, using the method of Section 3.2 and a Gaussian window with $\sigma = 6$. For the purpose of reference, the SPs in all prints were marked by a human expert. The average number of false and missed SPs are shown in Table 3.1, while the distribution of the numbers of false SPs is shown in Figure 3.10 for 4 different types of segmentation, which are discussed in Chapter 4:

1. No segmentation, the whole image is taken as fingerprint region.
2. Manual segmentation.
3. High resolution segmentation algorithm that uses the coherence estimate as feature and morphological operators to smooth the segmentation result [Baz00a].
4. High resolution segmentation algorithm that uses the coherence, the mean and the variance of the fingerprint image as features and morphological operators to smooth the segmentation result [Baz01d].

In Figure 3.11, the extracted SPs for fingerprints of the five Henry classes are shown. It can be seen that the SP-extraction algorithm has no difficulties in distinguishing a *tented*



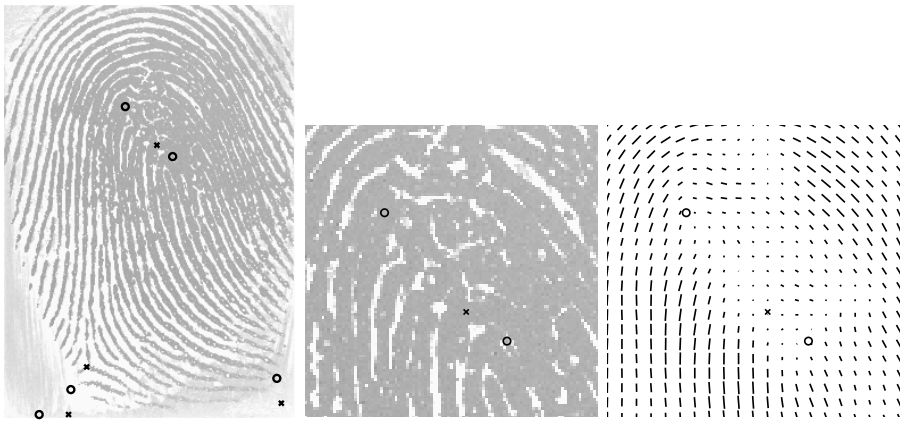
(a) Whorl (b) Left Loop

(c) Right Loop (d) Arch (e) Tented arch

Figure 3.11: SP extraction for examples from each of the five Henry classes.

arch, which contains one core and one delta, from an *arch*, which contains neither of both. Furthermore, the figure shows that the delta in the *right loop* (which can be seen in the lower left corner) is not detected, although it is visible in the image. The segmentation boundary, which is also shown in the figure, positions this delta just outside the foreground area.

In Figure 3.12, an example of the extraction of spurious SPs is shown. From the surroundings of the noisy center area, it can be concluded that this area should contain one core. However, the DF contains 2 cores and 1 delta in this area. As the coherence value is coded by the length of the lines in the DF of Figure 3.12(c), it can be seen that the coherence is very low in this area. These false SPs can be eliminated by further averaging of the DF, but that would need a window as large as $\sigma = 11$. In this case, it would be a better solution to develop segmentation algorithms that are capable of detecting low-quality areas, and discard spurious core-delta pairs from these areas. Another possibility is to assign a certainty measure that can be used in subsequent processing steps to each extracted SPs. Finally, fingerprints of very low quality, having a low coherence value in the entire print, should be rejected outright.



(a) Fingerprint (b) Center area (c) DF of center area

Figure 3.12: Example of extraction of spurious SPs.

3.4.2 Orientation of Singular Points

The last experiment shows the accuracy of the estimated orientation of SPs using the method of Section 3.3. In this experiment, the automatically determined orientations are compared to the manually marked orientations of all valid SPs.

The distribution of the errors of the orientations, $e_\varphi = \hat{\varphi} - \varphi$, is as follows. The estimate is not seriously biased since the mean error is mean$[e_\varphi] = -0.012\text{rad} = -0.7$ degrees, on a scale from 0 to 360 degrees. Furthermore, the variance of the estimate is $\sigma^2_{e_\varphi} = 0.044$, which means that the standard deviation is only $\sigma_{e_\varphi} = 0.21 = 12$ degrees. Therefore, we conclude that our method provides an accurate estimate of the orientations of SPs.

3.5 Conclusions

The singular-point-extraction method that is proposed in this chapter, offers consistent binary decisions and can be implemented very efficiently by using small 2-dimensional filters. It is capable of high resolution SP extraction and does not need to use heuristic postprocessing. Furthermore, it is shown that a high-resolution DF can be used for the accurate estimation of the orientation of SPs. To improve the error rates of SP extraction further, accurate segmentation algorithms have to be developed that are capable of detecting low-quality areas in a fingerprint. Then, spurious core-delta pairs can be discarded from these areas.

Chapter 4

Segmentation of Fingerprint Images

Abstract

An important step in automatic fingerprint recognition is the segmentation of fingerprint images. The task of a fingerprint segmentation algorithm is to decide which part of the image belongs to the foreground, originating from the contact of a fingertip with the sensor and used for further processing, and which part to the background, which is the noisy area at the borders of the image.

In this chapter, two algorithms for the segmentation of fingerprints are proposed. The first method uses three pixel features, being the coherence, the mean and the variance. An optimal linear classifier has been trained for the classification per pixel into foreground or background, while morphology has been applied as postprocessing to obtain compact clusters and to reduce the number of classification errors. The second method uses the Gabor response as additional feature, a third class for low quality regions, and *hidden Markov models* (HMMs) for context-dependent classification.

Human inspection has shown that the proposed methods provide accurate segmentation results. The first method misclassifies 7% of the pixels while the postprocessing further provides compact region estimates. The second method misclassifies 10% of the blocks, but needs no postprocessing. Experiments have shown that the proposed segmentation methods and manual segmentation perform equally well in rejecting false fingerprint features from the noisy background. Parts of this chapter have been published in [Baz01d] and [Kle02].

4.1 Introduction

An important step in an automatic fingerprint recognition system is the *segmentation* of fingerprint images. Segmentation is the decomposition of an image into its components. A captured fingerprint image usually consists of two components, which are called the *foreground* and the *background*. The foreground is the component that originated from the contact of a fingertip with the sensor. The noisy area at the borders of the image is called the background. Some segmentation algorithms also define some part of the foreground as *low quality area*. The task of the fingerprint segmentation algorithm is to decide which part of the image belongs to the foreground, which part to the background, and which part is a low quality area.

Accurate segmentation is especially important for the reliable extraction of features like minutiae and singular points. Most feature extraction algorithms extract many false features when applied to the noisy background or low quality area. Therefore, the main goal of the segmentation algorithm is to discard the background and low quality areas, and thus reduce the number of false features.

Several approaches to fingerprint image segmentation are known from literature. In [Meh87], the fingerprint is partitioned in blocks of 16×16 pixels. Then, each block is classified according to the distribution of the gradients in that block. In [Meh89], this method is extended by excluding blocks with a gray-scale variance that is lower than some threshold. In [Rat95] the gray-scale variance in the direction orthogonal to the orientation of the ridges is used to classify each 16×16 block. In [Jai97c], the output of a set of Gabor filters is used as input to a clustering algorithm that constructs spatially compact clusters. In this chapter, fingerprint images are segmented based on multiple features, using two different classification methods.

This chapter is organized as follows. First, Section 4.2 discusses pixel feature extraction, which is the basis of our segmentation algorithms. Next, Section 4.3 presents a direct pixel feature classifier that is equipped with post-processing methods for the reduction of classification errors. Finally, Section 4.4 presents a segmentation algorithm that is based on hidden Markov models.

4.2 Feature Extraction

The first step in the development of an algorithm for fingerprint image segmentation is the selection of useful *pixel* or *block features*. Note that the term 'feature' is used here to refer to properties of individual pixels whereas it was used earlier to refer to properties of the entire (foreground) image (such as the list of minutiae locations). In the rest of this chapter, the correct meaning of 'feature' should become clear from the context. For each pixel or block in the fingerprint image, the pixel features are extracted, and each block is classified according to the extracted feature values.

From many alternatives, we have selected four features that contain useful information for segmentation. These features are the *coherence*, the *local mean*, the *local variance* or

standard deviation, and the *Gabor response* of the fingerprint image. Instead of using pure block-wise processing, the smoothing window that is used for noise reduction and the block size are decoupled as explained in Section 2.3. For noise reduction, the features are averaged by a Gaussian window W with $\sigma = 6$, providing a combination of both localized and smoothly changing features.

1. The coherence gives a measure how well the gradients are pointing in the same direction. Since a fingerprint mainly consists of parallel line structures, the coherence will be considerably higher in the foreground than in the background. In a window W around a pixel, the coherence is defined as:

$$Coh = \frac{\left|\sum_W (G_{s,x}, G_{s,y})\right|}{\sum_W \left|(G_{s,x}, G_{s,y})\right|} = \frac{\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}}{G_{xx} + G_{yy}} \tag{4.1}$$

   where $(G_{s,x}, G_{s,y})$ is the squared gradient, $G_{xx} = \sum_W G_x^2$, $G_{yy} = \sum_W G_y^2$, $G_{xy} = \sum_W G_x G_y$ and (G_x, G_y) is the local gradient. More information on the coherence can be found in Section 2.2.

2. The average gray value is the second pixel feature that is useful for the segmentation of fingerprint images. For most fingerprint sensors, the ridge-valley structures can be approximated by black and white lines, while the background, where the finger does not touch the sensor, is rather white. This means that the mean gray value in the foreground is in general lower, i.e. darker gray, than it is in the background. Using I as the intensity of the image, the local mean for each pixel is given by:

$$Mean = \sum_W I \tag{4.2}$$

3. The variance or standard deviation is the third pixel feature that can be used. In general, the variance of the ridge-valley structures in the foreground is higher than the variance of the noise in the background. The variance is for each pixel given by:

$$Var = \sum_W (I - Mean)^2 \tag{4.3}$$

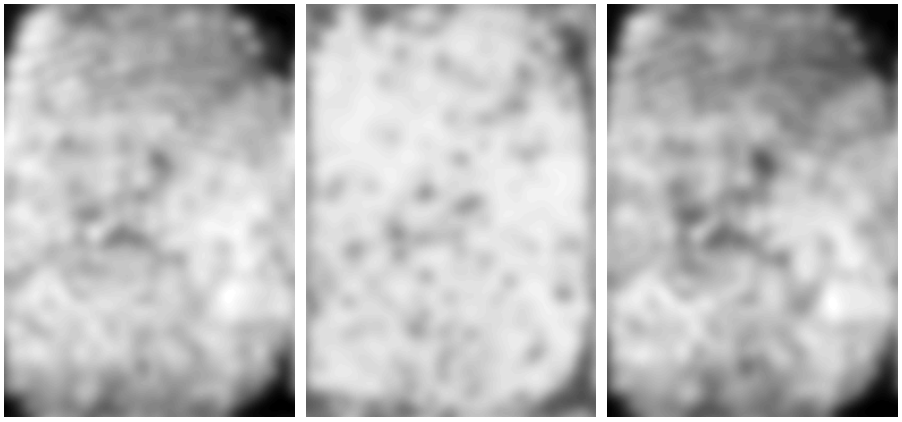
   and the standard deviation by

$$Std = \sqrt{Var} \tag{4.4}$$

4. The Gabor response is the smoothed sum of the absolute values of the fingerprint images that have been filtered by the complex Gabor filter, as discussed in Appendix D. It can be interpreted as the local standard deviation of the fingerprint image after enhancement. Therefore, the Gabor response is expected to be higher in the foreground region.

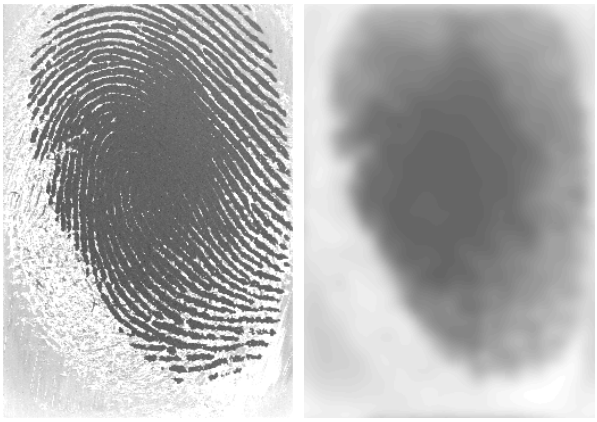
The characteristics of these features in a fingerprint of high quality are shown in Figure 4.1, while the features in a fingerprint with a low quality region are shown in Figure 4.2.

(a) Fingerprint (b) Mean

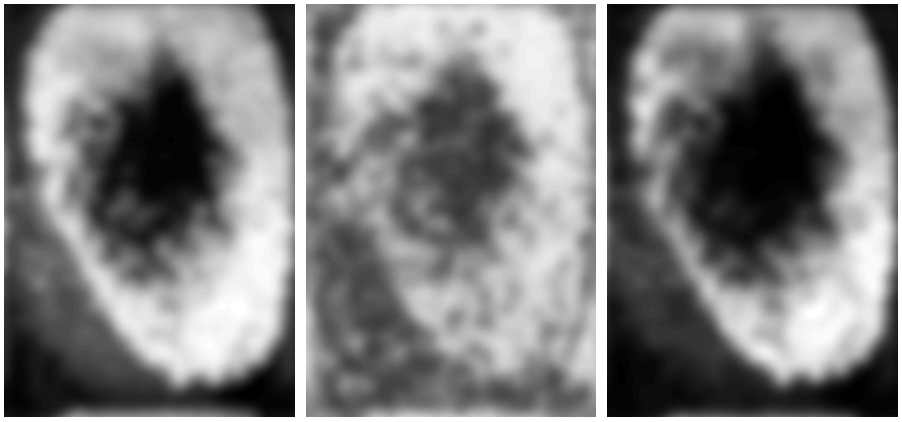


(c) Standard deviation (d) Coherence (e) Gabor response

Figure 4.1: Segmentation features for the high quality fingerprint image. High feature values are mapped to white, while low values are mapped to black.

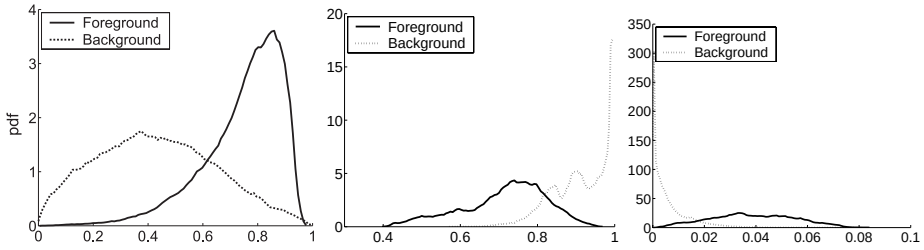


(a) Fingerprint (b) Mean



(c) Standard deviation (d) Coherence (e) Gabor response

Figure 4.2: Segmentation features for the fingerprint image with low quality region. High feature values are mapped to white, while low values are mapped to black.



(a) Coherence (b) Mean (c) Variance

Figure 4.3: Distributions of the pixel features in the foreground and background areas of Database 2.

4.3 Direct Feature Classification

In this section, a direct feature classification approach is taken. Blocks of the size of one pixel are used and the features in each pixel are classified independently, which is described in Section 4.3.1. We use only two classes: foreground and background. In order to obtain compact clusters for each region, postprocessing has to be applied to the classification result. This is discussed in Section 4.3.2. Experimental results that are achieved using this method are presented in Section 4.3.3.

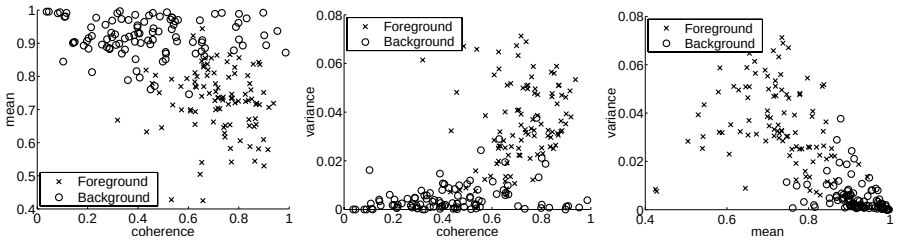
4.3.1 Classification

To evaluate the usefulness of the pixel features that are proposed in Section 4.2, their probability density functions have been determined for both the foreground and the background area. For this experiment, Database 2 of the Fingerprint Verification Competition (FVC2000) has been used [Mai00, Mai02]. This database has been acquired from untrained volunteers with a capacitive sensor. Examples of fingerprints from this database are shown in Figure 4.5.

In this experiment, the first three features (*Mean*, *Var* and *Coh*) are used. In order to obtain the distributions of these three pixel features in both the foreground and the background area, 30 fingerprint images (`100_1.tif` - `29_1.tif`) have been segmented manually. The distributions of *Coh*, *Mean* and *Var* are shown in Figure 4.3, while the joint distributions of the combinations of two features are depicted in Figure 4.4.

Many segmentation algorithms use unsupervised clustering algorithms to assign feature vectors to either foreground or background [Jai97c, Pal93]. However, in this chapter we will follow a supervised approach since examples of the pixel features in both areas are available. Using this method, a classification algorithm can be constructed that minimizes the probability of misclassifying feature vectors.

Many different classification algorithms exist that can be applied to this problem. One can for instance think of neural networks, support vector machines, etc. to find the optimal



(a) Coherence - Mean (b) Coherence - Variance (c) Mean - Variance

Figure 4.4: Joint distributions of the pixel features for foreground and background areas of Database 2.

decision boundaries [Jai00a]. However, since we want to apply the classifier to all pixels in a fingerprint image, we prefer a classification algorithm that has low computational complexity. We have therefore chosen to use a linear classifier, which tests a linear combination of the features, given by:

$$v = \mathbf{w}^T\mathbf{x} = w_0 + w_1 Coh + w_2 Mean + w_3 Var \tag{4.5}$$

where v is the value to be tested, $\mathbf{w} = [w_0\ w_1\ w_2\ w_3]^T$ is the weight vector and $\mathbf{x} = [1\ Coh\ Mean\ Var]^T$ is the feature vector. Then, using class ω_1 for the foreground, class ω_0 for the background and $\hat{\omega}$ for the assigned class, the following decision function is applied:

$$\hat{\omega} = \begin{cases} \hat{\omega}_1 & \text{if } \mathbf{w}^T\mathbf{x} > 0 \\ \hat{\omega}_0 & \text{if } \mathbf{w}^T\mathbf{x} \leq 0 \end{cases} \tag{4.6}$$

This classifier essentially tests the output of a linear neuron of which the output is given by $v = \mathbf{w}^T\mathbf{x}$ [Hay99]. This classifier is trained by first setting the desired responses d. Here, we choose $d_{\omega_1} = 1$ and $d_{\omega_0} = -1$. Then, the error e is given by:

$$e = d - \mathbf{w}^T\mathbf{x} \tag{4.7}$$

and the neuron can be trained by the LMS algorithm [Hay99]. This algorithm adapts the weight vector $\hat{\mathbf{w}}$ according to:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta(n)\mathbf{x}(n)e(n) \tag{4.8}$$

For better convergence, a decreasing learning rate is used. More specifically, we use the search-then-converge schedule:

$$\eta(n) = \frac{\eta_0}{1 + n/\tau} \tag{4.9}$$

Table 4.1: Results of the linear classifier on Database 2.

Features	**w**	$p(\hat{\omega}_0\|\omega_1)$	$p(\hat{\omega}_1\|\omega_0)$	p_{error}
Coh	[-0.54 0.84]	0.205	0.185	0.195
Mean	[0.64 -0.77]	0.142	0.146	0.144
Var	[-0.017 1.00]	0.095	0.102	0.099
Coh, Mean	[0.59 0.13 -0.80]	0.105	0.117	0.111
Coh, Var	[-0.019 0.0071 1.00]	0.074	0.104	0.089
Mean, Var	[0.026 -0.053 1.00]	0.075	0.069	0.072
Coh, Mean, Var	[0.015 0.013 -0.05 0.99]	0.074	0.062	0.068

Monotonic classifier behavior can be obtained by using the same error measure for both training and evaluation. Since only the sign of the output is tested for classification, the corresponding activation function is given by:

$$y = \text{sign}(v) \tag{4.10}$$

where

$$\text{sign}(v) = \begin{cases} 1 & \text{for } v > 0 \\ -1 & \text{for } v \leq 0 \end{cases} \tag{4.11}$$

This classifier is known as Rosenblatt's Perceptron; it consists of a single neuron of the McCulloch-Pitts type. For this configuration, the LMS training scheme, as given by Expressions 4.7 and 4.8, can still be used. However, in this case, the error is only non-zero for input vectors that are incorrectly classified. Rosenblatt's Perceptron is proven to converge for 2 linearly separable classes. However, it oscillates when applied to our data set, where the classes are not linearly separable. Furthermore, the combination with a decreasing learning rate makes the weights go to zero. Such a behavior can be avoided by a regularization condition on the weights. Therefore, the length of the weight vector is normalized to $|\mathbf{w}| = 1$ after each adaptation.

Using a normalized **w** and appropriate parameter values, for instance 10^6 epochs, $\eta_0 = 10^{-4}$ and $\tau = 10^4$, the weights converge to good classification boundaries. The results are shown in Table 4.1. This table gives for all combinations of pixel features the optimal weight vector **w**, the probability that a foreground pixel is classified as background $p(\hat{\omega}_1|\omega_0)$, the probability that a background pixel is classified as foreground $p(\hat{\omega}_0|\omega_1)$ and the probability of error p_{error} which is the average of $p(\hat{\omega}_0|\omega_1)$ and $p(\hat{\omega}_1|\omega_0)$. For each combination, the vector **x** is composed the feature values given in the first column preceded by a '1'. The table shows that the best performance is obtained by using a combination of all three features, providing an error rate of 6.8%.

Figure 4.5: Segmentation results of some fingerprints from Database 2.

4.3.2 Postprocessing

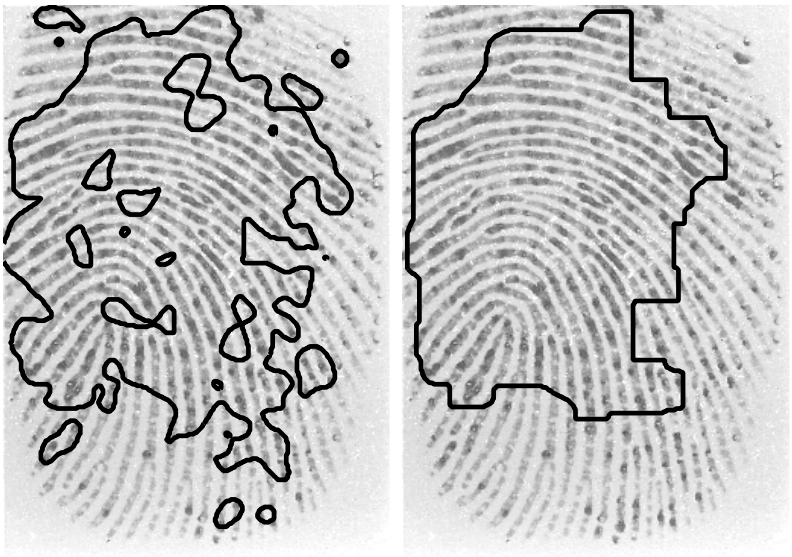
It was shown in the previous section that the classification algorithm misclassifies 6.8 % of the pixels. In some cases, this leads to a 'noisy' segmentation, where spurious small areas of one class show up inside a larger area of the other class. However, meaningful segmentation of fingerprints whould consist of compact clusters. In [Pal93], it is suggested to use information of neighboring pixels for this purpose. However, this is already taken care of up to some extent by the classification algorithm since the pixel features are calculated by averaging over a spatial window W.

More compact clusters can be obtained by a number of different postprocessing methods. It is possible to use either boundary-based methods like *curve fitting* and *active contour models*, or region-based methods like *region growing* and *morphology* [Jai89]. We have chosen to apply morphology to the classification estimate. This method 'repairs' the estimate by removing small areas, thus creating more compact clusters. It reduces the number of false classifications. First, small clusters that are incorrectly assigned to the foreground are removed by means of an *open* operation. Next, small clusters that are incorrectly assigned to the background are removed by a *close* operation.

4.3.3 Experimental Results

This section presents experimental results of the segmentation algorithm. First, in Figure 4.5, segmentation results are shown for three fingerprints from FVC2000 Database 2. The segmentation algorithm has been trained on fingerprints of this database, but not on these particular fingerprints. Human inspection shows that the algorithm provides satisfactory results. The effect of the morphology is shown in Figure 4.6.

Apart from human inspection, there are several ways of quantitatively evaluating the results of a segmentation algorithm. For instance, the number of classification errors could be



(a) Before morphology (b) After morphology

Figure 4.6: Effect of postprocessing.

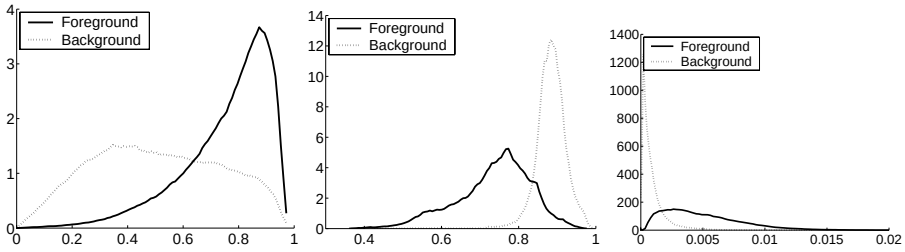
Table 4.2: Results of singular point extraction from Database 2.

Segmentation method	no	manual	all
Average number of false SPs	15.4	0.8	0.5
Ratio of fingerprints with false SPs	0.97	0.17	0.13
Ratio of fingerprints with missed SPs	0	0	0.05

used as a performance measure. This is exactly the measure that was used during training and was found to be 6.8% for the optimal classifier. Another possibility is to evaluate a segmentation algorithm by counting the number of false and missed fingerprint features like minutiae or singular points. The results for the singular-point extraction are shown in Table 4.2.

In this table, results of the singular point extraction algorithm are shown for no segmentation, manual segmentation, and the proposed segmentation method. For each segmentation method, the table shows three measurements: the average number of false singular points (SPs) in a fingerprint image, the ratio of the fingerprint images in which false SPs are found, and the ratio of the fingerprint images in which true SPs are discarded by the segmentation algorithm.

The table shows that the proposed segmentation method rejects more false SPs than the manual method. This is caused by the fact that the proposed segmentation method is allowed to estimate holes in the foreground area at noisy areas in a fingerprint image where false SPs are likely to occur. However, this may cause true SPs to be discarded since they may also be located in these areas.



(a) Coherence (b) Mean (c) Variance

Figure 4.7: Distributions of features in the foreground and background areas for Database 1.

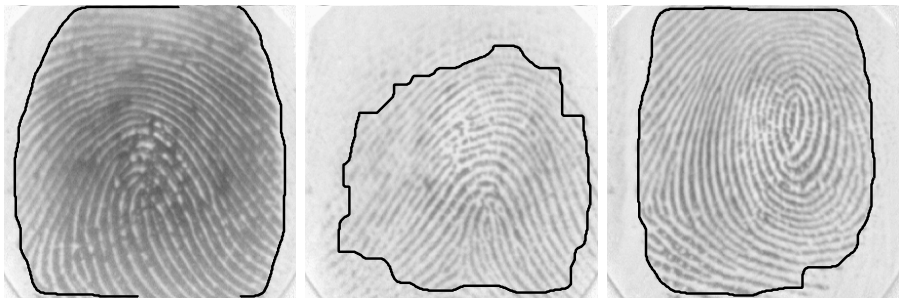


Figure 4.8: Segmentation results of some fingerprints from Database 1.

Next, the applicability of the optimal classifier to other databases has been investigated. For this purpose, FVC2000 Database 1 is used. This database is acquired from untrained volunteers using an optical sensor. The distribution of the pixel features for the foreground and background areas are shown in Figure 4.7, while examples of the fingerprint images are shown in Figure 4.8.

Comparing Figure 4.7 to Figure 4.3 shows that the distributions of the pixel features (especially the variance) are different for both databases. Therefore, another classifier was trained on the images features of Database 1. The results are shown in Table 4.3, which shows slightly different weight vectors and error probabilities. It can be seen that the classifier of Database 1 assigns most importance to *Mean*, while the classifier of Database 2 assigns most importance to *Var*. The segmentation results of some fingerprint images are shown in Figure 4.8.

The last experiment is the direct application of the optimal classifier to one database, while it is trained on another database. The results of this experiment for Databases 1 and 2 are shown in Table 4.4. The columns are labelled with $\mathbf{w}_1$ and $\mathbf{w}_2$ where the subscript refers to the database for which the classifier has been trained. It can be seen that the application of classifier 1 on Database 2 performs suboptimally (14.3% error instead of 6.8%), while the application of classifier 2 on Database 1 results in very low performance (40.4% error instead

Table 4.3: Results of the linear classifier on Database 1.

Features	**w**	$p(\hat{\omega}_0\|\omega_1)$	$p(\hat{\omega}_1\|\omega_0)$	p_{error}
Coh	[0.82 -0.57]	0.279	0.247	0.263
Mean	[-0.76 0.65]	0.110	0.115	0.113
Var	[1.00 -0.0018]	0.155	0.122	0.139
Coh, Mean, Var	[0.027 -0.77 0.061 0.63]	0.104	0.103	0.103

Table 4.4: Error probabilities of the classifiers on databases that they are not trained on.

	$\mathbf{w}_1$	$\mathbf{w}_2$
Database 1	0.103	0.404
Database 2	0.143	0.068

of 10.3%). Therefore, it can be concluded that a classifier always has to be trained on that specific fingerprint sensor characteristics that it has to be applied to.

4.4 Segmentation using Hidden Markov Models

An alternative to the morphological postprocessing is the use of *hidden Markov models* (HMMs) which are widely used in speech recognition [Rab89]. This method takes into account the context, or surroundings, for each feature vector to be classified. Using estimations of the probability of class transitions and the conditional feature distributions, the segmentation is found that maximizes the likelihood of these observations. For the segmentation method that is proposed in this section, a third class, representing low-quality regions, is used to improve the usefulness of the segmentation results.

4.4.1 Hidden Markov Models

A *hidden Markov model* (HMM) is a model that describes the statistical properties of a system that generates a signal. At each time instance, the system is in one state of a number of possible states. The system generates an output signal of which the probability density function depends on the current state. However, only the output signal can be observed, while the actual states are hidden for the observer. At the transitions between the time instances, the state of the system may change according to a state transition matrix that contains probabilities for all state transition.

An HMM can be used to find the most likely state sequence, given the sequence of output values. First the HMM has to be trained, using labelled data. From the data, the transition probabilities can be estimated directly by counting. For the simplest model, the output probability densities can be modelled by a Gaussian function, which can also be estimated directly from the set of data per state. If more complex HMMs are used, containing for instance more states per class or Gaussian mixtures as probability density functions, more advanced estimation methods, like the EM algorithm, have to be used.

Table 4.5: Confusion matrix for HMM segmentation.

	FG	BG	LQ
$\hat{\text{FG}}$	0.8851	0.0400	0.2062
$\hat{\text{BG}}$	0.0522	0.8936	0.1866
$\hat{\text{LQ}}$	0.0627	0.0663	0.6072

Given a trained HMM and an output sequence, the most likely state sequence can be found by means of the Viterbi algorithm [Rab89]. The Viterbi algorithm is a recursion that makes use of the Markov properties of the system: the next state is only dependent of the current state, and not of all previous states. Given the optimal state sequences up to all states at the previous time instance, the state transition probabilities, and the current observation, the Viterbi algorithm determines the optimal state sequences up to all possible current states. Applying this optimization to the entire sequence gives the optimal state sequence.

4.4.2 Segmentation using HMMs

An HMM can be used for classification of sequences of feature vectors by letting the hidden states represent the classes and the outputs the feature vectors. In fingerprint image segmentation, sequences of blocks have to be classified, and the context of a block provides important additional information. Therefore, hidden Markov models are well suited for this task. Although HMMs have been applied to other image segmentation problems successfully, approaches for their application to fingerprint segmentation are not known from the literature.

For complexity reasons, we have chosen a 1-dimensional approach (complexity linear in the size of the image) instead of a full 2-dimensional HMM (complexity exponential in the size of the image) [Li00], which may give better segmentation performance. The HMMs are used to classify the blocks in each row of the fingerprint image.

4.4.3 Experimental Results

The three areas (foreground, background and low-quality) have been labelled manually in 20 fingerprints, using a block size of 8 by 8 pixels. From this training data, the state transition probability matrix and the Gaussian probability density functions of the features have been estimated. Next, the trained HMM was used for classification.

The best results were obtained by using all the features that are discussed in Section 4.2. The addition of other features, like for instance DCT coefficients, did not improve the performance. The confusion matrix over a test set of another 20 manually labelled fingerprints is given in Table 4.5, where 'FG' indicates foreground, 'BG' indicates background, 'LQ' indicates the low quality region. The plain letters indicate the true class and the letters with a hat indicate the estimated class.

Although the confusion matrix is not significantly better than the confusion matrix without the use of HMMs (which is obtained by setting the values in the transition matrix all

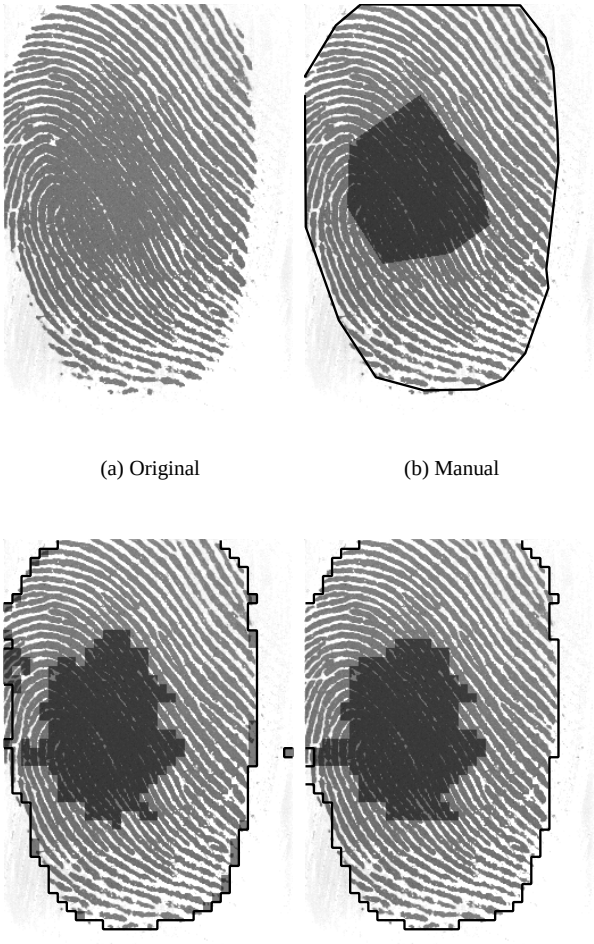
equal), the shape of the areas fits the manual segmentation better, and the number of isolated misclassifications is reduced. An example of the results is shown in Figure 4.9. The main differences between both segmentations are the incorrect low quality areas at the transitions between foreground and background in the direct classification, which vanish when using HMM segmentation. Furthermore, the shape of the low quality region is slightly more regular for the HMM segmentation. The minimal differences in the confusion matrices of both methods can be explained by inaccuracies in the exact location of the area borders in the manually labelled examples. A solution would be to define don't-care regions in between the classes.

4.5 Conclusions and Recommendations

In this chapter, two algorithms for the segmentation of fingerprints are proposed. The first method uses three pixel features, being the coherence, the mean and the variance. An optimal linear classifier has been trained for the classification per pixel into foreground or background, while morphology has been applied as postprocessing to obtain compact clusters and to reduce the number of classification errors. The second method uses the Gabor response as additional feature, an additional class for low quality regions and HMMs for context dependent classification.

Human inspection has shown that the proposed methods provide accurate segmentation results. The first method misclassifies 7% of the pixels while the postprocessing further provides compact region estimates. The second method misclassifies 10% of the blocks, and obtains compact clusters from the data statistics instead of using postprocessing. Experiments have shown that the proposed segmentation methods and manual segmentation perform equally well in rejecting false fingerprint features from the noisy background.

The performance of the direct feature classification method may be improved by choosing a better suited classifier. The performance of the HMM based method can be improved by several adaptations. First, more realistic probability density functions can be considered than single Gaussian densities. One can for instance think of Gaussian mixtures for this purpose. Furthermore, more states can be used per class, representing for instance the orientations of the ridge lines in the fingerprint. This may help estimating more accurate output probability density functions and constrain the transition matrix to only valid transitions.



(a) Original (b) Manual

(c) Without HMM (d) With HMM

Figure 4.9: Segmentation results of a fingerprint image with and without HMM. The solid line indicates the border between foreground and background areas, while low quality areas are indicated by the darker gray-scales.

Chapter 5

Minutiae Extraction

Abstract

This chapter discusses the extraction of minutiae from a gray-scale fingerprint image. This is a crucial step since the minutiae are the basis of most fingerprint matching systems. First, the traditional approach is discussed, and improvements in computational complexity and extraction performance are proposed. Next, two alternative approaches are presented that are based on learning image exploring agents, that use genetic programming and reinforcement learning respectively. It turns out that the traditional approach provides the best quality minutiae extraction. Part of the traditional approach has been published in [Baz02a], the genetic programming approach has been published in [Meu00, Meu01] and the reinforcement learning approach has been published in [Baz01e].

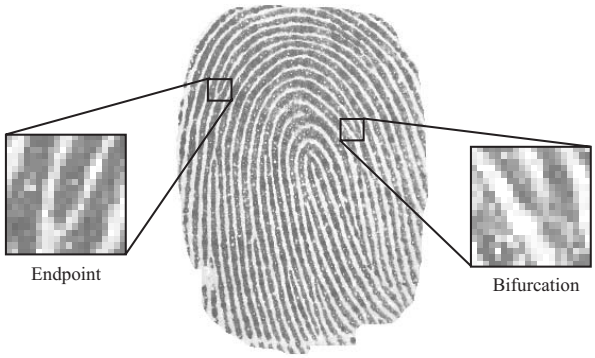


Figure 5.1: Example of a fingerprint and two minutiae.

5.1 Introduction

This chapter discusses the extraction of minutiae from a gray-scale fingerprint image, illustrated in Figure 5.1. This is a crucial step since the minutiae are the basis of most fingerprint matching systems. Any errors that are made in the minutiae extraction will cause a lower matching performance in minutiae based matching algorithms directly.

This chapter is organized as follows. First, in Section 5.2, the traditional approach is discussed. Next, two alternative approaches are presented that are based on learning image exploring agents. Section 5.3 presents an agent that is trained by means of genetic programming, while the approach of Section 5.4 makes use of reinforcement learning. In Section 5.5, a new method is proposed for the evaluation and comparison of different minutiae extraction algorithms.

5.2 Traditional Minutiae Extraction

Traditional minutiae extraction from a gray-scale fingerprint image uses a number of sequential processing steps [Jai97b]. Although most algorithms use more or less the same approach, many possible implementations exist. This section describes our implementation of this method. First, the ridge-valley structures are enhanced using Gabor filters, as described in Section 5.2.1. Next, Section 5.2.2 describes the binarization and thinning of the image, and the actual minutiae extraction. Finally, in Section 5.2.3, postprocessing is applied to remove false minutiae.

5.2.1 Enhancement

One of the problems in fingerprint recognition is the poor quality of fingerprint images. The task of the *enhancement* is to suppress the noise in the fingerprint image and to enhance the

(a) High quality (b) Poor quality

Figure 5.2: Fingerprint images of high and poor quality.

ridge-valley structures, thereby enabling more accurate minutiae extraction. Examples of a high-quality and a poor-quality fingerprint image are shown in Figure 5.2. To illustrate the need for enhancement algorithms, Figure 5.3(a) shows a medium-quality fingerprint image and Figure 5.3(b) shows an example of a fingerprint image that has been binarized using a local threshold without any enhancement. It is clear that this image contains too much noise for reliable minutiae extraction.

Various fingerprint enhancement methods exist. The simplest enhancement method applies a low-pass filter to the image, thus suppressing the high-frequency noise. An example of a thresholded fingerprint image that is enhanced by a low-pass filter is shown in Figure 5.3(c).

The second class of methods makes use of FFT-based techniques [Can95, Wil01]. The fingerprint image is subdivided into overlapping blocks of 32 by 32 pixels that are processed separately. The FFT is taken, and the result is processed non-linearly in order to enhance the ridge valley structures. Each element in the block is for instance multiplied by its absolute value, possibly raised to a specific power between 1 and 2. This enhances the spatial frequencies that are strongly present in the block and attenuates the other components. Next, the block is transformed back to the spatial domain by means of an inverse FFT. An example of the results of this method is shown in Figure 5.3(d).

The third class of method makes use of oriented filters that are controlled by the estimated ridge orientation [She94, Hon98]. To each pixel, a filter is applied that enhances those structures that agree with the local ridge orientation. For the sake of computational efficiency, the entire image is pre-filtered by a number of bandpass filters with fixed orientations, distributed along the entire orientation range from 0 to π. These filters all enhance a different part of the ridge-valley structures, as shown in Figure 5.5. Next, the pre-filtered images are combined as a locally weighted sum. The weights are set according to the local ridge orientations, such that the pre-filtered image that enhances a specific part of the image best, receives the largest weight in that area.

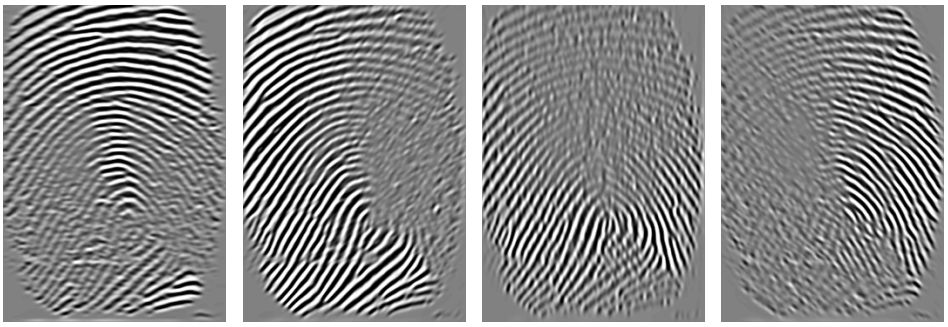
(a) Fingerprint image (b) No enhancement

(c) Low-pass (d) FFT (e) Gabor

Figure 5.3: Fingerprint images that have been enhanced by different methods and binarized for visualization purposes.

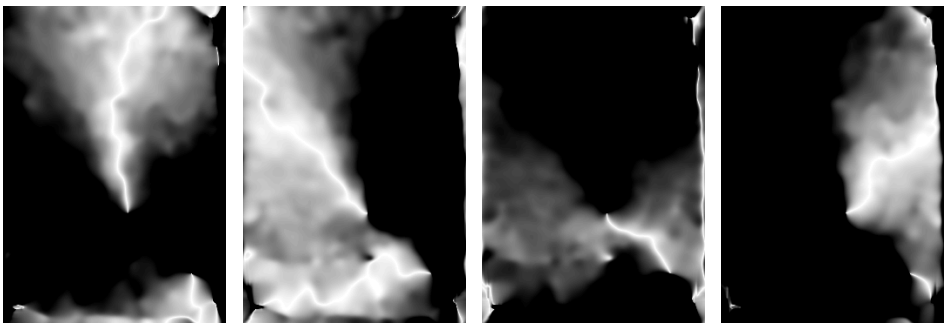
(a) Original (b) Enhanced

Figure 5.4: Original and Gabor enhanced fingerprint images.



(a) 0π (b) $\pi/4$ (c) $\pi/2$ (d) $3\pi/4$

Figure 5.5: Fingerprint images that have been filtered by Gabor filters with different orientations.



(a) 0π (b) $\pi/4$ (c) $\pi/2$ (d) $3\pi/4$

Figure 5.6: DF-based weights for combining filtered images of Figure 5.5.

There are a few critical points in this algorithm. The first is the number of filters and their exact shape. We use Gabor filters for this task, which provide an optimal balance between locality and frequency selectivity [Jai97c]. Apart from the orientation, the spatial frequency and the smoothness have to be selected for Gabor filters. These filters are discussed in more detail in Appendix D.

The spatial frequency of the ridge-valley structures varies considerably within a single fingerprint and between fingerprints. One possibility to handle this is to use filters with several spatial frequencies for each orientation, and to weigh these according to a spatial frequency estimate. However, to reduce computational complexity, we tune the spatial frequency of the Gabor filters to the average ridge-frequency of the entire fingerprint database. Consequently, the peaks in the frequency responses of the filters have to be chosen relatively wide. Otherwise, the filter may for instance construct two small ridges, where one broad ridge is in fact present in the fingerprint. Furthermore, the minutiae are the discontinuities in the image, where the local gray-value structure differs from a single plain sine wave. A filter that enhances only a very small frequency band would suppress these feature points. On fingerprints that were acquired at 500 dpi, we obtained the best results using $f = 0.125$, which corresponds to a ridge-valley period of 8 pixels, and $\sigma = 3$. Although 8 filters are used in most applications, we found that for these parameter settings four filters are sufficient, oriented at $\theta = \{0,\ \pi/4,\ \pi/2,\ 3\pi/4\}$.

Next, the question has to be answered how to combine the pre-filtered images into one enhanced image. When using the DF for this purpose, each pixel in the filtered image is interpolated in a linear way between the pre-filtered images that correspond to the two closest filter orientations. The DF-based weight images for each filter orientation are shown in Figure 5.6.

Finally, the algorithm depends heavily on the local ridge orientation estimate. If this estimate is inaccurate, the algorithm may result in spurious ridges in the estimated ridge direction. An alternative approach that solves this problem is to use smoothed absolute values of the complex Gabor responses (see Figure D.4 in Appendix D) as weight images. However, these weight images do not sum to one, and they are not equal to zero for ridge orientations

that are orthogonal to the filter orientation. Human inspection indicates that this results in less suppression of the noise. For these reasons, use of the DF as weight image provides better results than use of the absolute value of the complex Gabor response.

5.2.2 Minutiae extraction

After enhancement (and segmentation), the minutiae have to be extracted. For this purpose, the image is first binarized by a thresholding operation. This converts each gray-scale pixel value to a binary value (black or white). A dynamic threshold has to be applied, which means that the threshold is adjusted to the local characteristics of the image, thus compensating for differences in image intensity.

Next, the binarized image is thinned. This is a morphological operation that reduces the width of each ridge to a single pixel, thus creating a skeleton of the image. We use a very efficient thinning algorithm that codes each 3 by 3 pixel neighborhood in a 9-bit integer. This number is used as index into a lookup table that contains the binary value after the current thinning step. Two different lookup tables that both take care of a different thinning direction are used iteratively, until the image has converged to its final skeleton. Our thinning algorithm is inspired by the thinning algorithm that is supplied with Matlab, but it replaces 3 lookup table operations, 3 binary 'and's and 1 binary 'not' with only 1 lookup table operation.

The next phase is the actual minutiae detection in the skeleton image. For this task, again a lookup table operation is used for efficiency. In this table, endpoints are defined as black pixels that have only one black pixel in their 3 by 3 neighborhood, while bifurcations have three black pixels that are pairwise unconnected in their 3 by 3 neighborhood.

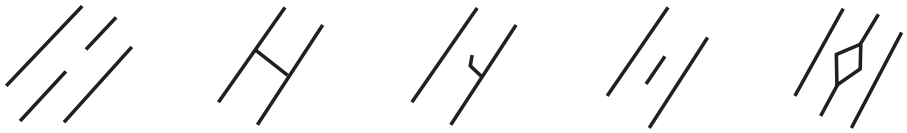
5.2.3 Postprocessing

Although many false minutiae are suppressed in the skeletons of enhanced fingerprint images, in general those skeletons still contain a number of false minutiae. The objective of postprocessing is to eliminate those false minutiae from the skeleton image, while maintaining the true ones.

Many postprocessing methods have been proposed in literature, see for instance [Xia91, Hun93, Far99, Kim01]. Most algorithms operate on the ridge skeleton, verifying the validity of minutiae that are extracted using standard minutiae extraction algorithms. Heuristics have been constructed that are able to remove minutiae that originate from frequently occurring defects such as ridge-breaks, bridges, spurs, short ridges, and islands or holes, which are shown in Figure 5.7.

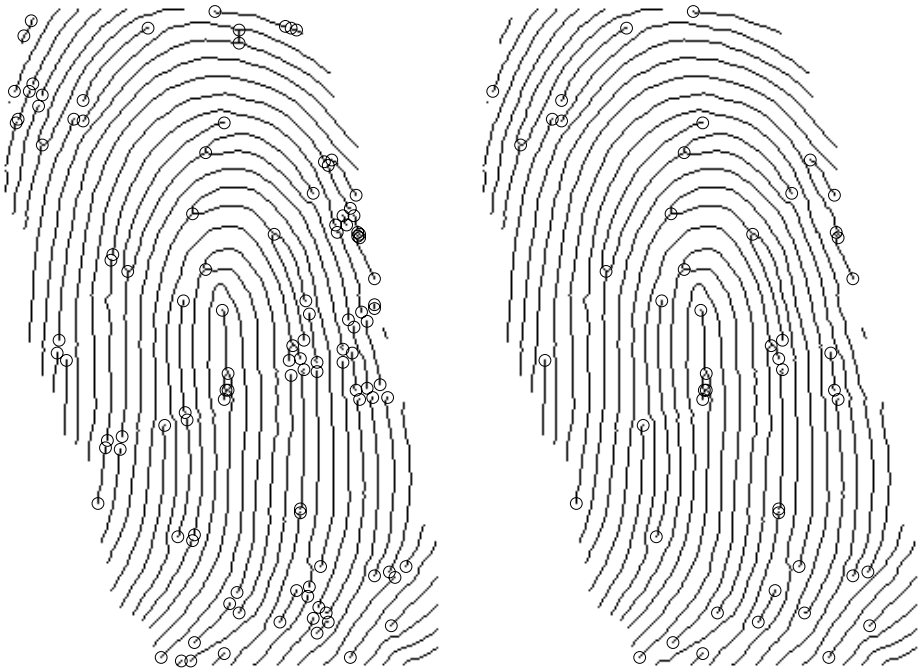
For instance, to repair a broken ridge, the heuristic 'two endpoints are connected if they are closer than a specified distance and facing each other' can be used. Such a heuristic is constructed for each type of false minutiae structure. The most important tool for these heuristics is the tracing of skeleton ridges. An example of a skeleton image and its minutiae before and after postprocessing is shown in Figure 5.8.

Another approach is the verification of minutiae in the original gray-scale image by means



(a) Broken ridge (b) Bridge (c) Spur (d) Short Ridge (e) Hole

Figure 5.7: Examples of false minutiae shapes.



(a) Before postprocessing (b) After postprocessing

Figure 5.8: Skeleton image and extracted minutiae before and after postprocessing.

of neural networks. Again, the potential minutiae are first detected by a standard minutiae extraction algorithm. The minutiae neighborhood is then taken from the gray-scale image and normalized, for example with respect to orientation, and enhanced. In [Pra00], this 32 by 32 neighborhood is directly fed to a learning vector quantizer (LVQ) or a Kohonen neural network. In [Mai99] the neighborhood is first reduced to a feature vector of 26 elements by means of a Karhunen-Loève transform, after which it is classified by a multi-layer perceptron.

5.3 Minutiae Extraction Using Genetic Programming

In this section, an image exploring agent is developed by means of genetic programming. The goal of the agent is to follow the ridge lines in the fingerprint and to stop at minutiae. First, genetic programming is described shortly in Section 5.3.1. Next, in Section 5.3.2, experiments are presented where an agent learns to perform its task.

5.3.1 Genetic Programming

Genetic programming (GP) [Koz92, Koz94, Koz99] is a method to automatically search for computer programs that perform a specific task, using the principle of survival of the fittest. GP works with a population of individuals: programs that can possibly solve the given problem. The first generation of this population is created randomly. Then, all programs are evaluated by applying them to typical problem instances. This results in a fitness measure being assigned to each program. The programs that have the highest fitness, have the highest probability to participate as parents in the creation of new individuals in the next generation. This process repeats until a satisfactory solution is found or a fixed number of generations is reached. When executed on a single PC, solving a real-life problem may take several days of processing time.

For the application of GP to the minutiae extraction problem, we have developed our own GP environment. This GP environment is called *Poor Man's Distributed Genetic Programming* (PMDGP), as it is designed to make use of existing heterogeneous hardware rather than expensive dedicated homogeneous hardware that has to be purchased for the purpose of GP. The environment performs tasks like: population management, genetic operators and evaluation of the programs. Furthermore, it provides a framework for implementation of the problem-specific part. Using object-oriented methods, the environment is designed to offer a high degree of flexibility and ease of use. The GP environment uses distributed fitness evaluation, which can be used on existing computer networks. The system is optimized for efficiency of distribution. Detailed descriptions of PMDGP can be found in [Meu00, Meu01].

5.3.2 Experiments and Conclusions

The goal of the experiments is to let GP develop an *image exploring agent* (IEA) [Köp99]. An IEA walks through a picture by continuously repeating its program and will terminate when it has found a region of interest (ROI). In this case the ROI is a ridge ending. So when

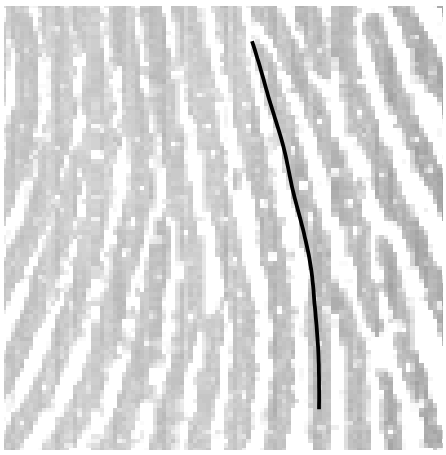


Figure 5.9: Path of the agent, which follows the ridge upwards and stops at the endpoint.

an IEA is placed on a ridge of the fingerprint, it should follow that ridge until it finds an endpoint of that ridge. This problem is in principle well suited for a learning algorithm like GP, because it can be trained by manually marking the endpoints.

During training all GP programs are evaluated on two different fingerprints. On each fingerprint five endpoints have been marked. For each endpoint two starting points have been marked. So each program is tested with 20 searches for endpoints. The maximum number of generations is 50 and each generation consists of a population of 5000 individuals. This process takes a lot of processing time, so parallelism can be used well. The fitness value of a program is calculated by summing the 20 distances between the manually marked endpoints on the pictures and the position where the IEA terminates.

The program mainly follows the directional field and does not move when it detects an ending. This simple approach gives good results for the training and testing, as can be seen in Figure 5.9. However, because of its simple nature, it does not provide robust minutiae extraction in lower quality fingerprints.

It can be concluded that the development of an image exploring agent for the direct gray-scale minutiae extraction from fingerprints by means of genetic programming was unsuccessful. For robustness, the IEA needs many more inputs than just a few pixel values in the neighborhood. However, GP is rather unsuited for handling large quantities of input variables.

5.4 Minutiae Extraction Using Reinforcement Learning

In this section, the design of an agent that extracts the minutiae from a gray-scale fingerprint image by means of reinforcement learning is discussed. Despite the results of Section 5.3, it has been shown that it is a good policy to follow the ridges in the fingerprint until a minutia is found. Maio and Maltoni [Mai97] presented an agent that takes small steps along the ridge. Jiang et al. [Jia01], enhanced the agent by using a variable step size and a directional filter

for noise suppression. This results in a rather complex system, especially since robustness is required. A much simpler solution is to use a agent that *learns* the task. By using a variety of training examples, a robust system can be obtained. In the approach of this section, the agent is trained by means of *reinforcement learning* (RL).

In this section we show that reinforcement learning can be used for minutiae detection in fingerprint matching. We propose a more robust approach, in which an autonomous agent walks around in the fingerprint and learns how to follow ridges in the fingerprint and how to recognize minutiae. The agent is situated in the environment, the fingerprint, and uses reinforcement learning to obtain an optimal policy. Multi-layer perceptrons are used for overcoming the difficulties of the large state space. By choosing the right reward structure and learning environment, the agent is able to learn the task. One of the main difficulties is that the goal states are not easily specified, for they are part of the learning task as well. That is, the recognition of minutiae has to be learned in addition to learning how to walk over the ridges in the fingerprint. Results of successful first experiments are presented.

The rest of this Section is organized as follows. In Section 5.4.1, reinforcement learning is explained. In Section 5.4.2, it is discussed how to apply RL to the problem of minutiae extraction. Finally, in Section 5.4.3, some experimental results are presented.

5.4.1 Reinforcement Learning

Reinforcement learning (RL) [Kae96, Sut98] is a learning technique in domains where there is no *instructive* feedback, as in supervised learning, but only *evaluative* feedback. Agents are trained by rewarding (both positive and negative), expressed in terms of real numbers.

In short, a RL agent learns in the following way. First, it perceives a *state* s_t. Then, on the basis of its experience, it chooses its best *action* or, with a small probability, a random action, action a_t. This action is rewarded by the environment with *reinforcement* r_t, after which the agent perceives the newly entered state s_{t+1}. This *interaction* continues until the agent enters a *terminal state*, i.e. its *episode* ends. Terminal states are either states in which the agent's goal is satisfied, or states in which the episode is terminated externally, possibly because of an illegal action. One of the main difficulties is that non-zero rewards are usually sparse and are sometimes only given at the end of the episode.

The goal of the agent is to maximize its reward by learning the optimal *policy*, i.e. a mapping from states to actions. This can be done by learning *value functions*. Value functions reflect the *expected cumulative reward* the agent receives by following its policy. In RL, one generally uses two kinds of value functions: $V : S \rightarrow \mathbb{R}$ (state values) and $Q : S \times A \rightarrow \mathbb{R}$ (state-action values). In this section we use the latter to reflect the *expected cumulative reward resulting from executing action a in state s and thereafter following the policy*. This mapping is approximated on the basis of interaction with the environment. The policy can easily be found by choosing in each state the action that maximizes the Q-value, i.e. the *greedy* action.

Q-Learning is a commonly used *off-policy* algorithm for learning the action values. In this section, however, we use the related algorithm Sarsa[1] [Rum94, Sut98], which is an *on-policy* temporal difference (TD) algorithm:

[1] Sarsa stands for ***S**tate-**A**ction-**R**eward-**S**tate-**A**ction*, which are the necessary elements for performing the update

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{5.1}$$

In this algorithm, $Q(s_t, a_t)$ is updated in the direction $[r_t + \gamma Q(s_{t+1}, a_{t+1})]$ with learning rate α. The *discount factor* γ determines how future rewards are weighted. The next state and action, s_{t+1} and a_{t+1}, are determined by the policy. The update of the approximation of one state-action pair uses the approximation of other state-action pairs. This is called *bootstrapping*.

For selection of the actions that are taken, ϵ-greedy action selection is used. This selection criterion chooses an exploratory action with probability ϵ and the greedy action, having the highest Q-value, otherwise.

Usually, when the state-action space is reasonably small, we can store all the Q-values in a simple lookup-table. Since the state-space in our problem has a high dimension, a function approximator for storing the values is used. The function approximator has one continuous output, $Q(s, a)$. We use a *multi-layer perceptron* (MLP) neural network for the approximator. At each step, the Q-function is updated by backpropagating the error in the right-hand side of (5.1).

For Sarsa, the update of the weights $\mathbf{w}$ of the neural network is given by:

$$\Delta \mathbf{w} = \eta \left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \nabla_{\mathbf{w}} Q(s_t, a_t) \tag{5.2}$$

where η is the learning rate and $\nabla_{\mathbf{w}} Q(s_t, a_t)$ is a vector of output gradients.

Training the neural network is performed by offering examples $((s_t, a_t),\ r_t + \gamma Q(s_{t+1}, a_{t+1}))$ to the neural network. These examples can be obtained by letting the agent interact with its environment such that the neural network determines the agent's actions. The training itself can be done either on-line [Rum94] or off-line [Lin92]. In this section, we will use on-line adaptation of the network.

5.4.2 RL for Minutiae Detection

This section describes how RL can be applied to the minutiae extraction problem. The *goal* of the agent is to follow a ridge and stop at a minutia. Furthermore, the minutia should be found in as few steps as possible in order to minimize the computational time needed for minutiae extraction. This is a typical example of an *episodic task*, which terminates when a minutia is found.

The agent only has a local view of the fingerprint image around it, e.g. it is *situated* in its environment. It can observe the gray scale pixel values in a segment of $n \times n$, for instance 12×12, pixels around it. The orientation of the agent is normalized by rotating the local view, such that the forward direction is always along the ridge-valley structures. For this purpose, the directional field is used (see Chapter 2). The local view is illustrated in Figure 5.10. This

in Eq. 5.1

Figure 5.10: Situatedness: local view of the agent of 12×12 pixels.

introduces considerable *a priori* knowledge. The agent is always aligned with the directional field, and this puts restrictions on the state space of the agent. Local views in which the agent has a direction other than approximately aligned with the ridges in the fingerprint do not occur.

As was explained in Section 5.4.1, the dimensionality of the state space requires function approximation which is e.g. provided by a multilayer perceptron. Furthermore, the length of the feature vector, which contains the pixel values in the local view, is reduced by a *Karhunen-Loève transformation* (KLT) [Jai89]. The KLT transforms feature vectors to a new basis that is given by the eigenvectors of their covariance matrix. Then, only those KL components that correspond to the largest eigenvalues are selected. This way, the largest amount of information is preserved in the smallest transformed feature vector. This has two useful effects. First, the length of the feature vector is reduced, which simplifies the learning process. Second, the KL components that carry the least information and therefore represent the noise, are discarded. This is beneficial since we are only interested in the rough structure of the local view, an not in the small details. This noise suppression method eliminates the need of a directional filter, which would require approximately 5 times as many computations.

The *actions* of the agent are the moves that it can make in the coordinate system of its local view. Since the agent is always approximately aligned with respect to the directional field, the forward action is always a move along the ridge-valley structures, and the agent does not need rotational or backward actions. To be able to achieve its goals, which is moving as fast as possible along the ridges and stopping at minutiae, the agent may move up to 4 pixels forward at each step and up to 1 pixel to the left or to the right. The left-right actions are necessary for keeping on the ridge. Although the directional field puts the agent in the right direction, it is not sufficient to keep the agent exactly on the ridges. The action results in a new position in the fingerprint, after which a new local view is extracted at that position.

The *reward structure* is another key element in the definition of an RL experiment. It determines the optimal actions for the agent to take. Setting up the right reward structure is very important for learning the (right) task. First, the agent receives rewards for staying at the center of a ridge. This is implemented by manually marking the ridge centers and using an exponential Gaussian function of the distance of the agent to the ridge center. Second, a much higher reward is given near the endpoints to be detected. Again a Gaussian function

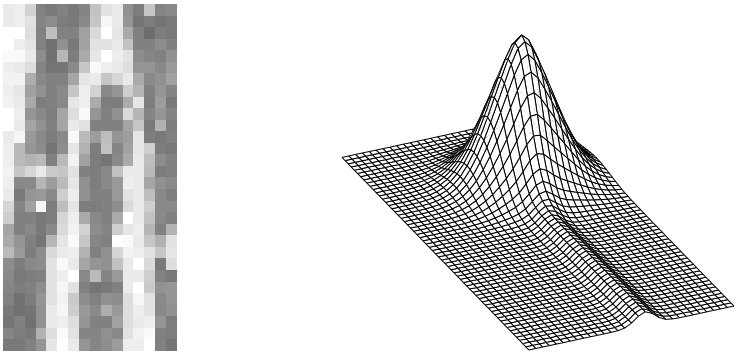


Figure 5.11: Reward structure around a ridge with an endpoint.

of the distance to the minutia is used. This structure encourages the agent to move in as few steps as possible to the endpoint by following a ridge and then to stop moving in order to receive the higher reward forever. This is enforced even more by scaling the reward such that large steps along the ridge receive a higher reward than small steps, while the opposite applies at endpoint locations. The reward structure around a ridge with an endpoint is shown in Figure 5.11. The peak in the reward corresponds to the endpoint of the ridge and the line of higher reward correspond to the ridge in the fingerprint.

The agent can be *trained* by selecting a ridge, choosing the initial position of the agent at some location on that ridge, and defining the reward structure with respect to that ridge, including the target minutia that the agent should find. Then, the agent starts moving and the network is updated until the episode ends when the agent moves too far from the target ridge. It is worth noticing that during one episode the agent is trained to follow a certain ridge and it only gets rewards for being near that particular ridge. Therefore, the reward structure is different for each different ridge that is used for training.

During *testing* and actual *use*, agents are released at a large number of positions in the image, for instance on a regular grid. They start moving according to their policy and follow the ridges. However, there is no clear *termination* criterion for the agents, since it is not known in advance which minutia they should find. To overcome this problem, endpoints are detected by counting the number of successive small steps. After 5 small steps, the agent is terminated and an endpoint is detected. Bifurcations are detected by keeping a map of the trajectories of all agents. When an agent intersects the path of another one, the agent is terminated and a bifurcation is detected.

5.4.3 Experimental Results

In this section, the setup of the training experiments and its results are presented. The training is performed on the fingerprint image of Figure 5.1. In this fingerprint, all ridges and all minutiae have been marked manually. For each episode, one ridge is selected and the agent is released a random position on that ridge. Then, the reward structure is calculated for that ridge as explained in Figure 5.11. Along the ridge, the Gaussian reward structure has parameters $\sigma^2 = 1$ and amplitude 1, while at a minutiae, $\sigma^2 = 10$ and the amplitude is 10.

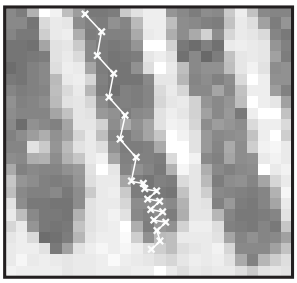


Figure 5.12: Path that was found by an agent.

Figure 5.13: Extracted minutiae.

Next, the agent is trained by the Sarsa algorithm as explained in Section 5.4.1. Since this is the training phase, the goal of the agent is known, and a clear termination criterion can be defined. Therefore, the agent is terminated if it is more than 7 pixels from the indicated ridge center.

A multitude of experiments have been performed to find the optimal setup of the algorithm. At this stage, no definitive parameter values can be given, although some numbers are presented here to give a first impression. For one training session, 50,000 episodes were used. During training, the exploration parameter decreases from $\epsilon = 0.01$ to $\epsilon = 0$. The size of the local view of the agent was taken as 12×12 pixels, which was reduced to a feature vector of length 50 by the KL transformation. The actions were constrained to a grid of discrete values up to 1 pixel to the left or to the right and up to 4 pixels forward. The multi-layer perceptron had 1 hidden layer of 22 neurons and the learning rate was $\eta = 10^{-2}$. This low value for η was necessary to deal with contradicting examples. The discounting factor was set to $\gamma = 0.9$.

Testing was performed on other fingerprints as described in Section 5.4.2. Starting points have been selected at a regular grid and the agents follow their policy until termination. Human inspection of the results, shown in Figures 5.12 and 5.13, indicates that the agent follows the ridges and that all minutiae have been detected. The agent takes relatively large steps along the ridges, while the step size decreases near the endpoints. Furthermore, it intersects its own path at bifurcations. However, the figure also shows a number of false minutiae. These might be eliminated by further training of the agent on other fingerprints and fine-tuning of the parameters. Another possibility is the application of post-processing techniques to eliminate false minutiae structures.

5.4.4 Conclusions

In this section we showed that reinforcement learning is a useful and intuitive way to tackle the problem of robust minutiae extraction from fingerprints. This new combination is now in its proof-of-concept phase. It has been shown that an adaptive agent can be trained to walk along the ridges in a fingerprint and mark minutiae when encountered. The system uses straightforward reinforcement learning techniques. There is still much room for fine tuning parameters and algorithms. An experimental study on a variety of parameters and other learning algorithms, like Q(λ)-learning, is recommended.

The use of *value-based RL* algorithms can turn out to be not the best choice in our application. The similar local views on different places on the ridges create a kind of *partially-observable* state representation, which did not cause severe problems in our case though. It might be better to search directly for a policy by using *policy gradient* [Sut00] methods or to use relative Q-values instead.

5.5 Evaluation of Minutiae Extraction Algorithms

Unfortunately, no common benchmarks are available in the literature for assessing the quality of minutiae extraction methods. To fill this lack of a common procedure, a method is proposed in this section that can be used well in practice. It does not require manual labelling of ground truth minutiae, it is based on the functional requirements to minutiae extraction algorithms, i.e. it evaluates the consistency of the extracted minutiae over different prints of the same finger, and it provides maximal feedback of the types of extraction errors that are made. For each extracted minutia, it indicates whether it is a false, missed, displaced or genuine minutia. First, the two naive methods are discussed in Section 5.5.1. Then, a new method is proposed in Section 5.5.2.

5.5.1 Naive Methods

At first thought, two simple methods, both featuring other disadvantages, can be used. The first method is to extract all minutiae from a large test set of fingerprints manually. The minutiae extraction algorithm is applied to this set, and the number of false and missed minutiae

is counted. Additionally, statistics of the displacements of genuine minutiae can be collected. This method has two clear disadvantages. First, it involves a lot of manual labelling, which is not the most favorite job of many researchers. Second, this method selects the algorithm that best approximates the subjective, manually determined, ground truth. However, this particular algorithm may not achieve the best matching performance. For matching purposes, consistency of the extracted minutiae is more important.

The second method is to compare different minutiae extraction algorithms functionally. The minutiae extraction algorithms are evaluated by combining them with the same minutiae matching algorithm. The combination that achieves the best fingerprint matching performance apparently used the best minutiae extraction algorithm. Again, two disadvantages of this method can be identified. First, the evaluation of a small change in the minutiae extraction algorithm requires an extensive matching experiment, which may take several hours. Second, the only feedback of this method is which algorithm is superior, without details of the numbers and types of minutiae extraction errors that made in specific situations.

5.5.2 Elastic Method

The method that is proposed here, combines parts of both methods that are described above. It first extracts the minutiae sets from a number of different prints of the same finger. Next, the minutiae sets are matched by the elastic minutiae matching algorithm that is proposed in Chapter 6. Finally, correspondences and differences between the elastically registered minutiae sets are used to classify the minutiae as false, missed, displaced or genuine. This method cannot be used without the elastic matching algorithm, which compensates for elastic deformations. For a rigid matching algorithm, minutiae displacements due to elastic deformations would dominate the errors in the minutiae extraction.

The proposed method combines the advantages of both naive methods, without being subject to their disadvantages. It does not require manual labelling of ground truth minutiae, it is based on the functional requirements to minutiae extraction algorithms, i.e. it evaluates the consistency of the extracted minutiae over different prints of the same finger, and it provides maximal feedback of the types of extraction errors that are made: for each extracted minutia, it indicates whether it is a false, missed, displaced or genuine minutia.

The different minutiae extraction methods that are discussed in this chapter, have not yet been optimized and evaluated extensively using the proposed method. Instead, initial estimates of the parameters have been determined by manual inspection, while some fine-tuning has been done using matching performance experiments. However, for more extensive evaluation and fine-tuning, the proposed method should be used.

5.6 Conclusions

At the moment, traditional minutiae extraction is the preferred method, compared to other approaches. The approach that has been proposed in this chapter combines an efficient implementation of Gabor enhancement, binarization, thinning, extraction and heuristic postpro-

cessing, which leads to satisfactory results. Enhancement techniques and segmentation to remove background and low-quality areas remain critical issues to be solved. Other challenges may include the reconstruction of low-quality areas for minutiae extraction from those regions.

Two alternative minutiae extraction methods have been investigated. The approach that uses reinforcement learning performs at a lower level than the traditional method, while the approach that uses genetic programming is not robust at all.

Part II

Matching

Chapter 6

Elastic Minutiae Matching Using Thin-Plate Splines

Abstract

This chapter presents a novel minutiae matching method that describes elastic distortions in fingerprints by means of a thin-plate spline model, which is estimated using a local and a global matching stage. After registration of the fingerprints according to the estimated model, the number of matching minutiae can be counted using very tight matching thresholds. For deformed fingerprints, the algorithm gives considerably higher matching scores compared to rigid matching algorithms, while only taking 100 ms on a 1 GHz P-III machine. Furthermore, it is shown that the observed deformations in practice are different from those described by the theoretical model that is proposed in the literature. Parts of this chapter have been published in [Baz02e], [Baz02b], and [Baz02c].

6.1 Introduction

The topic of this chapter is fingerprint matching, which is the task of comparing a *test* fingerprint that is provided by the user, to a *template* fingerprint that is provided earlier, during enrollment. Most fingerprint matching systems are based on the *minutiae*, which are the endpoints and bifurcations of the elevated line structures in the fingerprint that are called ridges. A minutiae-based fingerprint matching system roughly consists of two stages. In the *minutiae extraction* stage, the minutiae are extracted from the gray-scale fingerprint, while in the *minutiae matching* stage, two sets of minutiae are compared in order to decide whether the fingerprints match. This chapter deals with the compensation of elastic distortions to improve the performance of minutiae matching.

In minutiae matching, two stages can be distinguished. First, *registration* aligns both fingerprints as well as possible. Most algorithms use a combination of translation, rotation and scaling for this task. In the rest of this chapter, one global transformation for the whole fingerprint that is based on rotation, translation and scaling only will be called *rigid*. A non-rigid transformation for registration will be called *elastic*. After registration, the matching score is determined by *counting* the corresponding minutiae pairs that are present in both fingerprints. Two minutiae correspond if a minutia from the test set is located within a *bounding box* or *tolerance zone* around a minutia from the template set. The matching score, which is a number in the range from 0 to 1, is calculated as the number of matched minutiae divided by the total number of minutiae.

Unfortunately, there are a lot of complicating factors in minutiae matching. First of all, both sets may suffer from false, missed and displaced minutiae, caused by imperfections in the minutiae extraction stage. Second, the two fingerprints to be compared may originate from a different part of the same finger, which means that both sets overlap only partially. Third, the two prints may be translated, rotated and scaled with respect to each other. The fourth problem is the presence of non-linear elastic deformations (also called plastic distortions) in the fingerprints, which is the most difficult problem to solve. The method that is proposed in this chapter solves all these problems, while specifically addressing the elastic deformations.

This chapter is organized as follows. First, in Section 6.2, elastic deformations are discussed and an overview is given of other matching methods that try to deal with them. Next, in Section 6.3, an elastic minutiae matching algorithm is proposed that estimates a deformation model and uses this model for improved minutiae matching. Finally, in Section 6.4, experimental results of the elastic minutiae matching algorithm are given.

6.2 Elastic Deformations

Elastic distortions are caused by the acquisition process itself. During capturing, the 3-dimensional elastic surface of a finger is pressed onto a flat sensor surface. This 3D-to-2D mapping of the finger skin introduces non-linear distortions, especially when forces are applied that are not orthogonal to the sensor surface. This is especially a realistic situation when dealing with non-cooperative users that deliberately apply excessive force in order to create

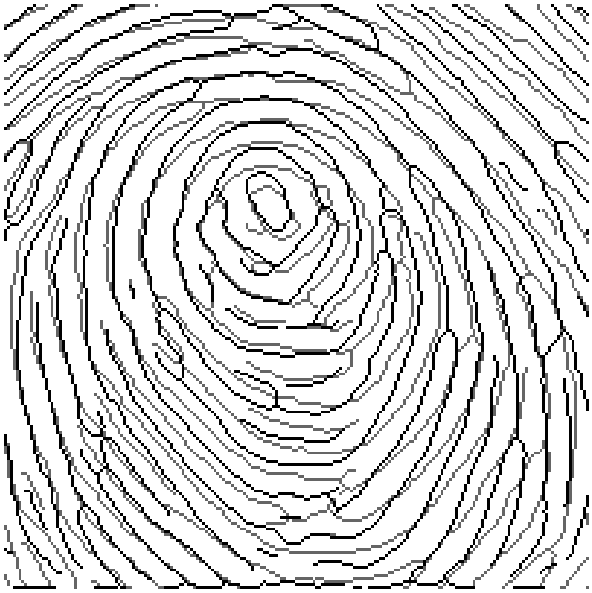


Figure 6.1: Ridge skeletons of elastically distorted fingerprints that are registered by means of the rigid algorithm.

intentional elastic deformations. The effect is that the sets of minutiae of two prints of the same finger no longer fit exactly after rigid registration. This is illustrated in Figure 6.1 where the ridge skeletons of two prints of the same finger have been registered optimally (using the minimum sum of squared minutiae distances) and displayed in one figure.

In order to tolerate minutiae pairs that are further apart because of elastic distortions, and therefore to decrease the *false rejection rate* (FRR), most algorithms increase the size of the bounding boxes [Pan01]. However, as a side effect, this gives non-matching minutiae pairs a higher probability to get paired, resulting in a higher *false acceptance rate* (FAR). Therefore, changing the size of the bounding box around minutiae only has the effect of exchanging FRR for FAR, while it does not solve the problem of elastic distortions. An alternative approach is to use only local similarity measures, as addressed in Section 6.3.1, since those are less affected by elastic distortions [Kov00, Rat00]. However, this also decreases the required amount of similarity, and therefore also exchanges FRR for FAR.

Recently, some methods were presented that deal with the problem of matching elastically distorted fingerprints more explicitly, thus avoiding the exchange of error rates. The ideal way to deal with distortions would be to invert the 3D-to-2D mapping and compare the minutiae positions in 3D. Unfortunately, in the absence of a calibrated measurement of the deformation process, there is no unique way of inverting this mapping. It is therefore reasonable to consider methods that explicitly attempt to model and eliminate the 2D distortion in the fingerprint image. In [Sen01], a method is proposed that first estimates the local ridge frequency in the entire fingerprint and then adapts the extracted minutiae positions in such a way that the ridge distances are normalized all over the image. Although the stricter matching conditions slightly increase the performance of the matching algorithm, this method only

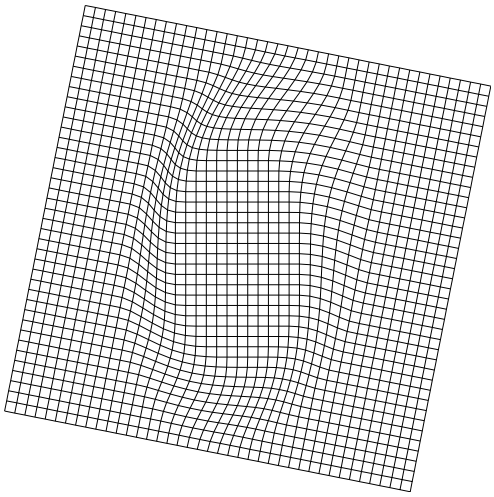


Figure 6.2: Elastic deformation model of [Cap01].

solves some specific part of the non-linear deformations.

Since it is not known in advance whether captured fingerprints contain any distortion, true normalization of the fingerprints to their genuine shape is not possible. The fact that no reference without distortion is available makes normalization in 2D a relative rather than an absolute matter. Instead of normalizing each fingerprint on its own, the non-linear distortions of one fingerprint with respect to the other have to be estimated and eliminated.

In [Cap01], the physical cause of the distortions is modelled by distinguishing three distinct concentric regions in a fingerprint. In the center region, it is assumed that no distortions are present, since this region tightly touches the sensor. The outer, or external, region is not distorted either, since it does not touch the sensor. The outer region may be displaced and rotated with respect to the inner region, due to the application of forces while pressing the finger at the sensor. The region in between is distorted in order to fit both regions to each other, as shown in Figure 6.2. Experiments have shown that this model provides an accurate description of the elastic distortions in some cases. The technique has successfully been applied to the generation of different synthetic fingerprints of the same finger [Cap00a]. However, the model has not yet been used in an algorithm for matching fingerprints. Accurate estimation of the distortion parameters is still a topic of research.

Furthermore, a number of non-rigid registration methods have been proposed in other fields than fingerprint matching, see e.g. [Chu00, Kum01]. Most of these methods, however, suffer from two problems. First, they assume that the correspondences between two sets of points are known from the local image structure around these points, which is not a realistic assumption in fingerprint matching where all minutiae points look similar. In addition, they require much processing time, ranging from 1 minute to several hours. In fingerprint matching the available time is limited to only a few seconds. Therefore, these methods cannot be applied to the fingerprint matching problem.

In this chapter, a minutiae matching algorithm is presented that models elastic distortions

by means of a thin-plate spline model, based on the locations and orientations of the extracted minutiae. This model is used to normalize the shape of the test fingerprint with respect to the template. It is therefore able to deal with elastically distorted fingerprints. As a consequence, the distances between the corresponding minutiae are much smaller and tighter bounding boxes can be used in the counting stage. This results in an algorithm that is able to decrease FAR and FRR simultaneously.

6.3 The Matching Algorithm

The elastic minutiae matching algorithm estimates the non-linear transformation model in two stages. The local matching that is presented in Section 6.3.1 determines which minutiae could possibly form a matching pair, based on local similarity measures. Without the local matching stage, each minutia in the test set could correspond to each minutia in the template set. This means that a problem with too many degrees of freedom would have to be solved in the subsequent global matching stage, that is presented in Section 6.3.2. The global matching stage uses the possible correspondences to estimate a global non-rigid transformation that is used to register the two fingerprints. After registration, the corresponding minutiae are counted using bounding boxes that can be chosen rather strictly since the distance between corresponding minutiae after elastic registration is small.

6.3.1 Local Matching

The first step in the proposed matching algorithm is the comparison of local structures. These structures can be compared easily since they contain few minutiae in a small area. In addition, since the structures originate from only a small area in a fingerprint, they are unlikely to be seriously deformed by elastic distortions. The local matching algorithm was inspired by the approach that was described in [Jia00].

Each minutia m in the template and test fingerprints is described by parameters (x, y, θ), where (x, y) are the pixel coordinates of the minutia and θ is the orientation of the minutia. The orientation is estimated by tracing the ridges that leave the minutia (1 ridge for an endpoint and 3 ridges for a bifurcation) over some distance and quantizing the obtained ridge directions to either the directional field [Baz02d] or the opposite direction. For an endpoint, this directly gives the orientation and for a bifurcation, the orientation that occurs twice is selected. This is illustrated in Figure 6.3. The matching algorithm does not distinguish between types of minutiae, because these can be easily interchanged by noise or pressure differences during acquisition. However, their orientations remain unchanged when this occurs.

Each minutia defines a number of local structures, which are called *minutia neighborhoods*. A minutia neighborhood consists of the minutia itself and two neighboring minutiae. When the reference minutia is called m_0 and its closest neighbors with increasing distance from m_0 are $m_1, m_2, \ldots, m_{n-1}$, with n the number of minutiae, the neighborhoods $\{m_0, m_1, m_2\}$, $\{m_0, m_1, m_3\}$ and $\{m_0, m_2, m_3\}$ are selected for each minutiae. Compared to selecting only one neighborhood, this provides more robustness in the local matching stage with respect to false and missing minutiae.

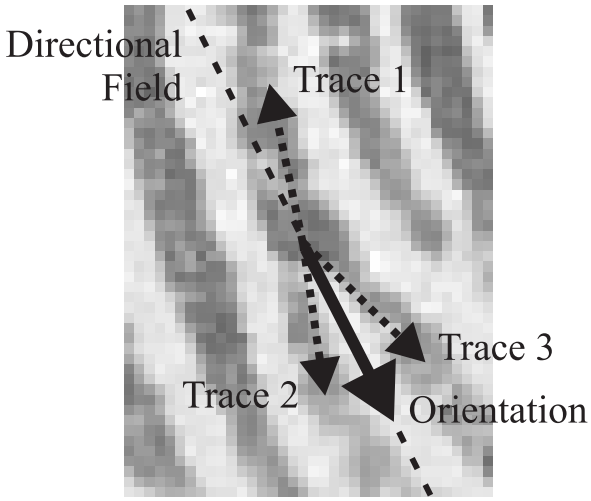


Figure 6.3: Estimation of the minutiae orientation.

The local matching algorithm compares each minutia neighborhood in the test fingerprint to each minutia neighborhood in the template fingerprint. First, the two structures are aligned using a least squares algorithm that determines the optimal rotation, translation and scaling. Next, the scaling, the sum of the squared distances between the corresponding minutiae and the differences of orientation are used to measure the similarity of the two minutia neighborhoods. If the structures are considered to match, the pair of minutia neighborhoods and the transformation (t, r, s), consisting of translation $t = (t_x, t_y)$, rotation r and scaling s, is stored.

After each minutia neighborhood in the test fingerprint has been compared to each minutia neighborhood in the template fingerprint, a list of corresponding minutia neighborhood pairs is obtained. Note that this list does not contain all true correspondences and that inclusion in this list does not necessarily indicate a true correspondence. However, its size gives a first indication of the degree of similarity of the two fingerprints.

6.3.2 Global Matching

The next step is the determination of the global transformation that optimally registers the two fingerprints. This is called the global matching stage. From the list of local similarities, the global transformation is determined that is consistent with the largest number of matching minutia neighborhood pairs. In general, this transformation also selects the largest number of matching minutiae pairs from the entire minutiae sets.

Several strategies to determine the optimal global transformation from the list of local transformations exist. Those methods mainly differ in handling the difficulty of false and contradictory local matches. In [Jia00], the transformation of the single minutia neighborhood pair that matches best, is taken. However, using more information of other local matches will certainly improve the accuracy of the registration. Another possibility is to quantize the

local registration parameters into bins and construct an accumulator array that counts all occurrences of each quantized registration. Next, the bin that occurs most is selected, and the average of all registrations in that bin is taken. This strategy roughly corresponds to the work of [Jai97b], although that method is not based on matching local minutia neighborhoods.

The method that is proposed here, achieves further improvement by determining the optimal registration parameters from the positions of the matching minutia neighborhoods instead of averaging the individual registration parameters. First, the largest group of pairs that share approximately the same registration parameters is selected. This is achieved by determining for each matching pair the number of pairs of which the registration parameters differ less than a certain threshold. Next, the transformation (t, r, s) that optimally registers the selected minutiae in the test set to the corresponding minutiae in template set is calculated in a least squares sense.

However, when applying this registration, elastically deformed fingerprints will not be registered well, as shown in Figure 6.1, simply because an accurate rigid registration (t, r, s) does not exist. This has to be compensated for in the counting stage. It has been reported in [Pan01] that for 97.5% of the minutiae to match, a threshold on the Euclidean distance of two minutiae of $r_0 = 15$ pixels has to be used in 500 dpi fingerprints. As a consequence, minutiae in a rather large part of the image (25% of the image for 30 minutiae in a 300×300 image) are considered to match even when they actually do not match.

In order to allow stricter matching, i.e. a smaller value of r_0, elastic registration has to be used to compensate for elastic distortions. A transformation that is able to represent elastic deformations is the *thin-plate spline* (TPS) model [Boo89]. This model has not been applied earlier to fingerprint recognition. The TPS model describes the transformed coordinates (x', y') both independently as a function of the original coordinates (x, y):

$$x' = f_x(x, y) \tag{6.1}$$
$$y' = f_y(x, y) \tag{6.2}$$

Given the displacements of a number of landmark points (for fingerprint recognition, the minutiae can be used as such), the TPS model interpolates those points, while maintaining maximal smoothness. The smoothness is represented by the bending energy of a thin metal plate. At each landmark point (x, y), the displacement is represented by an additional z-coordinate, and, for each point, the thin metal plate is fixed at position (x, y, z). The bending energy is given by the integral of the second order partial derivatives over the entire surface and can be minimized by solving a set of linear equations. Therefore, the TPS parameters can be found very efficiently. The TPS model for one of the transformed coordinates is given by parameter vectors $\mathbf{a} = [a_1\ a_2\ a_3]^T$ and $\mathbf{w} = [w_1\ \cdots\ w_n]^T$:

$$f(x, y) = a_1 + a_2 x + a_3 y + \sum_{i=1}^{n} w_i U(|P_i - (x, y)|) \tag{6.3}$$

where $U(r) = r^2 \log r$ is the basis function, $\mathbf{a}$ defines the affine part of the transformation, $\mathbf{w}$ gives an additional non-linear deformation, P_i are the landmarks that the TPS interpolates,

and n is the number of landmarks. More details on the estimation of thin-plate splines is given in Appendix E.

In [Roh99], a method is presented to estimate *approximating thin-plate splines*. These splines do not exactly interpolate all given points, but are allowed to approximate them in favor of a smoother transformation. The smoothness is controlled by a parameter λ, which weights the optimization of landmark distance and smoothness. For $\lambda = 0$, there is full interpolation, while for very large λ, there is only an affine transformation left. More details on approximating thin-plate splines is given in Appendix E.

In fingerprint matching, it is essential to use approximating thin plate splines, since this introduces some insensitivity to errors. For instance, minutiae may be displaced a few pixels by the minutiae extraction algorithm or false local matches may be included into the global matching stage. Interpolating TPS will include these displacement errors into the registration exactly, resulting in strange un-smooth transformations and incorrect extrapolations. Obviously, a smoother transformation that does not take all small details into account is much more robust. In that case, the TPS registration represents the elastic distortions, while in the counting stage, the threshold r_0 takes care of local minutiae displacements.

The TPS model is fitted in a number of iterations. First, an initial model is fitted to the minutiae in the minutia neighborhood pairs that were found in the local matching stage. Next, the corresponding minutiae in both sets, differing in location and orientation less than a threshold, are determined and a new model is fitted to those corresponding minutiae. This is repeated with a decreasing threshold r_0 until the model has converged to its final state. This iterative process improves the quality of the non-linear registration considerably and increases the matching score significantly. Finally, the matching score S is calculated by:

$$S = \frac{n_{\text{match}}^2}{n_1 \cdot n_2} \tag{6.4}$$

where n_{match} is the number of matching minutiae, n_1 the number of minutiae in the test fingerprint and n_2 the number of minutiae in the template fingerprint. This expression provides the best balance between false rejection and false acceptance for fingerprints that contain only a small number of minutiae. The match or non-match decision is then taken by comparing the matching score to a threshold.

Figure 6.4 shows the two deformed fingerprints after registration by means of thin-plate splines. The figure clearly shows the much more accurate registration with respect to the leftmost part. This means that a much lower threshold r_0 can be used in the counting stage, leading to a considerably lower false acceptance rate.

6.4 Experimental Results

The proposed algorithm has been evaluated by applying it to Database 2 of FVC2000 [Mai00, Mai02] and training Database 1 of FVC2002. The FVC2000 database consists of 880 capacitive 8-bit gray-scale fingerprints, 8 prints of each of 110 distinct fingers. The im-

Figure 6.4: Ridge skeletons of elastically distorted fingerprints that are registered by means of the thin-plate spline algorithm.

ages are captured at 500 dpi, resulting in image sizes of 364 $\times$ 256 pixels. The FVC2002 database contains 80 8-bit gray-scale fingerprints, 8 prints of 10 fingers, captured with an optical sensor at 500 dpi.

Unfortunately no benchmark results are available in the literature to measure the performance of minutiae-based matching for given fixed sets of minutiae. Any result reported on databases such as the ones of FVC2000 incorporate the performance of a minutiae extraction stage, which is not a topic of this chapter. Therefore, we used the traditional minutiae extraction method that was presented in Section 5.2.

First, the estimated elastic deformation models have been evaluated visually. In Figure 6.5, the registration results are depicted for typical and heavy distortions in the FVC2000 database. The figure shows the superposition of both minutiae sets (indicated by '$\times$' and '$\circ$') and a grid that visualizes the deformations. The figure clearly shows that elastic registration makes the minutiae sets fit much better, i.e. the corresponding '$\times$'s and '$\circ$'s are much closer to each other, while the fingerprints are not heavily distorted. For the upper row, the matching scores are 0.42 for the rigid registration and 0.68 for the TPS registration, while the scores for the bottom row are 0.04 and 0.25. Furthermore, it is worth noticing that all distortion patterns that we inspected were similar to the patterns that are shown in Figure 6.5, while none of them resembled the distortion model that was proposed in [Cap01] (see Figure 6.2).

Next, due to the lack of benchmark results for minutiae matching performance, it was decided to compare TPS-based elastic matching to rigid matching. In both cases, r_0 was chosen such that the matching performance was optimized. With $r_0 = 15$ for rigid matching and $r_0 = 5$ for elastic matching the equal-error rates of the ROC turned out to become 4% and 1.8% respectively for training Database 1 of FVC2002. In this experiment, 280 matches

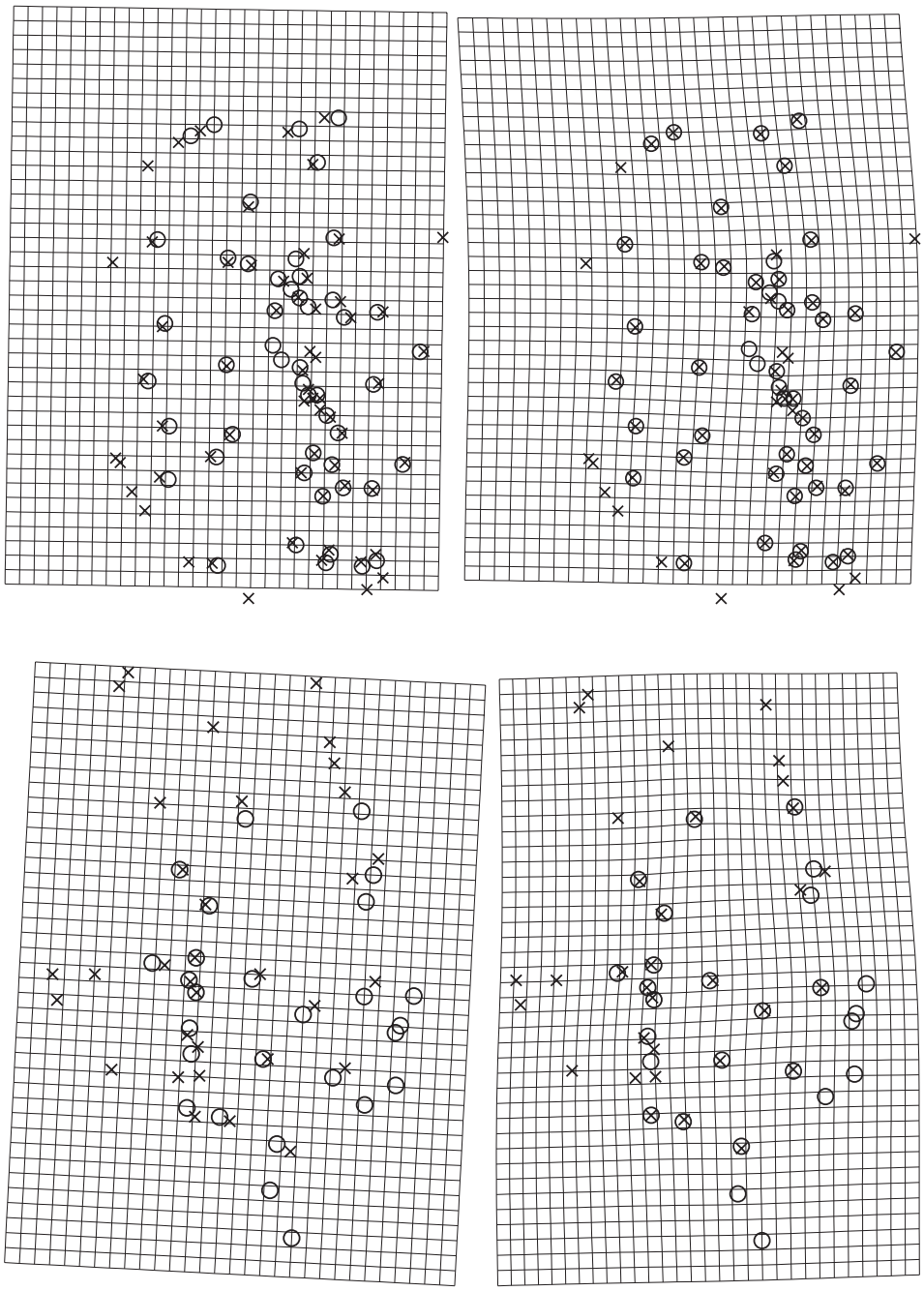


Figure 6.5: Registration results for the elastic matching algorithm: the superposition of both minutiae sets (indicated by '×' and 'o') and a grid that visualizes the deformations. Top row: typical distortions, bottom row: heavy distortions, left column: rigid registration, right column: TPS registration.

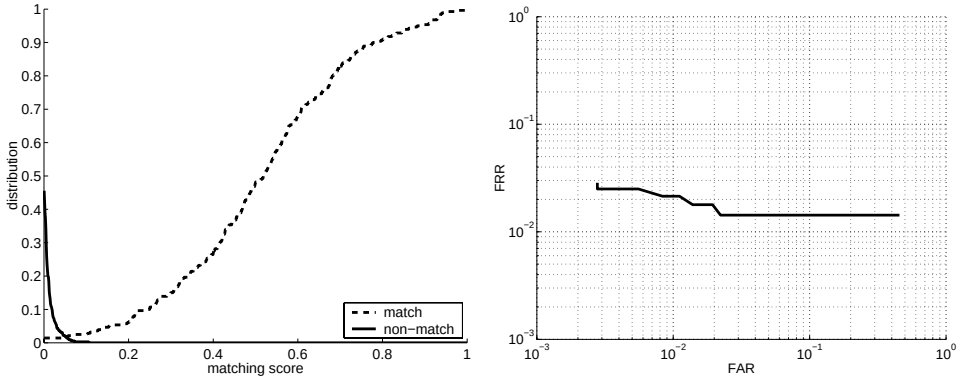


Figure 6.6: Results of the TPS-based matching algorithm on training database 1 of FVC2002. (a): matching score distribution, (b): ROC.

and 450 non-matches have been evaluated. The distributions of the matching scores and the ROC for the elastic matching experiment are shown in Figure 6.6.

To evaluate the matching performance of the elastic matching algorithm on database 2 of FVC2000, two experiments have been done. The first experiment considers all fingerprints in the database, leading to 3080 matches and 5995 non-matches. In this case, the equal error rate is 6%. In the second experiment, 10% of the fingers with lowest quality fingerprints have been left out. This compensates for the imperfections in the used minutiae extraction algorithm, and allows better evaluation of only the elastic matching algorithm. In this case, the equal error rate is 2.5%. The matching score distributions and ROCs are shown in Figure 6.7.

The combined local and global matching algorithm is able to keep the matching scores for non-matching fingerprints very low: values larger than $S = 0.1$ are never observed. However, some low matching scores have been observed for matching fingerprints, resulting in a fixed lower bound for FRR, relatively independent of the chosen FAR.

Analysis of these cases results in two possible causes for the low matching scores. The first cause is due to imperfections in the used minutiae extraction stage. Low-quality regions are discarded by the segmentation that precedes the minutiae extraction, leading to very few corresponding minutiae. It is expected that the matching performance increases when combined with better minutiae extraction algorithms. The second cause is a very small overlapping region between the two fingerprints that are matched. In this case too, there are only very few corresponding minutiae. This problem was partially solved by using only the minutiae in the overlapping region for the matching score determination, i.e. adjusting n_1 and n_2 in equation 6.4. However, this may also lead to higher non-matching scores.

Furthermore, it is worth noticing that none of the low matching scores is caused by the presence of elastic deformations. The algorithm correctly resolves these distortions, while in some cases, the rigid matching algorithm is not able to do so.

Finally, the proposed elastic matching algorithm is rather fast. In a C++ implementation

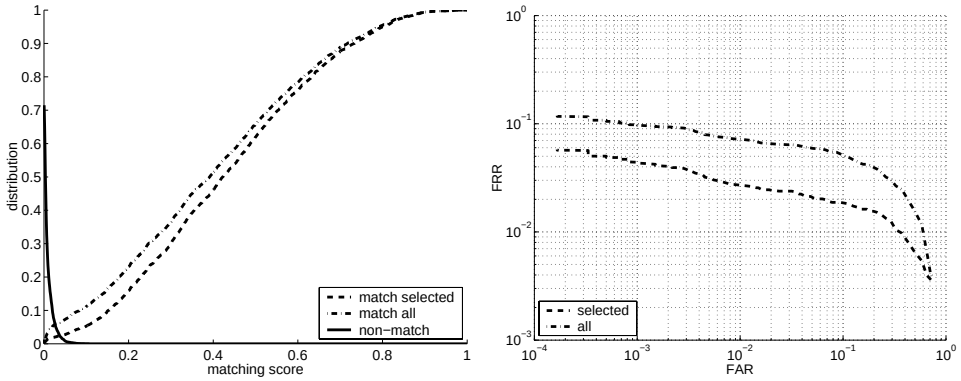


Figure 6.7: Results of the TPS-based matching algorithm on database 2 of FVC2000, with and without discarding low-quality fingerprints. (a): matching score distribution, (b): ROC.

on a 1 GHz P-III machine, the entire elastic minutiae matching algorithm takes less than 100 ms. Furthermore, it is only marginally more complex than the rigid matching algorithm. The local matching stage takes approximately 50 ms, rigid matching would take 10 ms and elastic matching takes 30 ms.

6.5 Conclusions

This chapter has proposed a novel minutiae matching algorithm that is able to deal with elastic distortions of fingerprints. Using thin-plate splines, the algorithm handles all non-linear distortions that might occur in fingerprints, while using very tight bounding boxes. It has been shown that it is able to register distorted fingerprints very well. When applied to elastically deformed fingerprints, the elastic matching algorithm provides considerably better matching scores than rigid matching algorithms. Since a relatively simple minutiae extraction algorithm was used, it is expected that the matching performance can be improved by linking the proposed matching algorithm to better minutiae extraction algorithms.

The algorithm detected only relatively small elastic deformations. Furthermore, we have given evidence that distortion patterns encountered in practice do not resemble much the models that were proposed in [Cap01].

Chapter 7

Likelihood Ratio-Based Biometric Verification

Abstract

This chapter presents results on optimal similarity measures for biometric verification based on fixed-length feature vectors. First, we show that the verification of a single user is equivalent to the detection problem, which implies that for single-user verification the likelihood ratio is optimal. Second, we show that under some general conditions, decisions based on posterior probabilities and likelihood ratios are equivalent, and result in the same ROC. However, in a multi-user situation, these two methods lead to different average error rates. As a third result, we prove theoretically that, for multi-user verification, the use of the likelihood ratio is optimal in terms of average error rates. The superiority of this method is illustrated by experiments in fingerprint verification. It is shown that error rates of approximately 10^{-4} can be achieved when using multiple fingerprints for template construction. This chapter will be published as [Baz02f].

7.1 Introduction

Biometric verification systems are used to verify the claimed identity of a user by measuring specific characteristics of the body, such as fingerprints, hand geometry, irises, or pressure signals. The verification system calculates the similarity between the measured characteristic and a template corresponding to the claimed identity. If the similarity is larger than an acceptance threshold, the user is accepted. Otherwise, the user is rejected.

Most fingerprint matching systems use minutiae-based algorithms [Baz02b, Jai97a], which are in general considered as most reliable. However, comparing two sets of minutiae is not a straightforward task. First, the number of minutiae that are extracted depends on the actual fingerprint. Second, it is not known beforehand which minutia in the template set corresponds to which one in the test set. Third, even if the first two problems are solved, the minutiae locations and associated characteristics cannot be compared directly due to translation, rotation, scaling, etc. of the fingerprints. Instead, comparing two sets of minutiae requires special point-pattern matching algorithms.

In contrast, this chapter presents a fingerprint matching algorithm that uses fixed-length feature vectors, consisting of a number of measurements that are performed at some specific, fixed, locations in the fingerprint. The advantage of this approach is that, once the features are extracted, the matching is very fast, which enables the search for a matching fingerprint in a large database.

Given a *test* feature vector $\mathbf{v}$ that is obtained from a user requesting access to a biometric system, and a class w_k that represents the users claimed identity (represented by a *template* feature vector), the task of a biometric verification system is to decide whether the offered feature vector can be accepted as a member of the given class or not. For this purpose, the system determines a measure that represents the similarity between the test and the template measurements, and the user is granted access to the system if the similarity measure exceeds a certain threshold. The subject of this chapter is the comparison of different similarity measures that can be used. We present results on optimal similarity measures for general biometric verification based on fixed-length feature vectors.

Various similarity measures for fixed-length feature vectors have been proposed in the literature. Here, we give an overview of the three most widely used measures, being Euclidean distance, posterior probabilities and likelihood ratios.

In [Jai00b], FingerCode is used as feature vector for fingerprint verification. This feature vector contains the standard deviation of the responses of Gabor filters in specified locations in the fingerprint. For comparison of these feature vectors, the Euclidean distance is used. For this reason this method treats all elements of the feature vector as equally important and uncorrelated. Although this is not a realistic assumption, the authors present experiments with relatively good recognition performance.

In [Gol97], biometric verification systems that are based on hand geometry and face recognition are presented. In that paper, it is claimed that decisions that are based on the posterior probability densities are optimal, where optimality means minimal error rates as defined in Section 7.3. The posterior probability density of class w_k given observed feature vector $\mathbf{v}$ is given by:

$$p(w_k|\mathbf{v}) = \frac{p(\mathbf{v}|w_k) \cdot p(w_k)}{p(\mathbf{v})} \tag{7.1}$$

where $p(\mathbf{v}|w_k)$ is the probability density of the feature vectors given class w_k, $p(w_k)$ is the probability density of the class w_k, and $p(\mathbf{v})$ is the prior probability density of the feature vectors. The feature vector $\mathbf{v}$ is accepted as member of the template class if its posterior probability density exceeds a threshold $t \in [0, t_{\max}]$.

On the other hand, in detection theory it is known since long that the use of likelihood ratios is asymptotically optimal [Tre68]. In detection, a given feature vector has to be classified as originating from a predefined situation (the presence of some object to be detected) or not. Since the detection problem is in some sense equivalent to the verification problem that is considered here, it is to be expected that using likelihood ratios for biometric verification is optimal as well. The likelihood ratio $L(\mathbf{v})$ is given by:

$$L(\mathbf{v}) = \frac{p(\mathbf{v}|w_k)}{p(\mathbf{v}|\overline{w_k})} \tag{7.2}$$

where $p(\mathbf{v}|\overline{w_k})$ is the probability of $\mathbf{v}$, given $\mathbf{v}$ is not a member of class w_k. Since we assume many classes, exclusion of a single class w_k does not change the distribution of the feature vector $\mathbf{v}$. Therefore, the distribution of $\mathbf{v}$, given $\mathbf{v}$ is not a member of w_k, equals the prior distribution of $\mathbf{v}$:

$$p(\mathbf{v}|\overline{w_k}) = p(\mathbf{v}) \tag{7.3}$$

and the likelihood ratio is given by

$$L(\mathbf{v}) = \frac{p(\mathbf{v}|w_k)}{p(\mathbf{v})} \tag{7.4}$$

In this framework, a test feature vector $\mathbf{v}$ is accepted as member of the template class if its likelihood ratio exceeds a threshold $t \in [0, \infty\rangle$. The acceptance region $A_{k,t}$ and rejection region $R_{k,t}$ can be defined in the feature space $\mathbf{V}$:

$$A_{k,t} = \{\mathbf{v} \in \mathbf{V} \mid L(\mathbf{v}) \geq t\} \tag{7.5}$$

$$R_{k,t} = \{\mathbf{v} \in \mathbf{V} \mid L(\mathbf{v}) < t\} \tag{7.6}$$

The probability density functions in Expressions 7.4 and 7.1 are in practice usually modelled by multidimensional Gaussian distributions. More details of that case are given in Appendix F.

This chapter focusses on likelihood ratio-based biometric verification, and on the differences in verification performance between posterior probability-based and likelihood ratio-based biometric systems. In Section 7.2, the general expressions for biometric system errors

are derived. Next, the optimality of likelihood ratio-based decisions is proved in Section 7.3. Then, Section 7.4 presents experimental results on fingerprint verification, confirming that using likelihood ratios instead of posterior probability densities decreases the error rates, even though the feature vectors are the same for both decision methods.

7.2 Biometric System Errors

In this section, we derive expressions for the system errors, using likelihood ratio-based decisions, as a function of the probability density functions of the feature vectors.

7.2.1 False rejection rate

The *false rejection rate* (FRR) measures the probability that a feature vector is rejected as a member of some class, although it does originate from that class. For a specific class w_k and a given threshold t, $FRR_k(t)$ is given by:

$$FRR_k(t) = P(\mathbf{v} \in R_{k,t} | \mathbf{v} \in w_k) = \int_{R_{k,t}} p(\mathbf{v}|w_k) d\mathbf{v} \tag{7.7}$$

Since $A_{k,t} + R_{k,t} = \mathbf{V}$, this can also be written as:

$$FRR_k(t) = 1 - \int_{A_{k,t}} p(\mathbf{v}|w_k) d\mathbf{v} \tag{7.8}$$

The (average) overall false rejection rate $FRR(t)$ is found by integrating over all classes:

$$FRR(t) = \int_W FRR_k(t) \cdot p(w_k) dw_k \tag{7.9}$$

where W is the space of all classes. The summation over all (discrete) classes is represented by a (continuous) integral to indicate the infinite number of classes.

7.2.2 False acceptance rate

The *false acceptance rate* (FAR) measures the probability that a feature vector is accepted as a member of some class, although it does not originate from that class. For a specific class w_k and a given a threshold t, $FAR_k(t)$ is given by:

$$FAR_k(t) = P(\mathbf{v} \in A_{k,t} \mid \mathbf{v} \in w_i, i \neq k) = \int_{A_{k,t}} p(\mathbf{v}) d\mathbf{v} \tag{7.10}$$

where again $p(\mathbf{v}|\overline{w_k}) = p(\mathbf{v})$ is used. Then, the (average) global false acceptance rate $FAR(t)$ is found by integrating over all classes.

$$FAR(t) = \int_W FAR_k(t) \cdot p(w_k) dw_k \tag{7.11}$$

7.2.3 Receiver operating curve

The dependence of both error rates on the threshold can be visualized in a plot of FRR against FAR for varying threshold values, which is called the *receiver operating curve* (ROC). In this section, we derive an expression that describes the trade off between FAR and FRR for a likelihood ratio-based verification system. This derivation proceeds similar to the work in [Gol97] for posterior probabilities.

Let $\mathbf{A}_{k,t}$ be the acceptance region of class w_k for threshold t and $\mathbf{A}_{k,t-\Delta t}$ the acceptance region of the same class for threshold $t - \Delta t$. If the threshold decreases, the acceptance region increases. Let $\Delta\mathbf{A}_{k,t}$ be the difference region

$$\Delta\mathbf{A}_{k,t} = \mathbf{A}_{k,t-\Delta t} - \mathbf{A}_{k,t} \tag{7.12}$$

for each $\mathbf{v}$ in the difference region we can write:

$$\forall \mathbf{v} \in \Delta\mathbf{A}_{k,t} : t - \Delta t \leq L(\mathbf{v}) \leq t \tag{7.13}$$

By substituting Expression 7.4, this can be written as:

$$\forall \mathbf{v} \in \Delta\mathbf{A}_{k,t} : (t - \Delta t) \cdot p(\mathbf{v}) \leq p(\mathbf{v}|w_k) \leq t \cdot p(\mathbf{v}) \tag{7.14}$$

By integrating over the difference region:

$$(t - \Delta t) \cdot \int_{\Delta\mathbf{A}_{k,t}} p(\mathbf{v}) d\mathbf{v} \leq \int_{\Delta\mathbf{A}_{k,t}} p(\mathbf{v}|w_k) d\mathbf{v} \leq t \cdot \int_{\Delta\mathbf{A}_{k,t}} p(\mathbf{v}) d\mathbf{v} \tag{7.15}$$

and introducing the differences in false acceptance rate and false rejection rate when the threshold decreases from t to $t - \Delta t$:

$$\Delta FRR_k(t) = -\int_{\Delta\mathbf{A}_{k,t}} p(\mathbf{v}|w_k) d\mathbf{v} \tag{7.16}$$

$$\Delta FAR_k(t) = \int_{\Delta\mathbf{A}_{k,t}} p(\mathbf{v}) d\mathbf{v} \tag{7.17}$$

we obtain:

$$(t - \Delta t) \cdot \Delta FAR_k(t) \leq -\Delta FRR_k(t) \leq t \cdot \Delta FAR_k(t) \tag{7.18}$$

By letting $\Delta t \rightarrow 0$, the inequalities become equalities, and $\Delta FRR_k(t)$ and $\Delta FAR_k(t)$ go to zero as well. Then, we can write:

$$\frac{dFRR_k(t)}{dFAR_k(t)} = -t \tag{7.19}$$

which is a general result that describes the tradeoff between FAR and FRR for a likelihood ratio based decision system. The same expression has been derived in [Tre68] using other methods.

7.3 Optimality of Likelihood Ratios

In this section, we prove that, for verification based on fixed-length feature vectors, the use of the likelihood ratio is optimal in terms of average overall error rates. In this context, optimality is defined as the lowest FAR for a given FRR, or alternatively the lowest FRR for a given FAR. First, we consider the less complex case of single-user verification, where the system has to decide whether or not an input feature vector originates from the only user that is known to the system.

7.3.1 Single-User Verification

For single-user verification, there is one fixed distribution $p(\mathbf{v}|w_k)$ of feature vectors from the genuine user and one fixed distribution $p(\mathbf{v})$ of feature vectors from impostors. This situation is equivalent to the detection problem, which implies that the likelihood ratio is optimal for single-user verification [Tre68].

The relation between the likelihood ratio and the posterior probability, which is derived from Expressions 7.1 and 7.4, is given by:

$$p(w_k|\mathbf{v}) = p(w_k) \cdot L(\mathbf{v}) \tag{7.20}$$

Since there is only one user in the system, $p(w_k)$ is a constant, and both methods provide the same error rates if the thresholds t_p for posterior probability density and t_L for likelihood ratio are set to $t_p = p(w_k) \cdot t_L$. This means that for single-user verification, both methods provide the same ROC, with a different threshold parameterization along the curve. This is shown in Figure 7.1, for instance by observing the leftmost curve, with the associated threshold values for both methods.

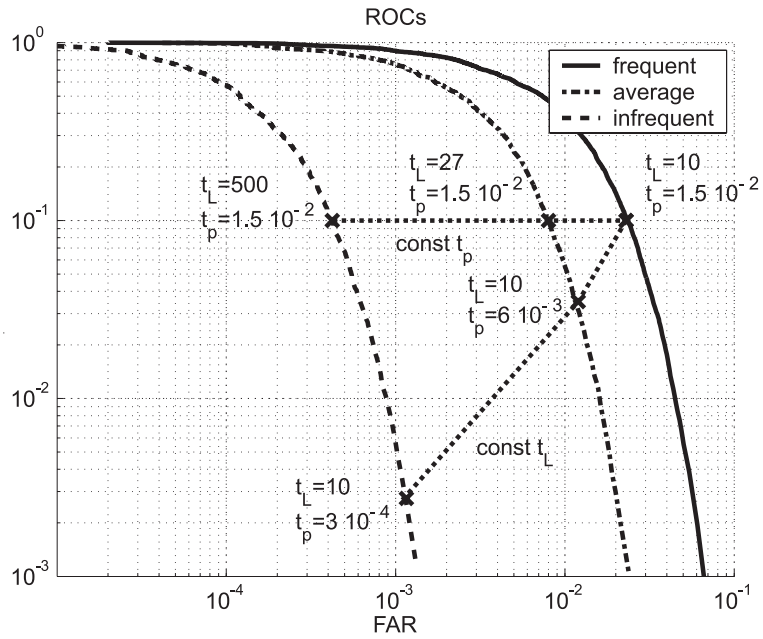


Figure 7.1: Averaging ROCs of classes with different occurrence probabilities.

7.3.2 Multi-User Verification

In a multi-user situation, the two methods lead to different average error rates. The difference between averaging with the two methods is illustrated in Figure 7.1, which shows three ROCs for frequently, average and infrequently occurring classes in a synthetic data set. At some specific positions at the ROCs, the threshold values for both methods are indicated. It can be seen that combining individual ROCs with a constant posterior probability threshold t_p will take the average of different points on the individual ROCs than combining individual ROCs with a constant likelihood ratio threshold t_L. The question that is answered in the rest of this section is which of all possible averaging paths will lead to the minimum overall error rates.

It can be explained intuitively that the use of likelihood ratios will lead to better overall matching performance than the use of posterior probabilities. A fixed posterior probability threshold requires equal similarity values for feature vectors from each individual, under the condition that the distribution of the feature vectors from the entire population is much wider than the distribution within one class. This means that a feature vector is relatively close to its associated class center, such that $p(\mathbf{v}) \approx p(w_k)$ (see Expression 7.1). This condition is easily satisfied in practice. The effect is an equal FRR and a lower FAR for less frequently occurring classes.

On the other hand, a fixed likelihood ratio threshold requires lower similarity values for less frequently occurring feature vectors (see Expression 7.20). As a consequence, the acceptance region for less frequently occurring classes is larger, which has two effects. The reduction in FAR is smaller, but at the same time, FRR is reduced. The overall recognition

performance can be optimized by choosing the right trade-off between these two effects.

Next, we prove that using likelihood ratios in multi-user verification is optimal. First, define $\varphi(L|w_k)$ as the probability density function of the likelihood ratio of an observation vector $\mathbf{v}$ that is taken from the true class w_k. Also, define $\varphi(L|\overline{w_k})$ as the probability density function of the likelihood ratio of an observation vector $\mathbf{v}$ that is not taken from the true class w_k. For these probability density functions, the following well-known relation holds [Tre68]:

$$\varphi(L|w_k) = L \cdot \varphi(L|\overline{w_k}) \tag{7.21}$$

The error rates FRR and FAR for class w_k, as a function of the threshold t, are given by

$$FRR_k(t) = \int_0^t \varphi(L|w_k)dL \tag{7.22}$$

and

$$FAR_k(t) = \int_t^\infty \varphi(L|\overline{w_k})dL \tag{7.23}$$

Next, we find expressions for the average $FRR(t(w_k))$ and $FAR(t(w_k))$ with a class-dependent threshold $t(w_k)$ by integrating over all classes:

$$FRR(t(w_k)) = \int_W p(w_k) \int_0^{t(w_k)} \varphi(L|w_k)dL\ dw_k \tag{7.24}$$

$$FAR(t(w_k)) = \int_W p(w_k) \int_{t(w_k)}^\infty \varphi(L|\overline{w_k})dL\ dw_k \tag{7.25}$$

For optimal verification performance, the question is how to choose the threshold t as a function of w_k, such that the resulting ROC is minimal. This is solved by Lagrange optimization, see for instance [Moo00]. The objective is to minimize FRR, subject to the condition of a constant FAR. The threshold is chosen as $t = t_{\text{opt}}(w_k) + \varepsilon f(w_k)$, where $t_{\text{opt}}(w_k)$ is the optimal threshold, $f(w_k)$ is some function of w_k, ε is a small constant, and some specific value for FAR is chosen as additional condition. Then:

$$\begin{aligned} J &= \int_W p(w_k) \int_0^{t_{\text{opt}}(w_k)+\varepsilon f(w_k)} \varphi(L|w_k)dL\ dw_k \\ &+ \lambda \left[\int_W p(w_k) \int_{t_{\text{opt}}(w_k)+\varepsilon f(w_k)}^\infty \varphi(L|\overline{w_k})dL\ dw_k - FAR(t) \right] \end{aligned} \tag{7.26}$$

has to be minimized by setting the derivative with respect to ε to zero:

$$\begin{aligned} \int_W p(w_k)\varphi(t_{\text{opt}}(w_k) + \varepsilon f(w_k)|w_k)\, f(w_k) dw_k \quad & - \\ \lambda \int_W p(w_k)\varphi(t_{\text{opt}}(w_k) + \varepsilon f(w_k)|\overline{w_k})\, f(w_k) dw_k \quad & = \quad 0 \end{aligned} \tag{7.27}$$

By realizing that this expression must hold for any $f(w_k)$, the integrals over all w_k can be omitted. Furthermore, since t_{opt} is optimal, ε is equal to zero, which further simplifies the expression to:

$$\varphi(t_{\text{opt}}(w_k)|w_k) - \lambda\varphi(t_{\text{opt}}(w_k)|\overline{w_k}) = 0 \tag{7.28}$$

Applying Expression 7.21 results in:

$$t_{\text{opt}}(w_k)\varphi(t_{\text{opt}}(w_k)|\overline{w_k}) - \lambda\varphi(t_{\text{opt}}(w_k)|\overline{w_k}) = 0 \tag{7.29}$$

which, by dividing both sides by $\varphi(t_{\text{opt}}(w_k)|\overline{w_k})$ and rearranging the expression, gives:

$$t_{\text{opt}}(w_k) = \lambda \tag{7.30}$$

Since λ is a constant, the optimal threshold $t_{\text{opt}}(w_k)$ is constant too, independent of w_k. Therefore, using a constant likelihood ratio threshold when averaging over the classes gives the optimal verification results.

7.4 Experimental Results

In this section, results of fingerprint matching experiments are presented. The proposed similarity measures, being Euclidean distance, posterior probabilities and likelihood ratios, have been evaluated by applying them to Database 2 of FVC2000 [Mai02]. The FVC2000 database consists of 880 8-bit gray-scale fingerprints, 8 prints of each of 110 different fingers. The images are captured with a capacitive sensor at 500 dpi, resulting in image sizes of 364 by 256 pixels.

7.4.1 Feature Vectors

We use two types of feature vectors that are extracted from the gray scale fingerprint images. The first feature vector is the squared directional field that is defined in Chapter 2, which is calculated at a regular grid of 11 by 11 points with spacings of 8 pixels and is centered at the core point (see Chapter 3). At each position in the grid, the squared directional field is coded in a vector of two elements. The resulting feature vector of length 242 is reduced to dimension 100 by principal component analysis over the entire population.

For approximately 10% of the fingerprints, the automatic core point extraction failed and for those fingerprints, the location of the core point was adjusted manually. The automatic core point extraction errors could be resolved by two related methods. First, feature vectors could be extracted at many regularly spaced locations from the fingerprint. That one feature vector that results in the highest matching score is used. This solution is inspired by the feature space correlation method that is described in [Ros02b]. Second, feature vectors could be extracted at each location where a (possibly false) core is detected. Again the best matching feature vector is used. This would save a lot of processing time compared to the first method.

The second feature vector is the Gabor response of the fingerprint, which is discussed in Appendix D. After substraction of the local mean, the fingerprint image is filtered by a set of four complex Gabor filters, which are given by:

$$h_{\text{Gabor}}(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \exp\left(j2\pi f(x \sin\theta + y \cos\theta)\right) \tag{7.31}$$

The orientations θ are set to 0, $\pi/4$, $\pi/2$, and $3\pi/4$, the spatial frequency is set to $f = 0.125$, which corresponds to a ridge-valley period of 8 pixels, and the width of the filter is set to $\sigma = 3$. The absolute values of the output images are taken, which are subsequently filtered by a Gaussian window with $\sigma = 6$. Next, samples are taken at a regular grid of 11 by 11 points with spacings of 8 pixels and centered at the core point. The resulting feature vector of length 484 is reduced to dimension 200 by principal component analysis of the entire population. This feature vector is inspired by FingerCode [Jai00b], but it can be calculated more efficiently since a rectangular grid is used rather than a circular one (see also [Jai01, Ros02a]), and it performs slightly better.

7.4.2 Matching Experiments

To enable calculation of the posterior probability density and likelihood ratio, we assume Gaussian probability density functions with unequal means but equal covariance matrices for the feature vectors from all individual classes. This covariance matrix represents the differences between multiple prints of the same finger, like noise, partial impressions, and elastic deformations. Another Gaussian probability density function is assumed for the feature vectors of the entire population, representing the differences between individual fingerprints. For both feature vectors, the inter class and intra class covariance matrices have been determined from the fingerprints in our database. Then the matching scores of 3080 genuine attempts and 5995 impostor attempts have been recorded.

The use of equal intra class covariance matrices for all users is motivated by the fact that in a biometric system in practice, only one feature vector is available as template. Therefore, no user dependent covariance matrix can be determined, and the best approximation possible is to use the average covariance matrix for all users.

There are a few motivations for using Gaussian distributions for the feature vectors. In general, measurements of natural features tend to a Gaussian distribution. Furthermore, as the dimension of the feature vectors is reduced by principal component analysis, the fea-

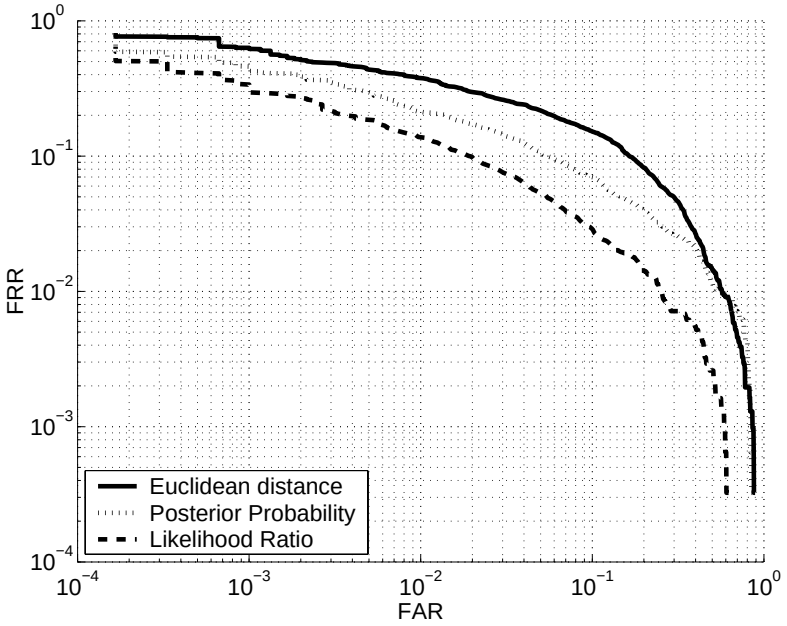


Figure 7.2: Results of the directional field based matching algorithm.

ture vector elements are weighted sums of measured features, which approximate a Gaussian distribution even better, as dictated by the central limit theorem. Appendix F provides expressions for the verification errors for Gaussian distributed feature vectors.

The resulting average ROCs of the matching experiment, using Euclidean distance, posterior probability and likelihood ratios, are shown in Figures 7.2 and 7.3. Both the directional field and the Gabor response perform at an equal error rate (EER) of approximately 5% when using likelihood ratios. The figures show that the use of posterior probabilities results in a higher EER of 8%, while the Euclidean distance performs at an even worse EER of 15%. Figure 7.3 shows that posterior probabilities perform better than likelihood ratios for FAR $>$ 20%. For that setting, the acceptance region is very large, and the criterion $p(\mathbf{v}) \approx p(w_k)$ is no longer satisfied. Combined with possibly unequal intra class covariance matrices, this might explain the results.

Next, a new feature vector has been constructed by concatenating the directional field and Gabor response into one large feature vector of dimension 300, and new inter class and intra class covariance matrices have been determined. For this combined feature vector, the performance differences are even more significant. Likelihood ratios perform at EER = 0.5%, posterior probability at EER = 7% and Euclidean distance at EER = 12%, as shown in Figure 7.4. The EER of 0.5% is a remarkably good result, equal to the best participant to FVC2000.

The algorithm takes less than 1 ms, compared to approximately 100 ms for a fast minutiae matching algorithm (see Chapter 6). Using appropriate preprocessing, consecutive matches of a single test fingerprint to a database of templates take less than 25 μs per template, which enables the search through a database that contains 40,000 entries in 1 second.

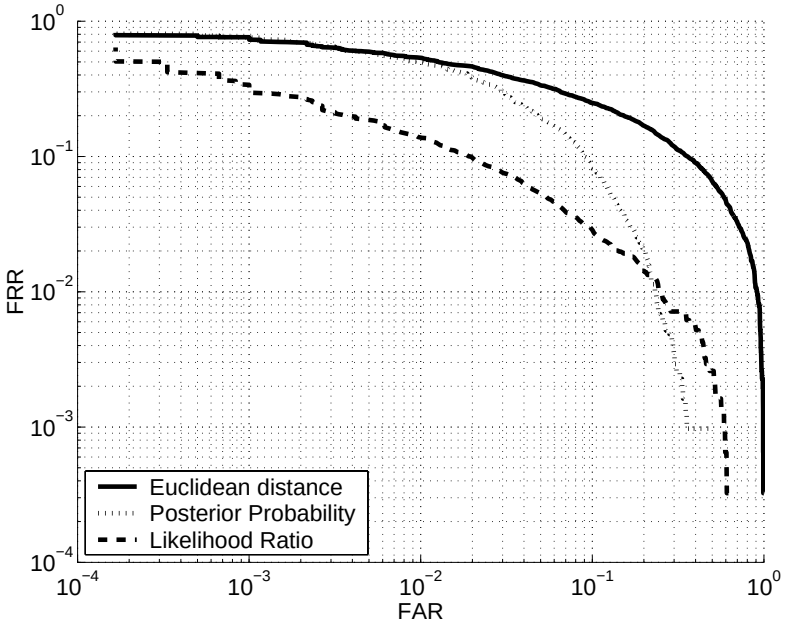


Figure 7.3: Results of the Gabor response based matching algorithm.

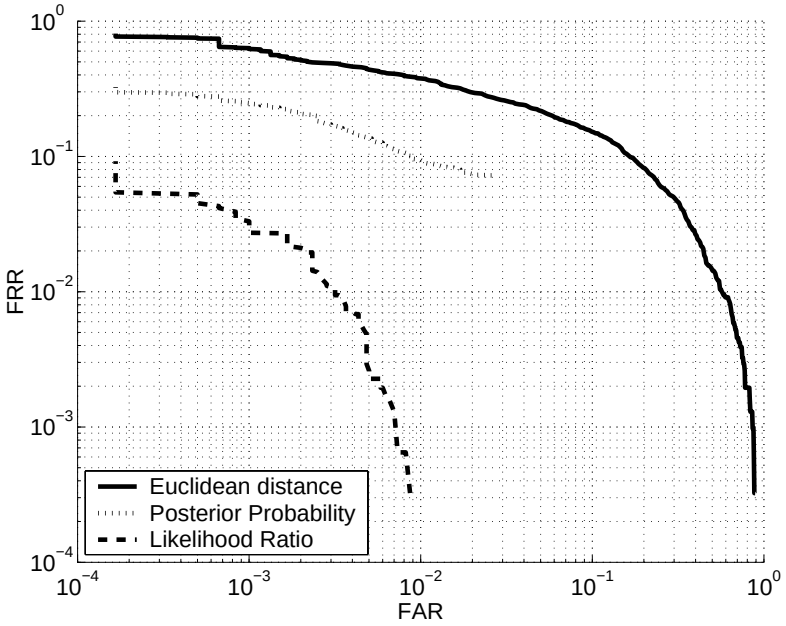


Figure 7.4: Results of the matching algorithm that uses both the directional field and the Gabor response as features.

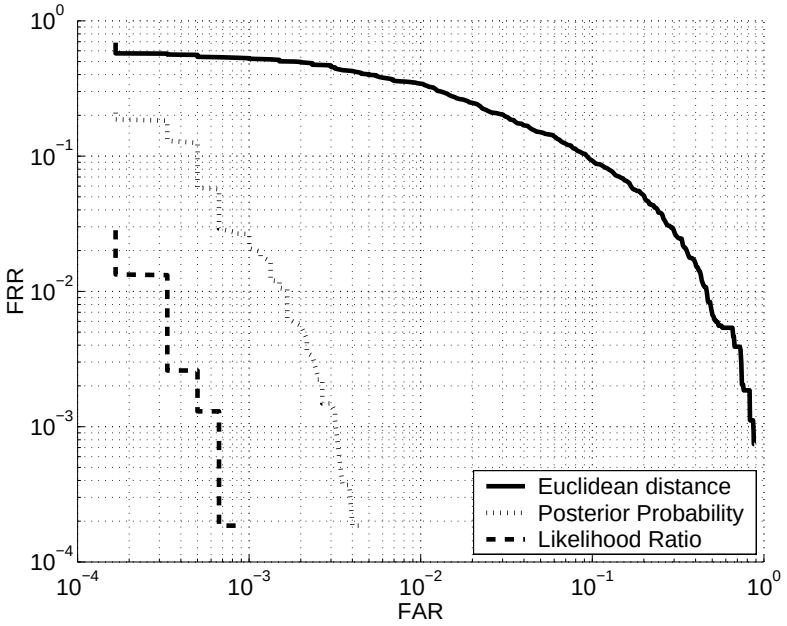


Figure 7.5: Results of the matching algorithm that is based on both features and uses two fingerprints for template construction.

Table 7.1: Matching performance using multiple fingerprints for template construction.

Number of templates	1	2	3	4	5
EER	0.5%	0.07%	0.03%	0.01%	0%

7.4.3 Unknown Versus Known Class Centers

For a biometric system in practice, the feature vectors that correspond to the true classes centers are not known. Instead, only one example of the feature vector is available as template. This means that all elements of the intra class covariance matrix are twice as large, compared to the situation with known class centers. As a result, the acceptance region has to be $2^{d/2}$ times as large (with d the dimension of the feature vector) for a specific FRR. If the inter class feature vector distribution is much wider than the intra class distribution, the impostor distribution can be assumed constant for thresholds that correspond to a relatively high likelihood ratio. Consequently, FAR is also $2^{d/2}$ times as large as it would be with known class centers. This is confirmed in experiments with low dimensional synthetic data sets.

The combined feature vector has dimension $d = 300$. This means that, at least theoretically, FAR can be reduced by a factor of $2^{300/2} \approx 10^{45}$ when using class centers as templates. To validate this effect, experiments have been set up where multiple fingerprints have been used for construction of the template. The template is taken as the average of multiple individual feature vectors, while only one of the remaining fingerprint is used as test vector. For this experiment, 6160 genuine attempts and 5995 impostor attempts have been carried out. The ROCs for the three similarity measures, using 2 fingerprints to construct the template, are

shown in Figure 7.5, and the equal error rates for likelihood ratios with 1 to 5 fingerprints for template construction are shown in Table 7.1. The table shows that the performance gain in practice is not as large as it is in theory. But still, the matching performance can be increased enormously by using multiple fingerprints for template construction.

Two more remarks have to be made on this subject. First, the performance is evaluated in a database of 880 fingerprints, using approximately 6000 genuine and 6000 impostor attempts. In this evaluation set, error rates smaller than 0.1% cannot be estimated reliably. Therefore, the 0% in Table 7.1 does not mean that we have implemented the perfect biometric system, but only that it made no errors on our database. Second, the practical performance gain of using multiple feature vectors for template construction is smaller than the theoretic gain since the inter class covariance matrix is not much wider than the intra class covariance matrix for most of the elements of the feature vector. Therefore, the assumption of a constant $p(\mathbf{v})$ is not true in practice, and the performance gain is smaller than predicted.

7.5 Conclusions

In this chapter, we have shown that the verification of a single user is equivalent to the detection problem, which implies that, for single-user verification, the likelihood ratio is optimal. We have also shown that, in single-user verification, decisions based on posterior probability and likelihood ratio are equivalent, and result in the same ROC. However, in a multi-user situation, the two methods lead to different average error rates. As a third result, we have proven theoretically that, for multi-user verification, the use of the likelihood ratio is optimal in terms of average error rates.

The superiority of the likelihood based similarity measure is illustrated by experiments in fingerprint verification. It is shown that error rates of approximately 10^{-4} can be achieved when using multiple fingerprints for template construction. Since the algorithm is extremely fast, it can be used to search through large fingerprint databases. For automatic application of the algorithm, improvements have to be made in the automatic extraction of the core point. This could be circumvented by trying all detected cores, but that would slow down a database search.

Chapter 8

Correlation-Based Fingerprint Matching

Abstract

In this chapter, a correlation-based fingerprint verification system is presented. Unlike the traditional minutiae-based systems, this system directly uses the richer gray-scale information of the fingerprints. The correlation-based fingerprint verification system first selects appropriate templates in the primary fingerprint, uses template matching to locate them in the secondary print, and compares the template positions of both fingerprints.

The correlation-based fingerprint verification system is capable of dealing with low-quality images from which no minutiae can be extracted reliably and with fingerprints that suffer from non-uniform shape distortions. This system has participated in the Fingerprint Verification Competition 2000 [Mai02] where it obtained an average rating. This chapter has been published in [Baz00b].

8.1 Introduction

Most fingerprint verification systems follow a minutiae-based approach. Apart from the presence of elastic deformations, the main drawback of the minutiae-based approach is the error propagation from the minutiae extraction to the decision stage. In general, the extracted minutiae templates contain a number of defects such as false, missed and displaced minutia, especially in low-quality fingerprints.

In order to deal with some of the problems of the minutiae-based approach, we have chosen an alternative approach. Instead of only using the minutiae locations, our method directly uses the (locally normalized) gray-level information from the fingerprint image, since a gray-level fingerprint image contains much richer, more discriminatory, information than only the minutiae locations. Those locations only characterize a small part of the local ridge-valley structures [Pra00, Jai00b, Mai99].

This chapter is organized as follows. First, Section 8.2 introduces the correlation-based fingerprint matching system. This section also provides an analysis of the theoretical error rates of the system. Then, Section 8.3 presents some experimental results and compares these to the theoretical errors.

8.2 Correlation-Based Fingerprint Matching

The correlation-based fingerprint verification system has been inspired by [Rod99]. It first selects characteristic templates (small segments) in the *primary* fingerprint. Then, template matching is used to find the positions in the *secondary* fingerprint at which the templates match best. Finally, the template positions in both fingerprints are compared in order to make the decision whether the prints match. In this chapter, the template fingerprint will be called the primary fingerprint an the test fingerprint will be called the secondary fingerprint in order to avoid confusion with the 'templates' that are used in the correlation.

8.2.1 Template Selection

The first step in the template matching algorithm is the selection of appropriate templates. This is a crucial step, since good templates will be easily localized in the secondary print at the right position, while low-quality templates will not. More generally, the templates should be uniquely localized in the secondary fingerprint. The template should fit as well as possible at the correct location, but as badly as possible at other locations.

The first template property to consider is the size of the templates. There must be an optimal template size, as can be seen from two extreme situations. When the entire fingerprint is taken as template, any attempt to align specific corresponding positions will lead to misalignments at other positions due to shape distortions. On the other hand, if templates of only 1 by 1 pixel are chosen, it is clear that the templates do not offer enough distinction. Experiments have shown that a template size of 24 by 24 pixels is a good compromise.

The second problem in selecting the right templates is which template positions to chose. Research has shown for instance, that a template that contains only parallel ridge-valley structures cannot be located very accurately in the secondary fingerprint. In this paper, three template selection criteria are proposed, being *minutiae-based*, *coherence-based* and *correlation-based*.

Minutiae-Based Template Selection

As mentioned before, templates that only contain parallel ridge-valley structures do not offer much distinction. On the other hand, when a template contains one or more minutiae, it will be much easier to find the correct location in the secondary print. Using this assumption, one possible approach to select template locations is to extract minutiae from the fingerprint image and to define templates around the minutiae locations.

A drawback of this technique is that it suffers from most of the problems of minutiae-based systems. Still, many false minutiae are extracted, causing at least a part of the templates to be rather unreliable.

Coherence-Based Template Selection

The coherence of an image area is a measure that indicates how well the local gradients are pointing in the same direction. In areas where the ridge-valley structures are only parallel lines, the coherence is very high, while in noisy areas, the coherence is low as shown in Chapter 2.

Templates that are chosen in regions of high coherence values cannot be located reliably in a second fingerprint [Sch00]. However, at locations around minutiae, more different gray-scale gradient orientations are present, resulting in a significantly lower coherence. Therefore, the coherence can be used as an appropriate measure that indicates the presence of minutiae as well as a measure that indicates how well a template can be located in the secondary fingerprint.

At first sight, this template selection criterion seems to conflict with segmentation, discussed in Chapter 4. While segmentation chooses the regions of low coherence values as noise or background areas, now the regions that have low coherence values have to be chosen as reliable templates. However, this contradiction is solved by the notion of scale [Lin94]. Segmentation selects a large, closed area as foreground, in which holes and other irregularities are filled. Instead, the coherence based template selection only searches for local coherence dips in this foreground area.

The drawback of this method is that noisy areas show coherence dips as well, while these are certainly not reliable templates. This problem may be solved by using appropriate filters.

Correlation-Based Template Selection

The third method satisfies the template requirements most directly. In this method, templates are selected by checking how well they fit at other locations in the same fingerprint. If a template fits almost as well at another location as it does at its original location, it is not a useful template. However, if a template fits much worse at all other locations in the fingerprint, it is a template that offers a lot of distinction. Therefore, the ratio of fit at a template's original location to the fit at the next best location can be used as a template selection criterion.

Since the correlation-based checking is carried out by means of template matching, this method consumes a lot of computational power. This makes it a less attractive method to use. However, it is for instance possible to combine this approach with the previous two methods. In that case, candidate template locations are extracted by one of the methods of the previous subsections. Then, the correlation characteristics of those locations are checked as an additional selection criterion.

8.2.2 Template Matching

Once the templates have been selected in the primary fingerprint, their corresponding positions in the secondary fingerprint have to be found. This can be done using standard template matching techniques.

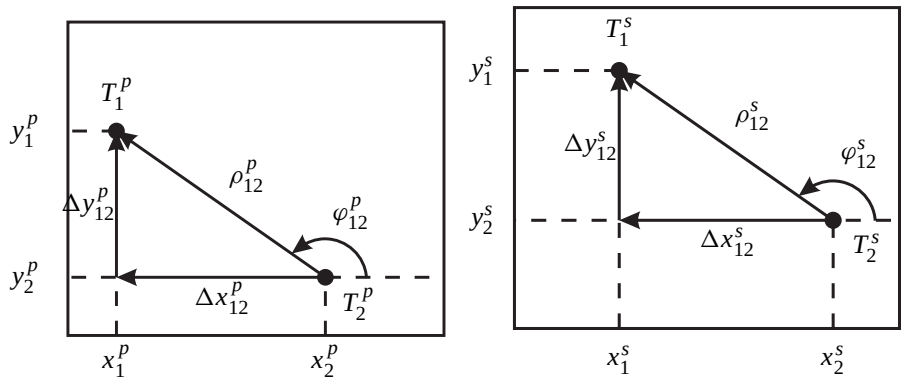
The template is shifted pixelwise over the secondary print. At each position, the gray-level distance between the template and the corresponding area in the secondary print is determined by summing the squared gray-level differences for each pixel in the template. After having shifted the template over the entire finger, the location where the distance is minimal is chosen as the corresponding position of the template in the second fingerprint.

This is a very computationally demanding technique. However, there are possibilities to speed up the process. When both the template and the fingerprint are normalized (we have chosen: $E[I_{x,y}] = 0$ and $Var[I_{x,y}] = 1$, where $I_{x,y}$ is the gray-scale image at pixel (x, y)), a convolution, or filter, can be used instead. For this method, it is required that these conditions do not only hold globally for the whole image, but also locally for each area in the image. If the size of the fingerprint is chosen appropriately, a 2-dimensional FFT can be used for even more efficiency [Ver00].

As result of the template matching, for each template position in the primary fingerprint, the corresponding, or best matching, template position in the secondary print is obtained.

8.2.3 Classification of Template Positions

The fingerprint matching algorithm, based on two sets of template positions, uses two decision stages. First, elementary decisions are made by classifying the individual template position pairs to be matching or not. Then, the information of all template pairs is merged in order to make a final decision whether the primary and secondary fingerprint match or not.



(a) Primary fingerprint (b) Secondary fingerprint

Figure 8.1: Relative template positions, tolerating translations.

Elementary Decisions

After template matching, there are two sets of corresponding template locations $(\mathbf{x}^p, \mathbf{y}^p)$ and $(\mathbf{x}^s, \mathbf{y}^s)$, where $\mathbf{x} = [x_1, \ldots, x_n]^T$ and $\mathbf{y} = [y_1, \ldots, y_n]^T$ are the coordinates of the templates and the superscripts p and s refer to the primary and secondary fingerprints. Now, for all n template pairs, a decision has to be made whether the positions correspond to each other:

$$(x_i^s, y_i^s) \stackrel{?}{\approx} (x_i^p, y_i^p) \qquad \text{for } 1 \leq i \leq n \tag{8.1}$$

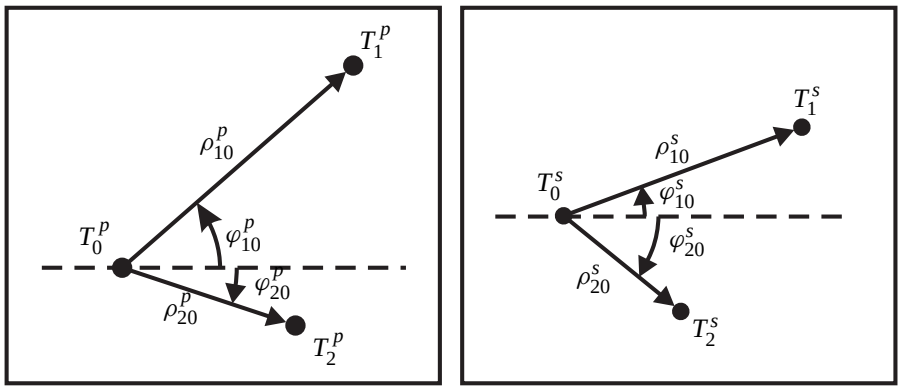
Directly examining the difference of both template coordinate pairs would only allow some fixed translation in x and y directions. Template pairs that are some more translated with respect to each other would be classified as non-matching. In order to deal with the translations, relative template positions (RTPs) are used instead. Now, the test becomes:

$$\Big[(x_i^s, y_i^s) - (x_j^s, y_j^s)\Big] \stackrel{?}{\approx} \Big[(x_i^p, y_i^p) - (x_j^p, y_j^p)\Big] \qquad \text{for } 1 \leq i, j \leq n,\ i \neq j \tag{8.2}$$

or

$$(\Delta x_{ij}^s, \Delta y_{ij}^s) \stackrel{?}{\approx} (\Delta x_{ij}^p, \Delta y_{ij}^p) \qquad \text{for } 1 \leq i, j \leq n,\ i \neq j \tag{8.3}$$

using the notations that are illustrated in Figure 8.1. Instead of comparing n template positions, now $n(n-1)/2$ RTPs are classified, of which $n-1$ are independent.



(a) Primary fingerprint (b) Secondary fingerprint

Figure 8.2: Use of 3 templates to allow rotation and scaling.

Direct application of this test allows for some fixed displacement of the templates in the x and y directions, which is set by a threshold (x_T, y_T). However, to allow rotation and scaling as well, the RTPs are converted to polar coordinates, as illustrated in Figure 8.1:

$$\rho = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{8.4}$$

$$\varphi = \angle(\Delta x, \Delta y) \tag{8.5}$$

where $\angle(x, y)$ is defined as:

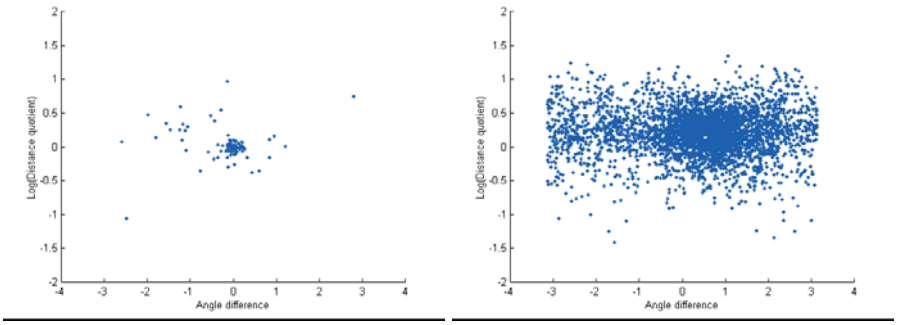
$$\angle(x, y) = \begin{cases} \tan^{-1}(y/x) & x \geq 0 \\ \tan^{-1}(y/x) + \pi \quad \text{for} & x < 0 \wedge y \geq 0 \\ \tan^{-1}(y/x) - \pi & x < 0 \wedge y < 0 \end{cases} \tag{8.6}$$

This leads to the classification test:

$$(\rho_{ij}^s, \varphi_{ij}^s) \stackrel{?}{\approx} (\rho_{ij}^p, \varphi_{ij}^p) \qquad \text{for } 1 \leq i, j \leq n,\ i \neq j \tag{8.7}$$

which tolerates some fixed amount of rotation and scaling, especially when ρ_s and ρ_p are compared by division instead of subtraction. This tolerates more displacement for templates that are further away from each other, which is a much more natural restriction than fixed x and y displacements. This kind of tolerance is capable of handling some amount of non-uniform shape-distortion, caused by the fingerprint elasticity. Again, the degree of tolerance can be set by thresholds ρ_T and φ_T.

The next step is to allow any rotation and scaling instead of only some fixed amount. As



(a) Matching fingerprints (b) Non-matching fingerprints

Figure 8.3: Scatter diagrams of the $(\ln \rho_s/\rho_p,\ \phi_s - \phi_p)$ pairs.

illustrated in Figure 8.2, a third template is used to obtain the test that is not only fixed for translation of the template positions, but for rotation and scaling as well:

$$(\varphi^s_{ji} - \varphi^s_{ki},\ \rho^s_{ji}/\rho^s_{ki}) \stackrel{?}{\approx} (\varphi^p_{ji} - \varphi^p_{ki},\ \rho^p_{ji}/\rho^p_{ki}) \quad \text{for } 1 \leq i, j, k \leq n,\ i \neq j \neq k \tag{8.8}$$

However, there is one practical drawback to this method. The template locations are obtained by means of template matching. If a template is scaled or rotated more than some constant, it is not possible anymore to localize it in the secondary image. Furthermore, the same holds, to some extent, for scaling. This makes the tolerance of any amount of rotation and scaling less useful.

Since the test of Expression 8.7 uses one more independent classification than the test of Expression 8.8, and the last step does not add much value, we adopt the test of Expression 8.7 for the rest of the paper. The scatter diagrams of the $(\ln \rho^s_{ij}/\rho^p_{ij},\ \varphi^s_{ij} - \varphi^p_{ij})$ pairs, both for matching fingerprints and for non-matching prints, are given in Figure 8.3. These pairs are classified by applying an elliptical threshold:

$$\left(\frac{\ln \frac{\rho^s_{ij}}{\rho^p_{ij}}}{\rho_T}\right)^2 + \left(\frac{\varphi^s_{ij} - \varphi^p_{ij}}{\varphi_T}\right)^2 < 1 \tag{8.9}$$

where ρ_T and φ_T are the parameters determining the shape of the ellipse.

The result of this procedure is a match or non-match classification for all $n(n-1)/2$ template combinations. The probability that an RTP is classified non-matching while the prints match, is given by $p(\hat{\omega}_{0,T}|\omega_{1,F})$, while the probability that a template distance is

classified matching while the prints actually do not match is denoted by $p(\hat{\omega}_{1,T}|\omega_{0,F})$. Here, the subscripts T and F denote template and fingerprint respectively.

The thresholds that provide the best discrimination of matching and non-matching RTPs, give for this database:

$$p(\hat{\omega}_{0,T}|\omega_{1,F}) = 0.2 \tag{8.10}$$
$$p(\hat{\omega}_{1,T}|\omega_{0,F}) = 0.02 \tag{8.11}$$

Combining Elementary Decisions

We now have to classify the two fingerprints as being a match or not. This decision is based on $n(n-1)/2$ relative template positions. This can be solved using the theory of Bernoulli experiments, which combine n independent experiments using binominal distribution. The probability $P(X = k)$ for k positive outcomes when doing n independent experiments that all have a probability of success p is given by:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{8.12}$$

where

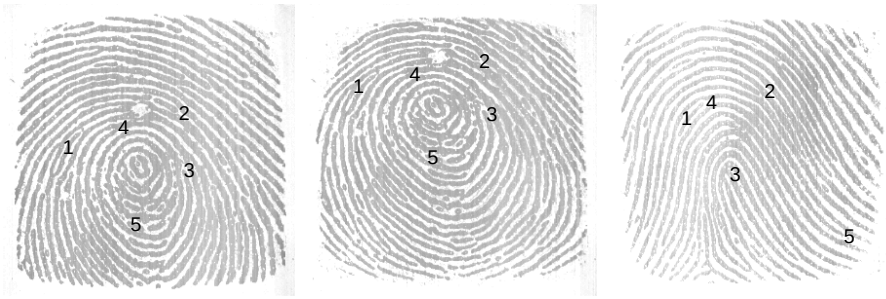
$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \tag{8.13}$$

is the binominal coefficient and $n!$ denotes factorial.

The $n(n-1)/2$ RTPs are certainly not independent. If we choose one template as reference, all template positions are fixed by only $n-1$ distances, while all other distances can be calculated. Therefore, only $n-1$ independent RTPs are available.

The choice of the reference template is rather important. If one template is not localized at the right position in the secondary fingerprint, and this template is chosen as reference template, all RTPs will be classified non-matching. This results in a false rejection of the fingerprint. Therefore, we chose the reference template as the one that has the most RTP matches.

Once the reference template has been chosen, there are $n-1$ RTPs that are in principle independent. This means that the combination of all RTP matches can be considered as a Bernouilli experiment, and Expression 8.12 can be applied. A threshold k_T is set for the final fingerprint match or non-match classification, resulting in:



(a) Primary fingerprint (b) Matching fingerprint (c) Non-matching fingerprint

Figure 8.4: Template positions in primary and secondary fingerprints.

$$\begin{aligned} FAR &= p(\hat{\omega}_{1,F}|\omega_{0,F}) \\ &= P(X \geq k_T \mid \text{non-match}) \\ &= \sum_{k=k_T}^{n-1} \binom{n-1}{k} p(\hat{\omega}_{1,T}|\omega_{0,F})^k \cdot \\ &\quad (1 - p(\hat{\omega}_{1,T}|\omega_{0,F}))^{n-k-1} \end{aligned} \tag{8.14}$$

and

$$\begin{aligned} FRR &= p(\hat{\omega}_{0,F}|\omega_{1,F}) \\ &= P(X < k_T \mid \text{match}) \\ &= \sum_{k=0}^{k_T-1} \binom{n-1}{k} p(\hat{\omega}_{0,T}|\omega_{1,F})^{n-k-1} \cdot \\ &\quad (1 - p(\hat{\omega}_{0,T}|\omega_{1,F}))^k \end{aligned} \tag{8.15}$$

In order to meet the commonly used requirements for fingerprint verification systems, being $FAR = 10^{-4}$ and $FRR = 10^{-2}$, and using the values given in Expressions 8.10 and 8.11, it can be calculated that the use of $n = 10$ templates with threshold $k_T = 4$ satisfies the required performance. For these parameters, the exact performance is given by:

$$FRR = p(\hat{\omega}_{0,F}|\omega_{1,F}) = 1.8 \cdot 10^{-5} \tag{8.16}$$

$$FAR = p(\hat{\omega}_{1,F}|\omega_{0,F}) = 3.1 \cdot 10^{-3} \tag{8.17}$$

An example of the results of this method, using only 5 templates, is given in Figure 8.4. The figure shows that for matching prints, indeed the relative template positions are about equal, while for non-matching prints, they are completely different.

8.2.4 Discussion of the Method

The correlation-based fingerprint verification that is proposed in this section, is compared to the traditional minutiae-based methods. The advantage of the correlation-based method are:

- The method uses the much richer gray-level information of the fingerprint image instead of only positions of minutiae.
- The method is also capable of dealing with fingerprints of low image quality from which no minutiae can be extracted reliably.
- False and missed minutiae do not decrease the matching performance.
- Unlike the minutiae templates, the template locations are already paired, which results in much simpler matching methods. During minutiae matching, it is not known in advance which minutiae from both sets should correspond.
- The first decision stage only classifies relative template positions. This method tolerates non-uniform local shape distortions in the fingerprint, unlike the minutiae templates for which the optimal global transform is searched.

The disadvantages of the correlation-based fingerprint verification method are:

- Template matching is a method that demands a rather high computational power, which makes the method less applicable for real time applications. On the other hand, more and more hardware solutions becomes available for this task.
- The method is at the moment not capable of dealing with rotations of more than about 10 degrees. This is caused by the fact that, for larger rotations, the templates do not match well anymore, causing incorrect positions to be found. A solution to this problem is rotating the templates and then performing the matching again. However, this is a solution that requires a lot of additional computational power.
- Problems might arise if the minutiae-based or coherence-based template selection methods are used while 2 minutiae surroundings can not be distinguished. In this case, there is a possibility that template matching will find the incorrect template position, which degrades the matching performance. Use of the correlation-characteristic template selection will solve this problem.

8.3 Experimental Results

The correlation based fingerprint verification system that is described in this paper has participated in FVC2000, a Fingerprint Verification Competition [Mai00] which was part of the

ICPR2000 conference. This section will present some of the results of this competition as well as a discussion how to interpret these results.

8.3.1 Experiment Setup

Although many research groups have developed fingerprint verification algorithms, only a few benchmarks are available. Developers usually perform tests over self-collected databases, since in practice, the only available sets are the NIST databases, containing thousands of scanned inked impressions of fingers. Since these images significantly differ from those acquired electronically, they are not well-suited for testing on-line fingerprint systems.

FVC2000 [Mai02] attempts to establish a first common benchmark, allowing companies and academic institutions to compare performance and track improvement in their fingerprint recognition algorithms unambiguously. However, the databases used in this contest have not been acquired in a real environment according to a formal protocol and the images originate from sensors that are not native to the participating systems. Therefore, FVC2000 is not an official performance certification of the participating systems, but it should be considered as a technology evaluation.

The system performance is evaluated on images from four different fingerprint databases. Three of these databases are acquired by various sensors, low-cost and high-quality, optical and capacitive. The fourth database is synthetically generated using the approach described in [Cap00a]. All databases contain 8 prints of 110 different fingers, so 880 fingerprints in total. All prints were captured by untrained volunteers, resulting in fingerprints ranging from high quality to very low quality.

Of all fingerprints, 8 prints of 10 fingers were distributed to the participants to tune their algorithms to the databases. The algorithms were tested using the other 100 fingers.

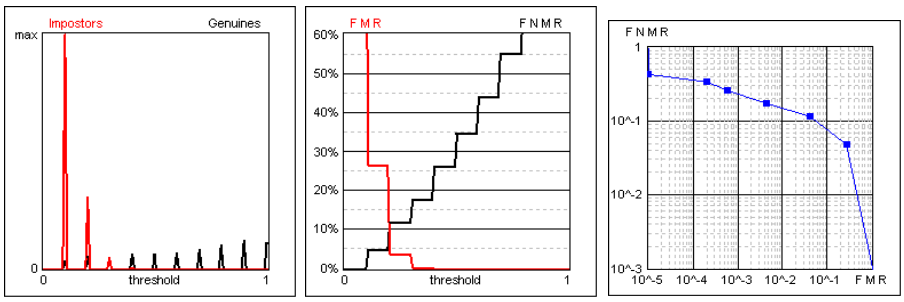
For each database, an FAR and an FRR test is performed. For the FAR test, the first print of each finger is matched against the first print of all other fingers, leading to 4,950 impostor attempts. For the FRR test, each print of each finger is matched against all other prints of the same finger, leading to 2,800 genuine attempts.

8.3.2 Experimental Results

In the Fingerprint Verification Competition 2000 this system obtained an average rating. Some charts, showing performance measures for the first database, are given in Figure 8.5. The discrete distribution is caused by the fact that the competition required a confidence measure between 0 and 1, indicating how well 2 fingerprints match. The confidence measure of our method is the quotient of the number of matching RTPs and the maximum number of matching RTPs that is possible, causing a distribution that consists of n discrete peaks.

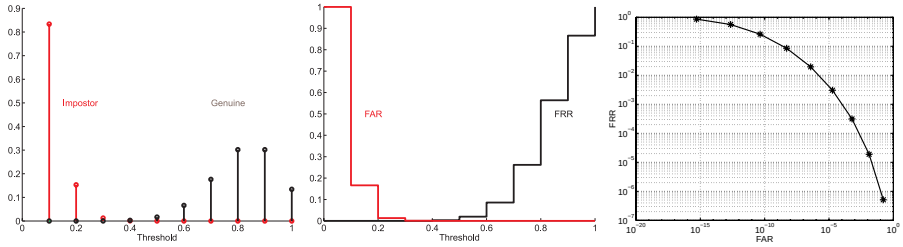
One interesting measure is the equal error rate (EER) of the system, which is the operating mode of the system for which FAR and FRR are equal. For this database, the EER was 7.98%.

The correlation based fingerprint verification system consumes relatively much CPU time.



(a) Distributions (b) FAR and FRR (c) ROC

Figure 8.5: Results of the correlation based fingerprint verification system. In this figure, FAR is called false matching rate (FMR) and FRR is called false non-matching rate (FNMR).



(a) Distributions (b) FAR and FRR (c) ROC

Figure 8.6: Theoretic results.

The enrollment is the most time-consuming part of the algorithm. The version that was sent to FVC2000 takes 10 seconds, which was the maximum time allowed. The enrollment time can be reduced by evaluating fewer candidate templates, but this will decrease the overall template quality and therefore also the system performance. The average matching time was 2 seconds.

8.3.3 Comparison to Theoretic Results

In order to compare the experimental results to the theory, Figure 8.6 shows the same performance measures, according to the theory of Expressions 8.10, 8.11, 8.14 and 8.15. Using these expressions, the theoretic EER is given by $4.6 \cdot 10^{-4}$.

From these performance measures, it is clear that the correlation-based fingerprint verification system does not perform as well as it was supposed to. When comparing Figure 8.5(a) to Figure 8.6(a), it can be seen that the experimental impostor distribution approximately equals the theoretic one, but that the genuine distribution is much wider than it is in theory.

The deviation of the genuine distribution can be explained by the fact that the RTPs are not independent for genuine matches, while this was assumed by using the binominal distribution of Expression 8.12. The most probable cause of the lower genuine scores is the rotation of fingerprints. If two fingerprints are rotated more than about 10 degrees with respect to each other, template matching cannot localize some of the templates correctly. This causes the dependent RTPs to be classified incorrectly.

This situation certainly exists for some pairs of fingerprints in the databases. The fingerprints are specified to have a rotation of -15 to +15 degrees from normal, which means that relative rotations up to 30 degrees might appear in these databases.

A possible solution to this problem is the determination of the relative rotation of two fingerprints, using for instance the singular points. Once the rotation is known, it can be compensated, after which the correlation based fingerprint verification algorithm can be applied directly.

8.4 Conclusions

The correlation-based fingerprint verification system that was presented in this chapter provides a very simple and direct solution to the fingerprint matching problem. Unlike the minutiae-based systems, this approach does not require much preprocessing. As a consequence, there will be no errors introduced in these steps.

The system uses the much richer gray-level information of a fingerprint image. It is capable of dealing with low image quality fingerprints and missed and spurious minutiae. Due to the paired templates, the decision stage is much simpler and it is able to deal with some non-uniform shape-distortion problems.

Experiments have shown that the correlation based fingerprint verification system performs approximately as well as other types of systems. The performance of the system can be enhanced by solving the problem of fingerprints that show more than some amount of rotation with respect to each other.

Chapter 9

An Intrinsic Coordinate System for Fingerprint Matching

Abstract

In this chapter, an intrinsic coordinate system is proposed for fingerprints. First the fingerprint is partitioned in regular regions, which are regions that contain no singular points. In each regular region, the intrinsic coordinate system is defined by the directional field. When using the intrinsic coordinates instead of pixel coordinates, minutiae are defined with respect to their position in the directional field. The resulting intrinsic minutiae coordinates can be used in a elastic distortion-invariant fingerprint matching algorithm. Elastic distortions, caused by pressing the 3-dimensional elastic fingerprint surface on a flat sensor, now deform the entire coordinate system, leaving the intrinsic minutiae coordinates unchanged. Therefore, matching algorithms with tighter tolerance margins can be applied to obtain better performance. This chapter has been published in [Baz01c] and [Baz01a].

9.1 Introduction

The first step in a fingerprint recognition system is to capture the print of a finger by a fingerprint sensor. In this capturing process, the 3-dimensional elastic surface of a finger is pressed on a flat sensor surface. This 3D-to-2D mapping of the finger skin introduces distortions, especially when forces are applied that are not orthogonal to the sensor surface. The effect is that the sets of minutiae of two prints of the same finger no longer fit exactly. The ideal way to deal with distortions would be to invert the 3D-to-2D mapping and compare the minutiae positions in 3D. Unfortunately, in the absence of a calibrated measurement of the deformation process, there is no unique way of inverting this mapping.

Instead of modeling the distortions, most minutiae matching techniques use local similarity measures [Jia00], or allow some amount of displacement in the minutiae matching stage [Jai97b]. However, decreasing the required amount of similarity not only tolerates small elastic distortions, but increases the false acceptance rate (FAR) as well. It is therefore reasonable to consider methods that explicitly attempt to model and eliminate the distortion. Such methods can be expected to be stricter in the allowed displacement during minutiae matching. As a consequence, the FAR can be decreased without increasing the false rejection rate (FRR).

As far as we know, the only paper that explicitly addresses elastic distortions is [Cap01]. In that paper, the physical cause of the distortions is modeled by distinguishing three distinct concentric regions in a fingerprint. In the center region, no distortions are present, since this region tightly fits to the sensor. The outer, or external, region is not distorted either, since it does not touch the sensor. The outer region may be displaced and rotated with respect to the inner region, due to the application of forces while pressing the finger at the sensor. The region in between is distorted in order to fit both regions to each other.

Experiments have shown that this model provides an accurate description of the elastic distortions in some cases. The technique has successfully been applied to the generation of many synthetic fingerprints of the same finger [Cap00a]. However, the model has not yet been used in an algorithm for matching fingerprints. Accurate estimation of the distortion parameters is still a topic of research. Furthermore, the prints cannot be truly normalized by the model, since it only describes relative distortions.

In this chapter, an alternative approach is proposed, using a linked multilevel description of the fingerprint. The coarse level of this description is given by the directional field. The DF describes the orientation of the local ridge-valley structures, thus modeling the basic shape of the fingerprint. We use the high-resolution DF estimate (presented in Chapter 2) that is based on the averaging of squared gradients. The features at the detailed level of the fingerprint are given by the minutiae. Instead of the common practice of treating the DF and the minutiae as two separate descriptions, of which one is used for classification and the other for matching, we propose a link between these two levels, by defining the *intrinsic coordinate system* of a fingerprint.

The intrinsic coordinate system of a fingerprint is defined by the DF. One of its axes runs along the ridge-valley structures, while the other is perpendicular to them. Using the intrinsic coordinate system, minutiae positions can be defined with respect to positions in the DF, instead of using the pixel coordinates as minutiae locations, thus providing a more natural representation. If a fingerprint undergoes elastic distortions, the distortions do influ-

ence the shape of the coordinate system, but the intrinsic minutiae coordinates do not change. This means that matching the intrinsic coordinates of the minutiae sets is invariant to elastic distortions.

The rest of this paper is organized as follows. First, Section 9.2 proposes a method to partition the fingerprint in regular regions. Then, Section 9.3 defines the intrinsic coordinate system, which is used in Section 9.4 for a the minutiae matching algorithm. Finally, Section 9.5 presents some preliminary results.

9.2 Regular Regions

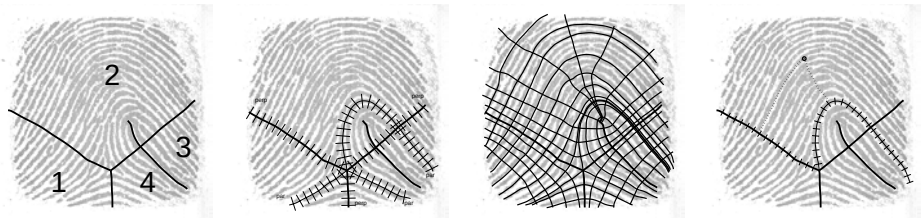
In this section, a method is proposed to partition the directional field in regular regions, which are the basis for the intrinsic coordinate system that is discussed in Section 9.3. The first step is extraction of the singular points (SPs) from the directional field (DF), using the methods that are presented in Chapters 2 and 3. The orientation of the SPs is used for initializing the flow lines.

The *flow lines*, which are traced in the DF, are used to partition the fingerprint into a number of regular regions. Flow lines are curves in a fingerprint that are exactly parallel to the ridge-valley structures. However, we also use this name for curves that are exactly perpendicular to the ridge-valley structures. Flow lines are found by taking line integrals in the directional field. This is discretized by a numerical Runge-Kutta integration, giving the following expression for tracing a flow line from a start position x_i which is a complex number that represents pixel coordinates in the fingerprint:

$$x_{i+1} = x_i + \Delta_x \cdot \text{mean}(DF_{x_i}, DF_{x_{i+1}}) \tag{9.1}$$

In this equation, Δ_x is the step size, DF_{x_i} is a unit-length complex number that indicates the orientation of the DF at x_i, $\text{mean}(DF_{x_i}, DF_{x_{i+1}}) = (DF_{x_i}^2 + DF_{x_{i+1}}^2)^{1/2}$ as discussed in [Baz00a] and $DF_{x_{i+1}} = DF_{x_i + \Delta_x \cdot DF_{x_i}}$. For the perpendicular flow lines, steps that are perpendicular to the DF should be taken. This method of tracing flow lines gives relatively small errors and causes circular contours to be closed.

The extracted SPs and the flow lines are used to partition the fingerprints in so called *regular regions*, which are region in which no singular points are located. In order to construct the regular regions, two sets of flow lines have to be traced. From each *core*, a curve is traced that is parallel to the ridge-valley structure, and from each *delta*, three curves are traced that are perpendicular to the ridge-valley structure. This scheme provides a partitioning in regular regions for all classes of fingerprints and is illustrated in Figure 9.1(a) for a “right loop”. The most important property of regular regions is that the fingerprint in such a region can be warped non-linearly such that it only contains straight parallel ridge-valley structures. This is illustrated in Figure 9.2.



(a) Regular regions (b) Reference axes (c) ICS (d) IMCs

Figure 9.1: Construction of the intrinsic coordinate system using flow lines.

Figure 9.2: Some intermediate stages in the non-linear warp of regular region 2.

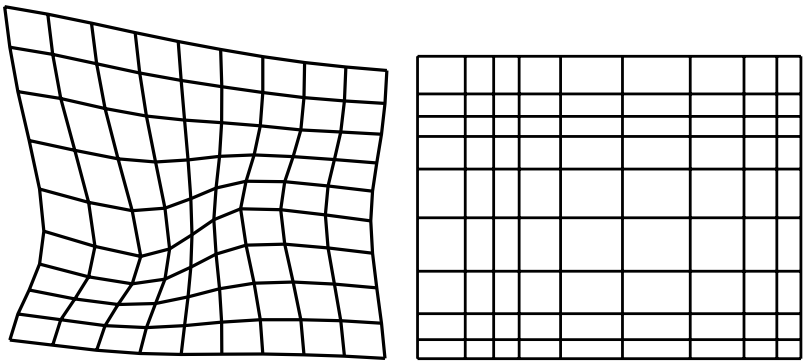
9.3 Intrinsic Coordinate System

The next step is to construct the *intrinsic coordinate system* (ICS) for each regular region in the fingerprint. This coordinate system is called intrinsic since it is the coordinate system that is defined by the DF of the fingerprint itself. Using the ICS, a multi-level description of the fingerprint can be made, in which the minutiae positions (fine level) are given with respect to their relative position in the DF (coarse level). Points in the fingerprint that are on the same ridge will have the same *perpendicular coordinate*, while all points on a curve that is perpendicular to the ridges share the same *parallel coordinate*.

The intrinsic coordinate system is defined in each regular region by two *reference axes*, which is illustrated in Figure 9.1(b). For each regular region, the first reference axis is given by the perpendicular curve through the delta, while the second reference axis is given by the parallel line through the delta. When there are no deltas in the fingerprint, there is only one regular region. In this case, any combination of one parallel and one perpendicular flow line can be taken as reference axes.

The resulting ICS grid can be visualized by tracing curves from equally spaced positions along the reference axes, as illustrated in Figure 9.1(c). Although the grid is equally spaced along the reference axes, this is not the case in the rest of the fingerprint. The parallel curves may for instance diverge because a ridge that is between them bifurcates.

The *intrinsic minutiae coordinates* (IMCs) can be determined directly from the DF by means of projection of the minutiae on the intrinsic axes. The parallel coordinate of a minu-



(a) Elastic Deformations (b) Application of ICS

Figure 9.3: Application of the ICS to a deformed grid.

tia is found by tracing a perpendicular curve until it crosses a parallel reference axis. The distance along the axis of this point to the origin of the ICS gives the parallel coordinate. The perpendicular coordinate of a minutia is found by tracing a parallel curve, until it crosses a perpendicular reference axis. The distance along the axis of this point to the origin of the ICS gives the perpendicular coordinate. This method is illustrated in Figure 9.1(d).

9.4 Minutiae Matching in the ICS

A *minutiae matching* algorithm has to determine whether both fingerprints originate from the same finger by comparing the minutiae sets of a template fingerprint (stored in a database) and a test fingerprint (provided for authentication). The goal of the algorithm is to find the mapping of the test to the template minutiae set that maximizes the subset of corresponding minutiae in both sets. If the size of this common subset exceeds a certain threshold, the decision is made that the fingerprints are matching.

In this section, an alternative minutiae matching method that makes use of the ICS is proposed. As a consequence of the definition of the ICS, the IMCs only change in some simple and well-defined ways. Since distortions do not affect the ordering of the IMCs of neighboring minutiae, dealing with distortions amounts to independent 1-dimensional non-linear dynamic (time) warps (see [Rab93]) in 2 directions. This is a huge reduction in the number of degrees of freedom and computational complexity, compared to a 2-dimensional non-linear dynamic warp. The effect of application of the ICS on the elastic deformations is illustrated in Figure 9.3.

In the ICS, minutiae matching reduces to finding the warp function that maximizes the number of minutiae that fit exactly. Once the minutiae sets have been ordered along one intrinsic coordinate, the problem is to find the largest subset of both minutiae sets in which the ordering in the other intrinsic coordinate exactly corresponds. This problem can be solved

by dynamic programming as described below. Usually, the next step would be to determine the warp function that interpolates the points that were found. However, since we are only interested in the number of matching minutiae, this step does not have to be performed.

The set of minutiae of the primary fingerprint is given by $\mathbf{a} = [a_1, \ldots, a_m]$. A minutia a_i of this set is described by its parallel coordinate $x(a_i)$ and perpendicular coordinate $y(a_i)$. The set is sorted by $x(a_i)$. In the same way, **b** describes the n minutiae of the secondary fingerprint. The problem is to find the longest series $[(a_{i_1}, b_{j_1}), \ldots, (a_{i_k}, b_{j_k})]$ with $1 \leq i_1 < \cdots < i_k \leq m$ and $1 \leq j_1 < \cdots < j_k \leq n$ such that a criterion of local similarity is satisfied:

$$a_{i_k} - a_{i_{k-1}} \approx b_{i_k} - b_{i_{k-1}} \tag{9.2}$$

Consider the partial solution $\lambda(i, j)$, representing the longest series of minutiae pairs that has (a_i, b_j) as the last pair in the series. This means that $\lambda(i, j)$ can be constructed recursively by adding (a_i, b_j) to the longest of all series $\lambda(k, l)$, with $k < i$ and $l < j$, behind which the pair fits:

$$\lambda(i, j) = \lambda(k, l) + (a_i, b_j) \tag{9.3}$$

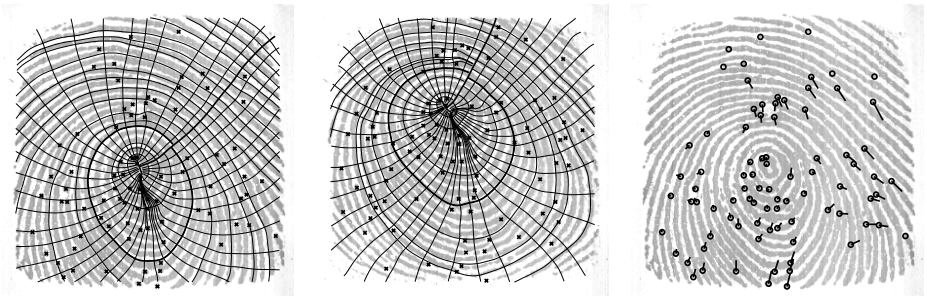
with k and l chosen to maximize the length of $\lambda(i, j)$. Using the starting conditions $\lambda(1, j) = [(a_1, b_j)]$ and $\lambda(i, 1) = [(a_i, b_1)]$, the final solution is the longest of all calculated partial solutions $\lambda(i, j)$.

9.5 Preliminary Results

Although the elastic distortion-invariant minutiae matching algorithm is not yet operational, this section is meant to show the potential of our algorithm. In order to make the matching result only dependent on the ability to deal with elastic distortions, and not on the minutiae extraction algorithm, the minutiae were extracted by human inspection. After cross-validation of both sets, 77 corresponding minutiae were in the two prints of the same finger that are shown in Figure 9.4(a) and 9.4(b).

In Figure 9.4(c), the sets have been aligned as well as possible, using only translation, rotation and scaling. In the figure, corresponding minutiae are connected by a line. The figure clearly shows that, although the two minutiae sets fit exactly at the center of the print, they do not fit very well in the rest of the fingerprint. Under these distortions, the displacements required to tolerate corresponding minutiae is larger than the distance between some neighboring minutiae. Therefore, tolerance of small displacements is not a solution in this case. For a reasonable tolerance, only 24 matching minutiae pairs are found, out of 77 true matching minutiae pairs.

When the ICS is used, the perpendicular ordering is preserved since the parallel lines exactly follow the ridge-valley structures. However, the elastic distortions can influence the parallel ordering. Especially the parallel coordinate of minutiae near the core are less reliable. Nevertheless, over 70 correctly ordered minutiae pairs can be found using the ICS.



(a) Fingerprint 1 (b) Fingerprint 2 (c) Superimposed

Figure 9.4: Minutiae sets located in the ICSs of two fingerprints of the same finger.

9.6 Conclusions

In this chapter, the intrinsic coordinate system of a fingerprint is presented. It is defined by the directional field of the fingerprint itself. The locations of minutiae are insensitive to elastic distortions if they are given in this intrinsic coordinate system. It is shown how minutiae-based matching is drastically simplified when minutiae are characterized by means of their intrinsic coordinates.

Part III

Database Search

Chapter 10

Classification

Abstract

Classification of fingerprint images can be used as a first step in fingerprint identification systems. All template fingerprints are classified, and a test fingerprint is matched only to the template fingerprints of the corresponding class. This reduces the number of matches that is required, and therefore the processing time and error rate. This chapter summarizes the so called “Henry classification” system. However, given the presence of classification errors and the limited reduction of the number of required matches, even for perfect classification, Henry classification is more an issue of academic interest and backward compatibility than of practical use. Part of this chapter has been published in [Boe01] and [Baz02a].

10.1 Introduction

The task of a classification systems is to determine to which class or category the input fingerprint belongs. Classification is an important first step in any identification system. The task of a fingerprint identification system is to find the fingerprint in a database that matches a query fingerprint. A naive identification system would simply compare the given fingerprint to all entries in the database. However, for databases of a realistic size, two closely related problems are encountered when applying this approach. Since many matches are performed, the required processing time will be excessive, and the performance will be inadequate. This is illustrated by a simple example, using the *system penetration coefficient* P, which measures the expected number of comparisons to be made [Way99].

$$P = \frac{E[\textit{number of comparisons}]}{n} \tag{10.1}$$

Consider the identification of a fingerprint in a database of $n = 10{,}000$ entries. The naive matching system compares the query fingerprint to all entries in the database, so $P = 1$. Since the probability of false acceptance is larger than zero, the search cannot be stopped if a matching fingerprint is found, and the best matching fingerprint is chosen. A fast matching algorithm, requiring only $t_{match} = 100$ milliseconds per match, needs almost 17 minutes for this task, and a matching algorithm with a very good performance of $FAR = 10^{-4}$ will find on average $n \cdot FAR = 1$ false match in this database. Furthermore, the probability of false acceptance over the entire database can be computed as: $FAR_{1:n} = 1 - (1 - FAR)^n = 0.63$. Obviously, these performance figures are unacceptable for any identification system.

To reduce these problems, classification is used [Way99]. In the absence of classification errors, this technique is known as hashing. All fingerprints in the database are classified. They are stored per class in separate databases. The query fingerprint is also classified and is only matched to the n_i fingerprints in that part of the database that contains fingerprints of the corresponding class i. The effect is two-fold, as illustrated when we carry on with the previous example. Assume that the fingerprints are classified into 100 equal-size classes and that therefore $n_i = 100$ for all i. The processing time will thus come down to 10 s, and the average number of false matches will be $n_i \cdot FAR = 0.01$ and $FAR_{1:n_i} = 10^{-2}$. However state-of-the-art fingerprint identification systems cannot identify 100 distinct classes. Therefore, the size of the fingerprint database is limited to a few hundred entries for reasonable performance and identification time [Cap00c].

10.2 Henry Classes

A well-known set of categories is formed by the *Henry* classes. They consist of five classes related to global fingerprint patterns, based on the structure of the directional field. Examples of fingerprints from the five Henry classes (whorl, left loop, right loop, arch, and tented arch) are shown in Figure 10.1. Sometimes, a 4-class Henry classification is used, where arch and tented arch are combined into a single class. On the other hand, it is also possible to define more than five classes, for example adding pocketed loops, double loops, and scars.

(a) Whorl

(b) Left Loop

(c) Right Loop

(d) Arch

(e) Tented arch

Figure 10.1: Examples of fingerprints from the five Henry classes.

Many ways to classify fingerprints to one of the Henry classes are known from literature. In [Kar96], fingerprints are classified in the Henry classes by examining the relative positions of the singular points. In [Cap99], a graph of homogeneous regions in the directional field is constructed and the graph is classified. In [Jai99a], FingerCode (see Section 11.2.2) is used as a feature vector for Henry classification. The most popular classification approach is the application of principal component analysis (PCA) or Karhunen-Loève transform (KL) [Hay99] to the block directional field (BDF) estimate [Can95]. Each element in the BDF is represented by a vector of two components that points in the direction of the local ridge orientation. The resulting vectors are concatenated to form a feature vector. Next, the dimensionality of this feature vector is reduced by the application of PCA. This method removes the redundancy and noise, while maintaining the maximum amount of information in the feature vector. The resulting small feature vector is classified by a neural network. Multiple fingerprint classifiers can be combined to improve their performance [Cap00b].

There are two reasons why Henry classification is not able to reduce the size of the database to be searched very much. First, there are only five Henry classes and 90% of all fingerprints belongs to only three classes. Without classification errors, this would reduce the number of possible matches to approximately 30% of the original, thus allowing a database size of three times as large as without classification for the same processing times and error rates.

Second, the state-of-the-art classifiers at the moment perform at an error rate of about 6% for the Henry classes [Cap00b]. This means that for 6% of the query prints, the corresponding print is not found correctly in the database due to classification errors. Alternatively, 20% of the query fingerprints is rejected for classification, resulting in an error rate of 1%. The effect of the rejection is that for 20% of the fingerprints the entire database has to be searched, while 1% of the prints still remains non-identified due to classification errors. From these figures, one can conclude that the practical value of Henry classification is limited.

10.3 Conclusions

A naive identification system matches the query fingerprint to each template in the database. This results in unacceptable processing times and error rates for databases of only moderate sizes. Classification can be used as a first selection mechanism to reduce the number of matches that is required. However, traditional Henry classification is not able to provide enough reduction of the size of the database to be searched in order to be of practical use.

The fact that there is still much research on Henry classification can be explained by the fact that it is an interesting classification problem. Since many approaches are known from the literature, it has become an issue of academic interest. Furthermore, many existing fingerprint databases are partitioned according to the Henry classification. To use those databases without re-processing them for use with better applicable classification schemes, automatic Henry classification still has to be used.

Chapter 11

Indexing Fingerprint Databases

Abstract

In a fingerprint identification system, a person is identified only by his fingerprint. To accomplish this, a database is searched by matching all entries to the given fingerprint. However, the maximum size of the database is limited, since each match takes some amount of time and has a small probability of error. A solution to this problem is to reduce the number of fingerprints that have to be matched. This is achieved by extracting features from the fingerprints and first matching the fingerprints that have the smallest feature distance to the query fingerprint. Using this *indexing* method, modern systems are able to search databases up to a few hundred fingerprints.

In this chapter, three possible fingerprint indexing features are discussed: the registered directional field estimate, FingerCode and minutiae triplets. It is shown that indexing schemes that are based on these features, are able to search a database more effectively than a simple linear scan. Next, a new indexing scheme is constructed that is based on combining these features. It is shown that this scheme results in a considerably better performance than the schemes that are based on the individual features or on more naive methods of combining the features, thus allowing much larger fingerprint databases to be searched. Part of this chapter has been published in [Boe01].

11.1 Introduction

Identification systems only receive one input, a *query* fingerprint. This might for instance be used to check whether new users are already known by the system with another name, to identify non-cooperative users, or for checking against a black list. In this case, the system has to search a database for a matching fingerprint. This task is also referred to as one-to-many matching. If a matching fingerprint is found, this identifies the person.

To overcome some of the problems of Henry classification, discussed in Chapter 10, a *continuous classification* scheme is proposed in [Lum97]. Again a feature vector that describes the global structure of the fingerprint is extracted. Instead of classifying a feature vector and matching the query fingerprint to all prints in the corresponding class, now the fingerprint is first matched to those fingerprints of which the feature vector is most similar. For this *indexing* method, the maximum part of the database to be searched is adjustable.

Continuous classification as proposed in [Lum97] slightly increases the performance, but is still not acceptable for commercial use in large databases. In realistic modern fingerprint identification systems, the size of the fingerprint database is limited to a few hundred entries for reasonable performance and identification time [Cap00c]. Further improvement of the performance can be expected from the combination of multiple features for indexing, which is one of the topics of this chapter.

The rest of this chapter is organized as follows. First, in Section 11.2, a number of possible fingerprint indexing features are discussed, being the registered directional field estimate, FingerCode and minutiae triplets. It is shown that indexing schemes that are based on these features are able to search a database more effectively than a simple linear scan. Next, in Section 11.3, a new indexing scheme is constructed that is based on combining these features. In Section 11.4, experiments are presented showing that this scheme results in a considerably better performance.

11.2 Indexing Features

Due to measurement noise and elastic distortions, fingerprints cannot be compared by just calculating the cross-correlation or the Euclidean distance of the gray scale images. Therefore, features are extracted from the fingerprints that are more robust to the distortions. Choosing the best feature is the most important part of the design of a identification system and greatly affects the performance and accuracy of the system. A good feature vector should have the following properties:

- discriminating over large number of fingerprints,
- invariant for rotation, displacement and plastic distortions,
- compact,
- easily computable.

In this section, a number of possible features that can be used for indexing fingerprint databases are discussed.

11.2.1 Directional Field

The most straightforward feature that describes the shape of a fingerprint is to use the directional field estimate directly. In order to keep the feature vector relatively small, the DF is calculated blockwise, which means that only one DF value is calculated in each block of for instance 16×16 pixels.

Each element in the DF is represented by a vector of 2 components that points in the direction of the local ridge orientation. The SDF is used, in order to give orthogonal orientations the largest distance. The resulting vectors are concatenated to form a feature vector. Next, the dimensionality of this feature vector is reduced by the application of the principal component analysis (PCA) or Karhunen-Loève transform (KL) [Hay99]. This method removes the redundancy and noise, while maintaining the maximum amount of information in the feature vector. We have found that a feature vector length of 30 gives the best results for Database 2 of FVC2000.

It is important that the DF estimates are *registered* well, which means that they are exactly aligned. For this task, the uppermost core is used. However, in some cases, this might not be easy to achieve. First, live-scanned fingerprints only capture a small area of the entire fingerprint. Therefore, some singular points may fall outside the actual image. Second, an arch (see Figure 10.1(d) on page 131) does not contain any singular points, and an alternative registration point has to be defined. Third, a DF that is estimated from a low-quality fingerprint image, may contain a number of false SPs. They can possibly be eliminated by high-resolution segmentation methods, as discussed in Chapter 4.

Similar feature vectors have been used by other researchers as well for fingerprint classification. In [Can95], this feature vector is the input for a probablistic neural network for Henry classification. In [Cap00c], this feature vector is used for indexing databases. However, a multi-space Karhunen-Loève (MKL) approach is used for calculation of the distances to the Henry classes, and this new feature vector is used for indexing the database.

11.2.2 FingerCode

The second feature vector we will use for indexing is the FingerCode [Jai99a, Jai00b]. This method filters the fingerprint image by a bank of Gabor filters that are tuned to different orientations. Next, the fixed-length feature vector, which is called FingerCode, is computed as the standard deviation in a number of sectors, which is indicative of the overall ridge activity in a certain orientation in that sector. The authors expect that this feature vector captures both global and local characteristics of the fingerprint image.

Calculation of the FingerCode from a fingerprint image is done as follows. First the reference point is determined. Again, the uppermost core can be used. Then, the fingerprint image

Figure 11.1: FingerCode: region of interest around the core, divided in 48 sectors.

is filtered by a bank of Gabor filters (see Appendix D) that are oriented in eight directions[1]. Finally, for each filtered image the standard deviation of the gray values is computed in a number of predefined sectors around the reference point. The vector of standard deviations is called FingerCode.

For the calculation of FingerCode, 8 values of θ are chosen that are equally spaced from 0 to π with respect to the x-axis. The value for σ is determined empirically and set to approximately half the inter-ridge distance. Before the filtering is applied, the fingerprint image is normalized to compensate for intensity variations due to pressure variations.

Around the core, 3 concentric circular regions of 20 pixels wide, which is approximately twice the inter-ridge distance, are defined and each region is divided in 16 sectors. This is illustrated in Figure 11.1. The immediate region around the core is not used, because the sectors would be too small and would not have enough pixels to calculate a reliable standard deviation. The standard deviations of each filtered image in each sector form the 384-dimensional feature vector that is called the FingerCode. An example FingerCode is shown in Figure 11.2, where each disk corresponds to one filtered image.

FingerCode has already been used in a fingerprint matching system [Jai00b] and for the classification of fingerprints in Henry classes [Jai99a], using a two-stage classifier which uses a K-nearest neighbor classifier in the first stage and a set of neural networks in the second stage. In this paper, we will demonstrate the use of FingerCode as a feature for indexing fingerprint databases.

11.2.3 Minutiae Triplets

The third feature vector is based on minutiae triplets. The algorithm has been proposed in [Ger97] and slightly adapted in [Bha01]. The algorithm first identifies all minutiae triplets in a fingerprint. Each triplet defines a triangle, from which various geometric features are extracted. One can for instance use the length of the sides, the angles, etc. Next, the similarity

[1] In Chapters 5 and 7 we used only four directions and the absolute value of the Gabor response. However, this research was performed earlier, and uses the original version of FingerCode as described in [Jai99a, Jai00b]

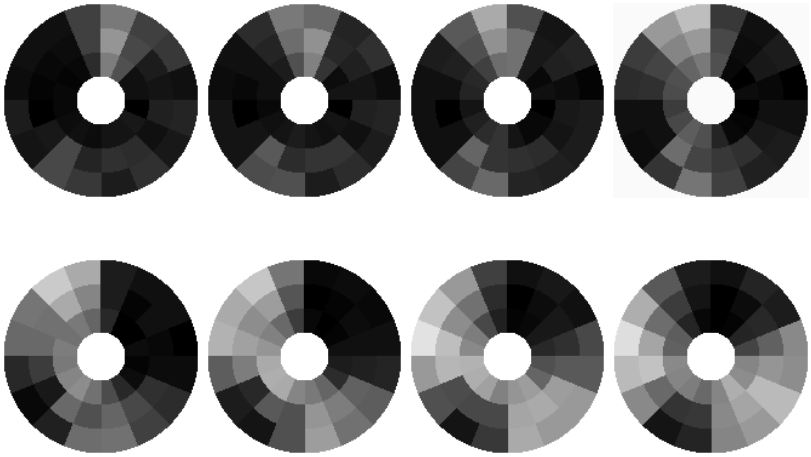


Figure 11.2: Example of a 384-dimensional FingerCode

to another fingerprint is defined by the number of approximately corresponding minutiae triplets that can be found by rigid transformations. An example of two matching fingerprints and the corresponding minutiae triplets is shown in Figure 11.3.

Instead of comparing the extracted triplets to each fingerprint, the authors use the Flash algorithm [Ger97] which constructs hash tables for fast indexing in a database. This is possible since, for each triplet, it is only necessary to determine whether the triplet is present in the other fingerprint or not.

The construction of the hash table is done as follows. A table is constructed that contains a cell for the quantized feature vectors of all possible triplets. Next, for all fingerprints in the database, the triplets are extracted and the ID of the fingerprint is added to the list in each cell that corresponds to a triplet that is found. The result is that each cell in the table contains a list of all fingerprints in which its corresponding triplet is present.

Using this table, indexing does not require a large amount of computation. Given a query fingerprint, all triplets are extracted and for each triplet, the list of fingerprints in which that triplet is present, is retrieved. A simple count of the occurrences of the fingerprints in all retrieved lists determines the ranking used for indexing.

11.3 Combining Features

When a number of different classifiers are available for a specific classification task, they will in general complement each other. Each classifier will misclassify different inputs, especially when they base their decisions on different feature vectors. This effect can be exploited by combining the decisions of the classifiers in order to improve the classification performance.

The outputs of different classifiers can be combined at several output levels [Sue00]. In

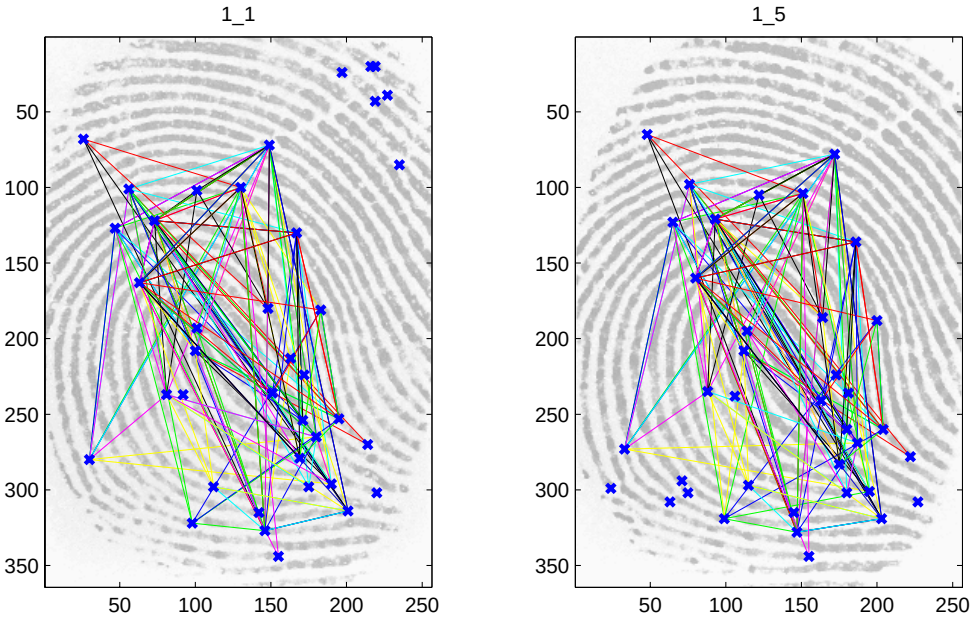


Figure 11.3: Matching fingerprints and corresponding minutiae triplets.

single class combination, the winning class outputs of the classifiers are combined. In *ranked list* combination, the new indexing is derived from the output rankings of the individual classifiers. In *measurement level* combination, the probabilities of the different classes as reported by the classifiers, are combined directly.

In this chapter, we consider the problem of continuous classification. In that case, each fingerprint in the database is defined as a different class. This is a classification problem with as many classes as the number of fingerprints in the database. Therefore, the correct class may not be the highest-ranked class for any of the classifiers. For this reason, single class combination is unlikely to be successful. Ranked list combination has a better chance for success provided that the correct class is sufficiently often close to the top of the list for at least one of the classifiers.

There exist several methods of combining ranked lists from different classifiers, aiming at achieving a new ordered list in which the true class is ranked higher than at the original lists. This is called *class set reordering* [Ho94]. The *highest rank* method can be applied very well when there are many classes and only a small number of classifiers, each of which specializes on inputs of a particular type. The new rank of a class is the highest of the ranks that are given by the individual classifiers. Another method is the *Borda count*, which is a generalization of the majority vote, providing a consensus ranking. The Borda count for a class is the sum of the number of classes ranked below it by each classifier, and the new ranking is given by rearranging the classes so that their Borda counts are in descending order. *Logistic regression* is a modification to the Borda count such that different weights are given to the scores produced by each classifier.

Another interesting concept in combining ranked lists is *class set reduction*. There are two methods of class set reduction that both aim at reducing the number of classes in the output list without losing the true class. The first method determines for each classifier the worst ranking ever given to the true class in the training set. When using the classifier, classes that receive a lower rank than this threshold are discarded. The second method determines for each training example the best ranking over all classifiers and stores that value with the best classifier. Then, for each classifier, the worst of all best rankings is determined. Each possible class is discarded if its ranking is worse than this threshold for each classifier.

Combination of the classifier outputs at the measurement level provides even more information than combining ranked lists. In [Kit98] it has been shown that the sum rule, which sums or averages the posterior probabilities of the classes, is the most robust combination method for multiple classifiers. However, one difficulty arises when applying this combination rule to classifiers that do not output posterior probabilities. This is the case with the classifiers we use, which output distance and similarity measures. These quantities first have to be transformed to posterior probabilities before the combination rule can be applied.

In [Cap00b], the combination of the measurement level outputs of different classifiers has been applied to indexing fingerprint databases. In that paper, two indexing features are combined by taking a weighted average of a non-linear function of the two feature-distances. It has been shown that this increases the performance of the indexing scheme. However, the increase of performance is only significant for small parts of the database to be searched. When higher matching rates are required, the combination does not perform better than the individual classifiers.

11.4 Experimental Results

We have run our experiments on Database 2 of the FVC2000 contest [Mai00, Mai02]. This database contains fingerprint images that are captured by a capacitive sensor with a resolution of 500 pixels per inch, resulting in images of 364×256 pixels in 8 bit gray-scale. In this database, 110 untrained individuals are enrolled, each with 8 prints of the same finger. The first of these prints is used to construct the database, while the other seven prints are used to test the classification performance.

Since our singular point detection algorithm [Baz01b, Baz02d] still detects some false singular points, the registration points that are used in the DF-based and FingerCode algorithms have been validated manually. In 13% of the fingerprints, the registration point has been corrected, while 1% of the fingerprints has been rejected in which no registration point could be found.

11.4.1 Single Features

The results of the identification of the remaining 7 prints per finger (one has been used to construct the database), using the three indexing features of Section 11.2, are shown in Figure 11.4. The vertical axis depicts the cumulative percentage of prints which are correctly

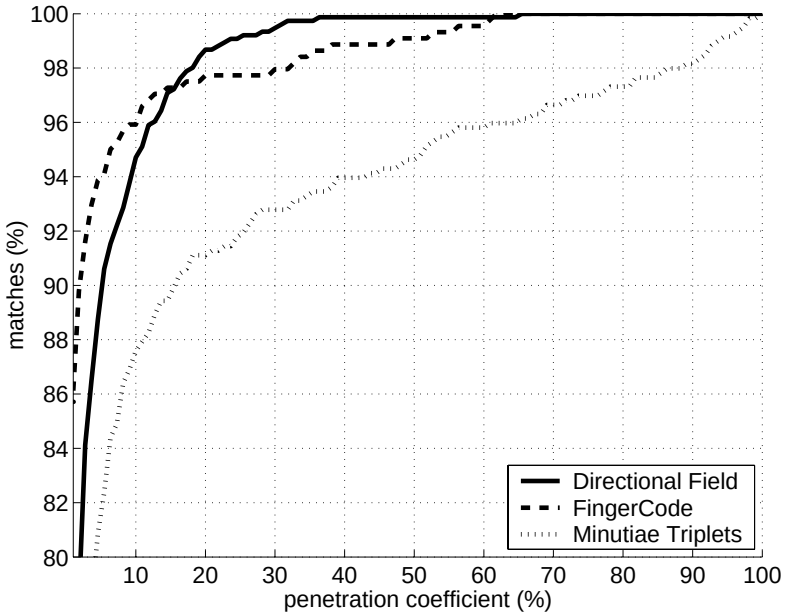


Figure 11.4: Indexing performance, based on only one feature.

identified when searching the part of the database shown on the horizontal axis. Some of these results are also summarized in Table 11.1.

The results of our DF-based method are better than the results found in [Cap00c], using the same database. In that article, a matching rate of 80% is reported when searching 10% of the database, while we achieved a matching rate of 95% when searching the same percentage of the database. This might be explained by the fact that we have manually changed the registration point for 13% of the fingerprints and the differences between KL and MKL.

Calculating a FingerCode from an image takes about 40 seconds on a 450 MHz Pentium II processor when using a prototype implementation in MATLAB. This is considerably longer than calculating the feature vector based on the directional field, which takes about 300 miliseconds in a C++ implementation. Opportunities to speed up the implementation of the FingerCode method exist, but had not yet been investigated at the time of writing.

The method that is based on minutiae triplets performs much worse than the results that were reported in literature. In [Bha01], 100% of the fingerprints is found in the first 2.5% of a database of 400 fingers. However, when searching 2.5% of our database, only 77% of the fingerprints is found. This may be caused by the use of another (private) database. Another possible explanation of the discrepancy may be that some aspects of the implementation are not explicitly mentioned in [Bha01].

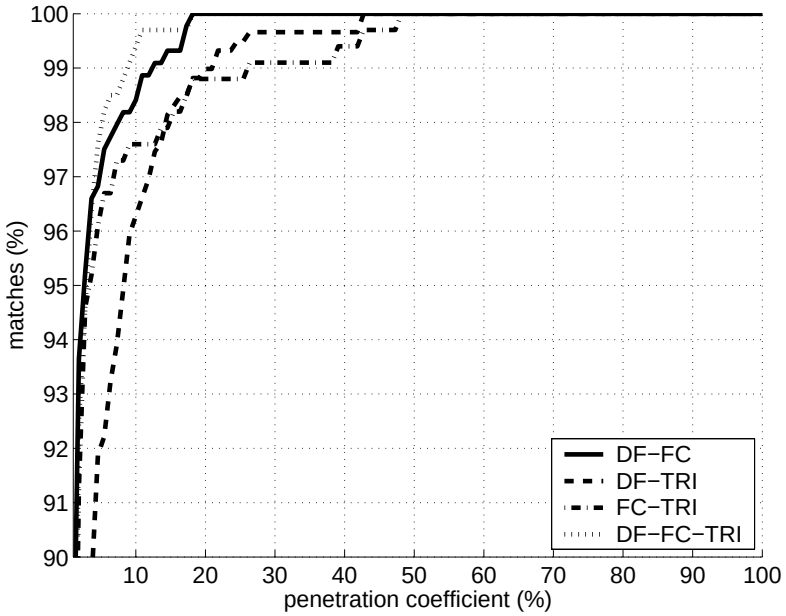


Figure 11.5: Indexing performance of multiple features using highest rank.

Table 11.1: Indexing performance: penetration coefficient by various matching rates.

Matching rate	90%	95%	99%	100%
Directional Field	5%	10%	23%	65%
FingerCode	2%	6%	47%	62%
Triplets	16%	51%	94%	100%
DF + FC + Tri, HR	2%	3%	9%	18%

11.4.2 Multiple Features

Next, a number of experiments have been performed to investigate the effect of using a combination of multiple features for indexing the database. All combinations of 2 and 3 features were combined by the highest rank and Borda count methods. The resulting indexing performances are shown in Figures 11.5 and 11.6. The most interesting results are summarized below and shown in Table 11.1.

- The best performance is achieved by combining all three features by means of highest rank. Using this method, 100% match is found when using a penetration coefficient of only 18%, while the lowest penetration coefficient for single features is achieved by FingerCode with 62% search for 100% match.
- For all combinations of features, highest rank outperforms Borda count. This can be explained by the fact that we use many classes and only a few classifiers.
- The combination of all three features by highest rank performs better than the individual features, while the combination of all three features by Borda count performs worse

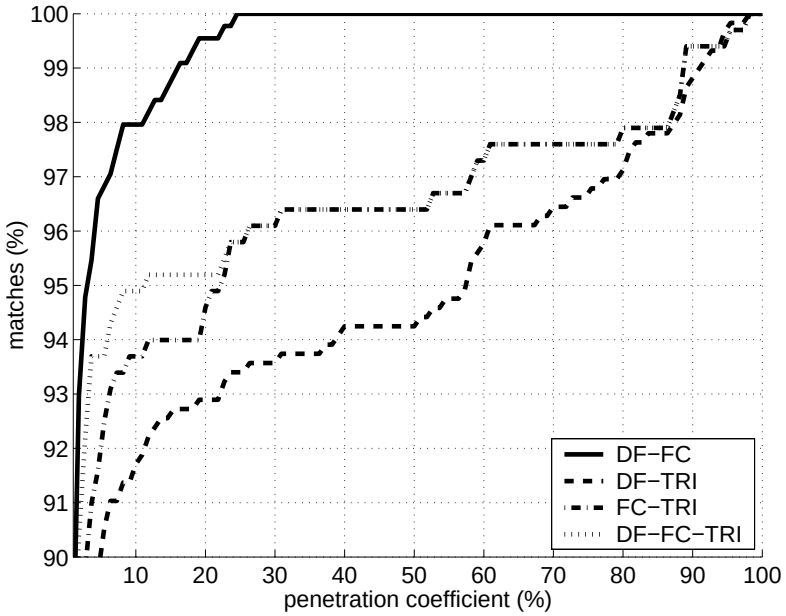


Figure 11.6: Indexing performance of multiple features using Borda count.

Table 11.2: Indexing performance: average part of the database to be searched.

Features	Incl. match	Excl. match
None	50.45%	49.54%
Directional Field	2.58%	1.67%
FingerCode	2.40%	1.49%
Triplets	7.27%	6.36%
DF + FC + Tri, HR	1.34%	0.43%

than the individual DF and FingerCode features.

- Adding minutiae triplets to any combination of features does not improve the performance significantly. For combination by means of Borda count, this even decreases the performance. The application of logistic regression might solve this problem by assigning a smaller weight to the minutiae triplet classifier.

The results that are shown in Table 11.1 indicate the percentage of the database that has to be searched in order to achieve a fixed probability that the corresponding fingerprint is found. However, the average percentage can be reduced by stopping the search once a match is found. This provides another measure of the indexing performance, which is shown in Table 11.2.

The first column of this table shows the average size of the part of the database to be searched until the matching fingerprint is found, including the true match. The use of DF or FingerCode as single feature reduces this part from approximately 50% to approximately

2.5% of the database, while the combination of features further reduces the size of the partition to less than 1.5%.

However, this part includes the actual matching fingerprint, which occupies 0.9% of a database of 110 fingerprints. Only the number of non-matching prints that have to be matched are responsible for the increased FAR. This percentage is reduced to less than 0.5% of the database by combining multiple features. Compared to a simple linear search, this allows the size of databases to be 100 times as large, while maintaining the same FAR and FRR.

11.5 Conclusions

In this chapter, three possible fingerprint indexing features have been discussed: the registered directional field estimate, FingerCode and minutiae triplets. It has been shown that indexing schemes that are based on these features are able to search a database more effectively than a simple linear scan.

Next, a new indexing scheme has been constructed that is based on combining these features. It has been shown that especially the highest rank combination of DF, FingerCode and minutiae triplets results in a considerably better performance than the schemes that are based on the individual features. Compared to a simple linear search, this allows the size of databases to be 100 times as large, while maintaining the same FAR and FRR. A further improvement might be achieved by using logistic regression as combination rule and by the additional use of class set reduction.

A number of possibilities for improving the performance of fingerprint database indexing exist. First, other features could be added, like for instance the absolute value of the complex Gabor response on a rectangular grid, as defined in Appendix D and applied in Chapter 7 to the matching problem. Second, the Euclidean distance (as proposed for FingerCode [Jai00b]) can be replaced by likelihood ratios, as proposed in Chapter 7 for verification.

Chapter 12

Conclusions and Recommendations

This thesis focusses on three principal challenges in fingerprint recognition: robust feature extraction from low-quality fingerprints, matching elastically deformed fingerprints, and efficient search of fingerprints in a database. Research into these challenges has yielded improvements for most of them, as will be pointed out in Section 12.1. A few directions for further research into fingerprint recognition are presented in Section 12.2.

12.1 Conclusions

In this thesis, the matching performances of three different fingerprint recognition configurations have been evaluated. The correlation based system of Chapter 8 participated in FVC2000, resulting in an equal error rate of EER = 10%. The elastic minutiae matching system of Chapter 6 participated in FVC2002, resulting in EER = 2% on the training database. The fixed-length feature vector-based system of Chapter 7 has been tested on the FVC2000 databases, resulting in EER = 0.5%. Unfortunately, there was no time to integrate all components into a system that could be tested in a real-life situation. That would certainly reveal weaknesses that have not been found in the laboratory on test databases.

The best fingerprint recognition system that can be constructed from the work of this thesis combines two matching algorithms: the elastic minutiae matching and the fixed-length feature vector matching. Since the features of both algorithms (minutiae sets and fixed-length feature vectors) are virtually uncorrelated, the equal error rate of this system is expected to be somewhere in the range from 10^{-3} to 10^{-4}. If multiple prints of the same finger are used for template construction, using simple averaging of the fixed-length feature vectors and elastic mosaicking methods for the minutiae sets (see Section 12.2.2), the equal error rate is expected to be smaller than 10^{-5}. Furthermore, large databases can be searched reliably with this system by first applying the fixed-length feature vectors as selection phase and then the elastic minutiae matching as matching phase.

This section continues with a summary of the conclusions of the individual methods. A new method has been proposed for the estimation of a high resolution directional field of fingerprints. This method computes the local ridge orientation in each pixel location, and the associated coherence, which provides a measure of its reliability. By decoupling the size and shape of the smoothing window from the block size that defines the resolution of the estimate, the proposed method combines an improved quality of directional field estimates, better noise suppression, and low computational complexity. Furthermore, a very efficient algorithm has been proposed to consistently extract all singular points and their orientations from this high-resolution directional field. The algorithm provides a binary decision without using thresholds, and is implemented efficiently in small 2-dimensional filters.

An improved algorithm for the segmentation of fingerprint images has been proposed that accurately segments fingerprint images into foreground, background, and low quality areas. The method combines several pixel features into a hidden Markov model in order to include context information, and to obtain spatially compact clusters. Furthermore, an efficient method for minutiae extraction has been developed that roughly follows the traditional approach, involving a number of image processing steps such as enhancement, thresholding, and thinning. A combination of good noise suppression performance and low computational complexity is achieved by an enhancement algorithm that uses separable complex-valued Gabor filters, while efficient minutiae extraction is obtained by look-up table operations.

To fill the lack of common benchmarks for the evaluation of different minutiae extraction methods, a new method has been proposed that does not require manual labelling of ground truth minutiae. It evaluates the consistency of the extracted minutiae over different prints of the same finger, i.e. it is based on the functional requirements to minutiae extraction algorithms. The method provides maximal feedback of the types of extraction errors that are made: for each extracted minutia, it indicates whether it is false, missed, displaced, or genuine.

To compare the minutiae sets of two fingerprints, an elastic minutiae matching algorithm has been proposed that is the first algorithm that explicitly deals with elastically distorted fingerprints. It uses thin-plate splines to model the elastic deformations and takes less than 100 ms on a 1 GHz Intel Pentium III machine. The elastic minutiae matching algorithm is able to determine the correspondence between elastically deformed fingerprints, where the traditional rigid matching algorithms fail.

An alternative fingerprint matching method has been proposed that uses fixed length feature vectors instead of minutiae sets. The new algorithm achieves higher matching performance than the well known FingerCode algorithm, not only by using more types of features, but also by accounting for the inter and intra class covariance matrices in its matching decision. Because of the very low computational complexity, matching algorithms that do not use minutiae are especially suited for searching fingerprint databases. Additional minutiae matching remains necessary for achieving a very low FAR.

It has been shown that the common practice of partitioning a fingerprint database into the five Henry classes is not an efficient solution for searching large databases, as it reduces the effective size of the database to be searched only to 30%. In contrast, the proposed indexing method is able to reduce it to less than 0.5% of its original size by combining the classification results of several feature vectors. This allows the size of the database to be 60 times as large,

while maintaining the same processing time and error rates, compared to the situation where Henry classes are used.

The state-of-the-art in fingerprint recognition research can be summarized as follows. Continuous research is performed on fingerprint sensors, aiming at higher image quality and the development of smaller and less expensive sensors. Feature extraction algorithms perform at a satisfactory level for high quality fingerprints, while they often fail when applied to low quality fingerprints. Given accurately extracted features, matching performs well, but searching large databases still remains one of the bigger challenges.

12.2 Recommendations

Despite the achievements of this project, a number of problems remain to be solved in fingerprint recognition. Further research should focus on three specific issues: robust feature extraction from low quality fingerprints, construction of robust templates by the integration of information from multiple prints of the same finger, and issues related to large databases. This section gives an exploration of the possible approaches to these issues.

12.2.1 Feature Extraction

This section proposes a method for robust feature extraction from low quality fingerprints in three processing steps: segmentation, reconstruction of the directional field, and reconstruction of the gray-scale image. As a basis for the second and third processing steps, it is essential to obtain very accurate estimates of the foreground, background and low quality areas of the fingerprint images. In our experience, hidden Markov model based segmentation methods have the highest potential. The performance of these methods can be improved further by several adaptations. First, instead of using a Gaussian probability density function for each class, more realistic probability density functions can be considered, such as Gaussian mixtures. Second, more states can be used per class, representing for instance the orientations of the ridges in the fingerprint. This may help to estimate more accurate output probability density functions and constrain the transition matrix. Third, two dimensional hidden Markov models can be used to provide more accurate context information.

Once accurate segmentation is achieved, the directional field is reconstructed in the low quality regions. Directional field estimation methods perform well when applied to fingerprints of reasonable quality, but the occurrence of very low quality areas or scars, etc. may lead to inaccurate results. In these regions, the gradient distribution may be dominated by those defects, which contain gray-scale gradients that are much stronger than the gradients caused by the ridge-valley structures. These defects inevitably result in the detection of false singular points and a lower performance in the image enhancement and indexing stages.

A solution to the problem of inaccurate directional field estimates is the model-based reconstruction of the directional field in the low quality areas. For reconstruction purposes, two types of models can be used. First, statistical models that are trained on many examples of observed data can be used to estimate the most likely values of missing data, see for instance

[Tur91]. Second, specific directional field models, like the ones proposed in [She93, Viz96, Ara02], can be fitted to the data in the high quality regions, and provide the missing values. An open question remains how to estimate those models efficiently.

Given an accurate estimate of the directional field in low quality areas, the gray-scale image can be reconstructed in those areas, enabling the reliable extraction of additional minutiae. A decrease in the number of missed minutiae results in better matching performance and lower rejection rates. For the reconstruction of the gray-scale image, methods can be used that are similar to the generation of synthetic fingerprints [Cap00a]. One can think of Gabor filters that are controlled by the directional field, AM-FM reaction-diffusion [Pat01, Act01], or autoregressive filters [Vel90]. It is essential that the reconstructed directional field is highly accurate. Otherwise, the reconstruction methods will create spurious ridge-valley structures, resulting in false minutiae, and consequently a decrease of the matching performance.

12.2.2 Robust Templates

It has been shown in Chapter 7 that a large increases in matching performance can be achieved by the integration of multiple prints of one finger into one template in order to reduce the variance of the template. Since the feature vectors that are used in that chapter are already aligned sufficiently, the combination of multiple feature vectors is no more than a simple averaging operation, which reduces the intra class variation that is present in the template. Combination of templates that consist of minutiae sets are much more complicated. Since those templates represent the entire fingerprint area instead of one small part of it, two templates of the same finger will overlap only partially. Furthermore, elastic deformations prohibit a perfect fit of the overlapping area, as shown in Chapter 6.

Two methods for combining minutiae sets exist [Jai02]. The first method registers the minutiae sets, and uses the correspondences between the sets to create a new set. The second method, called mosaicking, combines the gray-scale fingerprint images after registration, and extracts a new minutiae set from the combined image. For both methods, elastic deformations of the fingerprints reduce the quality of the combined minutiae sets. These problems can be solved by applying the elastic minutiae matching algorithm, which eliminates partial misalignments. The combined template not only contains more reliable minutiae information, but also contains less elastic deformation by averaging the elastic deformations of all contributing fingerprints.

12.2.3 Large Databases

The use of large fingerprint databases gives rise to two kinds of problems. The first problem is that most algorithms have been trained (or optimized) on a relatively small database. For instance the widely used FVC2000 databases contain only 110 individuals each (of which only 10 are used for training in the actual competition). Therefore, the algorithms have experienced only a very small part of the variations in fingerprints that will occur in practice, which results in sub-optimal matching performance. A possible solution is to continue training of the matching algorithms online, while functioning in practice.

The second problem is the performance of classification or indexing algorithms, which are needed to search the databases reliably. State-of-the-art algorithms can be used to search databases that contain up to a few hundred fingerprints with acceptable error rates and computational time. However, for wide commercial application of fingerprint identification, algorithms that can handle much larger databases (containing for instance up to 100,000 fingerprints) are needed.

The indexing performance is limited by two causes. First, false singular points cause misalignment errors in the extracted indexing feature vectors. Using the methods that are proposed in Chapter 7 in combination with the reconstructed directional field probably eliminates most of these problems. The second type of errors is caused by the limited discriminating power of the feature vectors that are used for indexing. To improve the discriminating power, the grid sizes that are used for feature extraction can be enlarged and research is needed to other types of features that can be added. Finally, cluster algorithms can be used for efficient search strategies in very large databases.

After successful research in the proposed directions, fingerprint recognition systems will be less sensitive to low quality fingerprint images and will support identification in large databases. These achievements will enable much wider commercial application of fingerprint recognition.

Appendix A

Equivalence of DF Estimation Methods

We prove that the squared gradient method and the PCA-based method for the estimation of the DF are exactly equivalent. The proof starts by deriving the inverse of Equation 2.4, which was given to be:

$$\begin{bmatrix} G_{s,x} \\ G_{s,y} \end{bmatrix} = \begin{bmatrix} G_x^2 - G_y^2 \\ 2G_xG_y \end{bmatrix} \tag{A.1}$$

Substituting the lower part of this expression, which is given by

$$G_y = \frac{G_{s,y}}{2G_x} \tag{A.2}$$

into the upper part, given by

$$G_{s,x} = (G_x^2 - G_y^2) \tag{A.3}$$

gives

$$G_x^4 - G_{s,x}G_x^2 - \frac{1}{4}G_{s,y} = 0 \tag{A.4}$$

Solving this for G_x gives:

$$G_x = \begin{cases} \frac{1}{2}\sqrt{2G_{s,x} + 2\sqrt{G_{s,x}^2 + G_{s,y}^2}} \\ -\frac{1}{2}\sqrt{2G_{s,x} + 2\sqrt{G_{s,x}^2 + G_{s,y}^2}} \\ \frac{1}{2}\sqrt{2G_{s,x} - 2\sqrt{G_{s,x}^2 + G_{s,y}^2}} \\ -\frac{1}{2}\sqrt{2G_{s,x} - 2\sqrt{G_{s,x}^2 + G_{s,y}^2}} \end{cases} \tag{A.5}$$

The second and fourth solutions can be eliminated since G_x is always positive. Furthermore, since $\sqrt{G_{s,x}^2 + G_{s,y}^2} \geq G_{s,x}$, the third solution results in the square root of a negative number. Therefore, only the first solution is valid:

$$G_x = \frac{1}{2}\sqrt{2G_{s,x} + 2\sqrt{G_{s,x}^2 + G_{s,y}^2}} \tag{A.6}$$

The next step is to consider the squared gradients, averaged over the window W and to substitute, according to Equation 2.6:

$$\overline{G_{s,x}} = G_{xx} - G_{yy} \tag{A.7}$$

$$\overline{G_{s,y}} = 2G_{xy} \tag{A.8}$$

The average gradients, derived from the averaged squared gradients, are:

$$\begin{aligned} \overline{G_x} &= \frac{1}{2}\sqrt{2\overline{G_{s,x}} + 2\sqrt{\overline{G_{s,x}}^2 + \overline{G_{s,y}}^2}} \\ &= \sqrt{\frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}} \end{aligned} \tag{A.9}$$

and

$$\overline{G_y} = \frac{\overline{G_{s,y}}}{2\overline{G_x}} = \frac{G_{xy}}{\overline{G_x}} = \frac{G_{xy}}{\sqrt{\frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}}} \tag{A.10}$$

Now, it will be shown that the vector:

$$\begin{bmatrix} \overline{G_x} \\ \overline{G_y} \end{bmatrix} = \frac{1}{c} \cdot \begin{bmatrix} \frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2} \\ G_{xy} \end{bmatrix} \tag{A.11}$$

with:

$$c = \sqrt{\frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}} \tag{A.12}$$

is an eigenvector of autocovariance matrix **C**, which is defined in Equation 2.14. This will prove that both methods are equivalent. For the eigenvectors of **C**, the following expression must hold:

$$\mathbf{C} \cdot \mathbf{V} = \mathbf{V} \cdot \Lambda \tag{A.13}$$

where the columns of **V** are the eigenvectors of **C** and **Λ** is the diagonal matrix of the corresponding eigenvalues. This expression must also hold for one eigenvector $\mathbf{v}_1$ with corresponding eigenvalue λ_1:

$$\mathbf{C} \cdot \mathbf{v}_1 = \lambda_1 \cdot \mathbf{v}_1 \tag{A.14}$$

In order to show this, $[\overline{G_x}\ \overline{G_y}]^T$ is substituted for $\mathbf{v}_1$

$$\mathbf{v}_1 = \begin{bmatrix} \overline{G_x} \\ \overline{G_y} \end{bmatrix} \tag{A.15}$$

in the left-hand side of Equation A.14. This gives:

$$\begin{aligned} \mathbf{C} \cdot \mathbf{v}_1 &= \begin{bmatrix} G_{xx} & G_{xy} \\ G_{xy} & G_{yy} \end{bmatrix} \cdot \frac{1}{c} \cdot \begin{bmatrix} \frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2} \\ G_{xy} \end{bmatrix} \\ &= \frac{1}{c} \cdot \begin{bmatrix} G_{xx}\left(\frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}\right) + G_{xy}^2 \\ G_{xy}\left(\frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}\right) + G_{xy}G_{yy} \end{bmatrix} \end{aligned} \tag{A.16}$$

This must be equal to $\lambda_1 \cdot [\overline{G_x}, \overline{G_y}]^T$. Calculating λ_1 from the upper half of these expressions, we find:

$$\lambda_1 = \frac{G_{xx}\left(\frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}\right) + G_{xy}^2}{\frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}} \tag{A.17}$$

which, by multiplying numerator and denominator by $\frac{1}{2}(G_{xx} - G_{yy}) - \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}$, can be simplified to:

$$\lambda_1 = \frac{1}{2}(G_{xx} + G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2} \tag{A.18}$$

From the lower half of these expressions we find:

$$\lambda_1 = \frac{G_{xy}\left(\frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}\right) + G_{xy}G_{yy}}{G_{xy}} \tag{A.19}$$

which can be easily simplified to:

$$\lambda_1 = \frac{1}{2}(G_{xx} + G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2} \tag{A.20}$$

Since both expressions give the same result for λ_1, $[\overline{G_x}, \overline{G_y}]^T$ is an eigenvector of $\mathbf{C}$. Therefore, both methods are exactly equivalent.

It is not difficult to derive the second eigenvector $\mathbf{v}_2$ and its corresponding eigenvalue λ_2:

$$\mathbf{v}_1 = \begin{bmatrix} \frac{1}{2}(G_{xx} - G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2} \\ G_{xy} \end{bmatrix} \tag{A.21}$$

$$\mathbf{v}_2 = \begin{bmatrix} \frac{1}{2}(G_{xx} - G_{yy}) - \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2} \\ G_{xy} \end{bmatrix} \tag{A.22}$$

$$\lambda_1 = \frac{1}{2}(G_{xx} + G_{yy}) + \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2} \tag{A.23}$$

$$\lambda_2 = \frac{1}{2}(G_{xx} + G_{yy}) - \frac{1}{2}\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2} \tag{A.24}$$

Note that λ_1 is always larger than or equal to λ_2 confirming that the average gradient angle is aligned with $\mathbf{v}_1$. The DF, which is perpendicular to the gradient is aligned with $\mathbf{v}_2$.

Appendix B

Equivalence of *Coh* and *Str*

By substituting Equations A.23 and A.24, *Str* is given by:

$$\begin{aligned} Str &= \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \\ &= \frac{\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}}{G_{xx} + G_{yy}} \end{aligned} \tag{B.1}$$

On the other hand, *Coh* is given by:

$$Coh = \frac{\left|\sum_W (G_{s,x}, G_{s,y})\right|}{\sum_W \left|(G_{s,x}, G_{s,y})\right|} \tag{B.2}$$

where, by substituting Equation 2.4,

$$\begin{aligned} \left|\sum_W (G_{s,x}, G_{s,y})\right| &= \sqrt{\left(\sum_W G_{s,x}\right)^2 + \left(\sum_W G_{s,y}\right)^2} \\ &= \sqrt{\left(\sum_W G_x^2 - G_y^2\right)^2 + \left(\sum_W 2G_x G_y\right)^2} \\ &= \sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2} \end{aligned} \tag{B.3}$$

and

$$\begin{aligned}
\sum_W \left|(G_{s,x}, G_{s,y})\right| &= \sum_W \sqrt{G_{s,x}^2 + G_{s,y}^2} \\
&= \sum_W \sqrt{(G_x^2 - G_y^2)^2 + (2G_x G_y)^2} \\
&= \sum_W \sqrt{G_x^4 + 2G_x^2 G_y^2 + G_y^4} \qquad \text{(B.4)} \\
&= \sum_W \sqrt{(G_x^2 + G_y^2)^2} \\
&= \sum_W G_x^2 + G_y^2 \\
&= G_{xx} + G_{yy}
\end{aligned}$$

Therefore, the coherence of the averaging method is given by:

$$Coh = \frac{\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}}{G_{xx} + G_{yy}} \qquad \text{(B.5)}$$

which proves the equivalence of *Coh* and *Str*.

Appendix C

Rotation of Singular Points

It can be proved that

$$SDF_{\text{core},\varphi} = SDF_{\text{core,ref}} \cdot e^{j\varphi} \tag{C.1}$$

by using polar notation (ρ_s, ϕ_s) instead of (x, y) for a position in the reference model of the SPs. The orientation of the SDF is given by:

$$2\theta_{\text{core,ref}}(\rho_s, \phi_s) = \phi_s + \frac{1}{2}\pi \tag{C.2}$$

and the DF is given by:

$$\theta_{\text{core,ref}}(\rho_s, \phi_s) = \frac{1}{2}\phi_s + \frac{1}{4}\pi \tag{C.3}$$

The problem is to determine the SDF at position (ρ_s, ϕ_s) after rotation of the reference model over an angle φ. The sample point at (ρ_s, ϕ_s) after the rotation is located at $(\rho_s, \phi_s - \varphi)$ before the rotation:

$$\theta_{\text{core,ref}}(\rho_s, \phi_s - \varphi) = \frac{1}{2}(\phi_s - \varphi) + \frac{1}{4}\pi \tag{C.4}$$

The rotation adds φ to the orientation at the sample point:

$$\theta_{\text{core},\varphi}(\rho_s, \phi_s) = \frac{1}{2}(\phi_s - \varphi) + \frac{1}{4}\pi + \varphi \tag{C.5}$$

Now, the rotated DF can be converted back to the rotated SDF:

$$2\theta_{\text{core},\varphi}(\rho_s, \phi_s) = (\phi_s + \frac{1}{2}\pi) + \varphi = 2\theta_{\text{core,ref}}(\rho_s, \phi_s) + \varphi \tag{C.6}$$

which completes the proof. From the formula it becomes obvious that the SDF model of a core has to be rotated over 2π in order to obtain the original model.

Following the same procedure for a delta, it can be proved that

$$SDF_{\text{core},\varphi} = SDF_{\text{core,ref}} \cdot e^{j3\varphi} \tag{C.7}$$

Now, the orientation of the SDF is given by:

$$2\theta_{\text{delta,ref}}(\rho_s, \phi_s) = -\phi_s + \frac{1}{2}\pi \tag{C.8}$$

Following the same procedure as for the core gives:

$$\theta_{\text{delta},\varphi}(\rho_s, \phi_s) = -\frac{1}{2}(\phi_s - \varphi) + \frac{1}{4}\pi + \varphi \tag{C.9}$$

and:

$$2\theta_{\text{delta},\varphi}(\rho_s, \phi_s) = (-\phi_s + \frac{1}{2}\pi) + 3\varphi = 2\theta_{\text{delta,ref}}(\rho_s, \phi_s) + 3\varphi \tag{C.10}$$

This corresponds to the fact that a delta has to be rotated over $\frac{2}{3}\pi$ in order to obtain the original model.

Appendix D

Gabor Filtering

This appendix deals with Gabor filtering of images. A Gabor filter is a bandpass filter that enhances some spatial frequencies and attenuates the others. The real valued Gabor filter is defined by the point-wise multiplication of a cosine with a Gaussian window:

$$h_{\mathrm{Re}}(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \cos 2\pi f (x \sin\theta + y \cos\theta) \tag{D.1}$$

where θ is the orientation, f the spatial frequency and σ the standard deviation of the Gaussian window. The frequency response of the Gabor filter contains two Gaussian peaks at $(f\cos\theta,\ f\sin\theta)$ and $(-f\cos\theta, -f\sin\theta)$.

The complex Gabor filter is given by:

$$h_{\mathrm{Cx}}(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \exp\left(j 2\pi f (x \sin\theta + y \cos\theta)\right) \tag{D.2}$$

such that

$$\mathrm{Re}\left[h_{\mathrm{Cx}}(x, y)\right] = h_{\mathrm{Re}}(x, y) \tag{D.3}$$

Use of the complex Gabor filter offers two advantages. First, the complex Gabor filter can be applied much more efficiently than the real Gabor filter because it is separable into a part that is only dependent on x and a part that is only dependent on y.

$$h(x,y) = h_{\text{row}}(x) \cdot h_{\text{col}}(y) \tag{D.4}$$

$$h_{\text{row}}(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \exp\left(j2\pi f x \sin\theta\right) \tag{D.5}$$

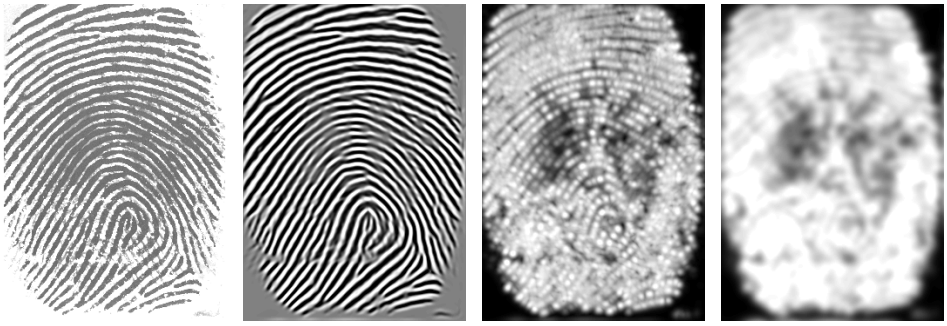
$$h_{\text{col}}(y) = \exp\left(-\frac{y^2}{2\sigma^2}\right) \exp\left(j2\pi f y \cos\theta\right) \tag{D.6}$$

Using this property, its response can be calculated by consecutive application of a row and a column filter, instead of a full matrix filter. Next, the response of the real filter can be obtained by taking the real part of the complex response.

For $\sigma = 3$ and and x and y ranging from -3σ to 3σ, the real Gabor filter requires $19 \cdot 19 = 361$ multiplications and additions per pixel, while the complex Gabor filter requires only $4 \cdot 19 = 76$ multiplications and additions per pixel. This is an improvement in computational complexity of more than a factor 4.

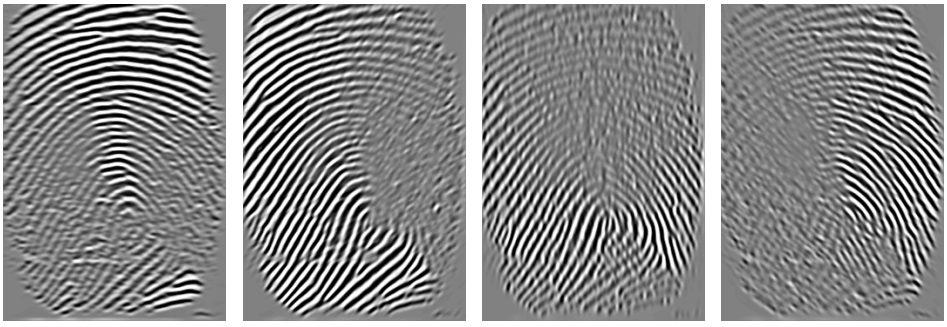
The second advantage of complex Gabor filtering is that not only the real part of the response can be used, but the imaginary part as well. Combining them to the absolute response is especially useful, since this provides a measure for the amplitude of the response. This measure can be used as an alternative to the local standard deviation of the real part of the response, which is for instance used in FingerCode [Jai99a, Jai00b] and for segmentation purposes [Jai97c]. Use of the absolute value requires a smaller smoothing filter and provides better indexing and matching performance.

Figure D.1 shows a fingerprint image, the accumulated real and absolute components of the Gabor response and the smoothed absolute component. Figure D.2 shows the real components of the fingerprint image that is filtered by Gabor filters with four different orientations, and Figure D.3 shows the corresponding absolute components. Finally, Figure D.4 shows the smoothed absolute values of the same fingerprint.



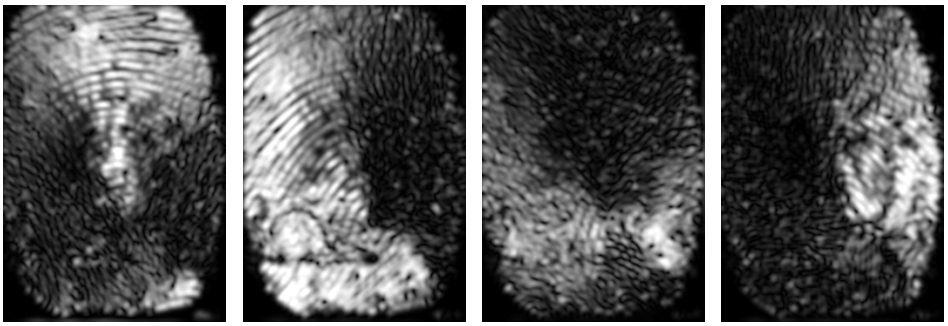
(a) Original (b) Real part (c) Absolute value (d) Smoothed absolute value

Figure D.1: Result of combined orientations by Gabor filtering.



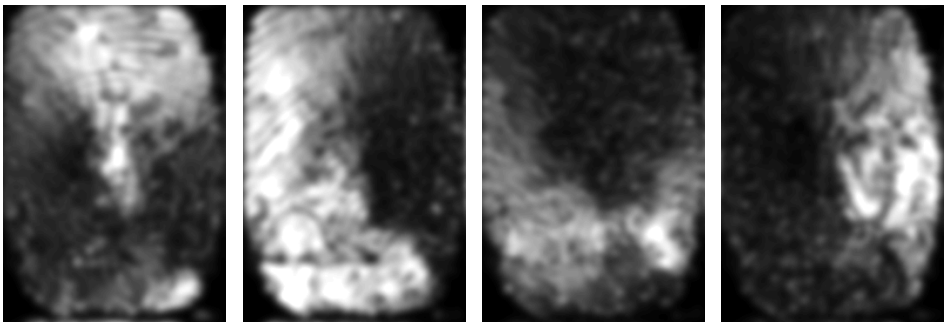
(a) 0π (b) $\pi/4$ (c) $\pi/2$ (d) $3\pi/4$

Figure D.2: Real part of fingerprint images that have been filtered by Gabor filters with different orientations.



(a) 0π (b) $\pi/4$ (c) $\pi/2$ (d) $3\pi/4$

Figure D.3: Absolute value of fingerprint images that have been filtered by Gabor filters with different orientations.



(a) 0π (b) $\pi/4$ (c) $\pi/2$ (d) $3\pi/4$

Figure D.4: Smoothed absolute value of fingerprint images that have been filtered by Gabor filters with different orientations.

Appendix E

Thin-Plate Splines

This appendix describes the estimation of thin-plate spline (TPS) model parameters. The TPS model provides a non-linear transformation, defined by 2 sets of corresponding landmark points that are extracted from 2 images. The TPS model describes the transformed coordinates x' and y' independently as a function of the original coordinates (x, y):

$$x' = f_x(x, y) \tag{E.1}$$
$$y' = f_y(x, y) \tag{E.2}$$

and gives a transformation for each point in the images.

Two different types of TPS parameter estimation methods can be distinguished. In [Boo89], interpolating thin-plate splines are presented, which find a transformation that exactly maps one set of landmark points on to the other set. On the other hand, in [Roh99], approximating thin-plate splines are presented. These splines define a transformation that does not exactly interpolate all given landmark points, but is allowed to approximate them in favor of a smoother transformation.

E.1 Interpolating Thin-Plate Splines

Given the displacements of a number of landmark points, the TPS transformation interpolates those points, while maintaining maximal smoothness. The smoothness is represented by the bending energy of a thin metal plate. At each landmark point (x, y), the displacement is represented by an additional z-coordinate, and, for each point, the thin metal plate is fixed at position (x, y, z). The bending energy J_b is given by the integral of the second order partial derivatives over the entire surface:

$$J_b = \iint \left(\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right) dx dy \tag{E.3}$$

The TPS model for one of the transformed coordinates (choose for instance x) is given by parameter vectors $\mathbf{a}$, $\mathbf{w}$ and P_i:

$$x' = f_x(x, y) = a_1 + a_2 x + a_3 y + \sum_{i=1}^{n} w_i U(|P_i - (x, y)|) \tag{E.4}$$

where $U(r) = r^2 \log r$ is the basis function, $\mathbf{a}$ defines the affine part of the transformation, $\mathbf{w}$ gives an additional non-linear deformation, and the P_i are the landmarks that the TPS interpolates.

For each coordinate, the bending energy of Eq. E.3 can be minimized by solving a set of linear equations:

$$\mathbf{K}\mathbf{w} + \mathbf{P}\mathbf{a} = \mathbf{v} \tag{E.5}$$

$$\mathbf{P}^T \mathbf{w} = 0 \tag{E.6}$$

where

$$\mathbf{w} = [w_x(1) \quad w_x(2) \quad \cdots \quad w_x(n)]^T \tag{E.7}$$

$$\mathbf{v} = [q_x(1) \quad q_x(2) \quad \cdots \quad q_x(n)]^T \tag{E.8}$$

$$\mathbf{a} = [a_x(1) \quad a_x(2) \quad a_x(3)]^T \tag{E.9}$$

$$\mathbf{P} = \begin{bmatrix} 1 & p_x(1) & p_y(1) \\ 1 & p_x(2) & p_y(2) \\ \vdots & \vdots & \vdots \\ 1 & p_x(n) & p_y(n) \end{bmatrix} \tag{E.10}$$

$$\mathbf{K} = \begin{bmatrix} U(P_1, P_1) & U(P_1, P_2) & \cdots & U(P_1, P_n) \\ U(P_2, P_1) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ U(P_n, P_1) & \cdots & \cdots & U(P_n, P_n) \end{bmatrix} \tag{E.11}$$

$P_i = (p_x(i), p_y(i))$ is the set of landmark points in the first image, $p_x(i)$ is the x-coordinate of point i in set P_i, $Q_i = (q_x(i), q_y(i))$ is the set of corresponding points in the second image, and n is the number of landmark points.

This set of equations can be solved very efficiently, as described in [Boo89]. Using block-matrices

$$\mathbf{L} = \left[\begin{array}{c|c} \mathbf{K} & \mathbf{P} \\ \hline \mathbf{P}^T & \mathbf{O} \end{array}\right] \tag{E.12}$$

$$\mathbf{Y} = \left[\mathbf{v}^T \mid 0 \quad 0 \quad 0\right]^T \tag{E.13}$$

where $\mathbf{O}$ is a 3×3 matrix of zeros, the TPS parameters are found by:

$$\left[\mathbf{w}^T \mid \mathbf{a}^T\right]^T = \mathbf{L}^{-1}\mathbf{Y} \tag{E.14}$$

Alternatively, the parameters for both x- and y-coordinates can be found simultaneously by:

$$\left[\begin{array}{c|c} \mathbf{w}_x^T & \mathbf{a}_x^T \\ \hline \mathbf{w}_y^T & \mathbf{a}_y^T \end{array}\right]^T = \mathbf{L}^{-1}\left[\begin{array}{c|ccc} \mathbf{v}_x^T & 0 & 0 & 0 \\ \hline \mathbf{v}_y^T & 0 & 0 & 0 \end{array}\right]^T \tag{E.15}$$

E.2 Approximating Thin-Plate Splines

In [Roh99], a method is presented to estimate *approximating thin-plate splines*. These splines do not exactly interpolate all given landmark points, but are allowed to approximate them in favor of a smoother transformation. The smoothness is controlled by a parameter λ, which provides a weight between the optimization of landmark distance and smoothness. For $\lambda = 0$, there is full interpolation, while for very large λ, there is only an affine transformation left, which is the smoothest transformation possible.

E.2.1 Anisotropic approximation

To represent anisotropic landmark errors, a covariance matrix Σ_i is constructed for each landmark, representing the distribution of the landmark errors. In this case, J_λ has to be minimized:

$$J_\lambda = J_a + \lambda \cdot J_b \tag{E.16}$$

where J_a is the approximation error, given by:

$$J_a = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{q}_i - f(\mathbf{p}_i))^T \Sigma_i^{-1}(\mathbf{q}_i - f(\mathbf{p}_i)) \tag{E.17}$$

To minimize this expression, a block-diagonal matrix $\mathbf{W}^{-1}$ is constructed from the covariance matrices Σ_i:

$$\mathbf{W}^{-1} = \mathrm{diag}\{\Sigma_1, \ldots, \Sigma_n\} \tag{E.18}$$

The TPS parameters are found by solving the system of equations

$$\begin{aligned} (\mathbf{K} + \lambda \mathbf{W}^{-1})\mathbf{w} + \mathbf{P}\mathbf{a} &= \mathbf{v} && \text{(E.19)} \\ \mathbf{P}^T \mathbf{w} &= 0 && \text{(E.20)} \end{aligned}$$

where in $\mathbf{K}$, each element K_{ij} is replaced with $K_{ij}\mathbf{I}_2$ with $\mathbf{I}_2$ the 2×2 identity matrix, each P_{ij} is replaced with $P_{ij}\mathbf{I}_2$, such that

$$\mathbf{K} = \left[\begin{array}{cc|c|cc} U(P_1,P_1) & 0 & \cdots & U(P_1,P_n) & 0 \\ 0 & U(P_1,P_1) & \cdots & 0 & U(P_1,P_n) \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline U(P_n,P_1) & 0 & \cdots & U(P_n,P_n) & 0 \\ 0 & U(P_n,P_1) & \cdots & 0 & U(P_n,P_n) \end{array}\right] \tag{E.21}$$

$$\mathbf{P} = \left[\begin{array}{cc|cc|cc} 1 & 0 & p_x(1) & 0 & p_y(1) & 0 \\ 0 & 1 & 0 & p_x(1) & 0 & p_y(1) \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline 1 & 0 & p_x(n) & 0 & p_y(n) & 0 \\ 0 & 1 & 0 & p_x(n) & 0 & p_y(n) \end{array}\right] \tag{E.22}$$

and $\mathbf{w}$, $\mathbf{v}$ and $\mathbf{a}$ are adjusted such that

$$\begin{aligned} \mathbf{w} &= [w_x(1) \quad w_y(1) \quad \cdots \quad w_x(n) \quad w_y(n)]^T && \text{(E.23)} \\ \mathbf{v} &= [q_x(1) \quad q_y(1) \quad \cdots \quad q_x(n) \quad q_y(n)]^T && \text{(E.24)} \\ \mathbf{a} &= [a_x(1) \quad a_y(1) \quad \cdots \quad a_x(n) \quad a_y(n)]^T && \text{(E.25)} \end{aligned}$$

E.2.2 Isotropic approximation

Alternatively, for equal isotropic errors at all landmarks, only $\mathbf{K}$ has to be adjusted, and a slightly changed version of Eq. E.15, in which $\mathbf{K}$ is replaced by $\mathbf{K} + \lambda\mathbf{I}$, can be used to solve the following system of equations

$$\begin{aligned} (\mathbf{K} + \lambda \mathbf{I})\mathbf{w} + \mathbf{P}\mathbf{a} &= \mathbf{v} && \text{(E.26)} \\ \mathbf{P}^T \mathbf{w} &= 0 && \text{(E.27)} \end{aligned}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix.

Appendix F

Gaussian Approximation of Error Rates

Assume that the different observed feature vectors $\mathbf{v}$ that are generated by class w_k, have a multi-dimensional Gaussian distribution with dimension d, mean $\mathbf{c}_k$ and covariance matrix Σ_E, independent of the class. Then, the conditional probability density $p(\mathbf{v}|w_k)$ is given by

$$\begin{aligned} p(\mathbf{v}|w_k) &= n(\mathbf{c}_k, \Sigma_E) && \text{(F.1)} \\ &= \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_E|^{1/2}} \cdot \exp\left(-\frac{1}{2}(\mathbf{v}-\mathbf{c}_k)^T \Sigma_E^{-1}(\mathbf{v}-\mathbf{c}_k)\right) && \text{(F.2)} \end{aligned}$$

Furthermore, assume that the centers $\mathbf{c}_k$ of classes w_k have a Gaussian distribution with zero mean (which can be guaranteed by subtraction of the mean) and covariance matrix Σ_M. Then, $p(\mathbf{c}_k)$ is given by:

$$\begin{aligned} p(\mathbf{c}_k) &= n(0, \Sigma_M) && \text{(F.3)} \\ &= \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_M|^{1/2}} \cdot \exp\left(-\frac{1}{2}\mathbf{c}_k^T \Sigma_M^{-1}\mathbf{c}_k\right) && \text{(F.4)} \end{aligned}$$

The prior probability density function $p(\mathbf{v})$ of all patterns is given by:

$$\begin{aligned} p(\mathbf{v}) &= \int_W p(\mathbf{v}|w_k)p(w_k)\, dw_k \\ &= \int_{\mathbf{V}} n(\mathbf{c}_k, \Sigma_E) \cdot n(0, \Sigma_E)\, d\mathbf{c}_k && \text{(F.5)} \end{aligned}$$

Under the condition that the distribution over the entire population is much wider than the distribution within one class,

$$|\Sigma_E| \ll |\Sigma_M| \tag{F.6}$$

$p(\mathbf{v})$ is equal to $p(\mathbf{c}_k)$, given by:

$$\begin{aligned} p(\mathbf{v}) &= n(0, \Sigma_M) && \text{(F.7)} \\ &= \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_M|^{1/2}} \cdot \exp\left(-\frac{1}{2}\mathbf{v}^T \Sigma_M^{-1}\mathbf{v}\right) && \text{(F.8)} \end{aligned}$$

Using the acceptance condition $L(\mathbf{v}) \geq t$, and Expression 7.4 for $L(\mathbf{v})$, an expression for the acceptance region $A_{k,t}$ can be derived:

$$p(\mathbf{v}|w_k) \geq t \cdot p(\mathbf{v}) \tag{F.9}$$

By substituting the Gaussian probability density functions F.2 and F.8, we obtain:

$$\begin{aligned} &\frac{1}{(2\pi)^{d/2} \cdot |\Sigma_E|^{1/2}} \cdot \exp\left(-\frac{1}{2}(\mathbf{v}-\mathbf{c}_k)^T \Sigma_E^{-1}(\mathbf{v}-\mathbf{c}_k)\right) \\ &\qquad \geq t \cdot \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_M|^{1/2}} \cdot \exp\left(-\frac{1}{2}\mathbf{v}^T \Sigma_M^{-1}\mathbf{v}\right) \end{aligned} \tag{F.10}$$

or

$$\exp\left(-\frac{1}{2}(\mathbf{v}-\mathbf{c}_k)^T \Sigma_E^{-1}(\mathbf{v}-\mathbf{c}_k) + \frac{1}{2}\mathbf{v}^T \Sigma_M^{-1}\mathbf{v}\right) \geq t \cdot \left(\frac{|\Sigma_E|}{|\Sigma_M|}\right)^{1/2} \tag{F.11}$$

By taking the natural logarithm and multiplying both sides by -2, we obtain:

$$(\mathbf{v}-\mathbf{c}_k)^T \Sigma_E^{-1}(\mathbf{v}-\mathbf{c}_k) - \mathbf{v}^T \Sigma_M^{-1}\mathbf{v} \leq -\log\left(t^2 \cdot \frac{|\Sigma_E|}{|\Sigma_M|}\right) \tag{F.12}$$

and the acceptance region is given by:

$$A_{k,t} = \{\mathbf{v} \in \mathbf{V} \mid (\mathbf{v}-\mathbf{c}_k)^T \Sigma_E^{-1}(\mathbf{v}-\mathbf{c}_k) \leq \beta^2(t)\} \tag{F.13}$$

with

$$\beta^2(t) = -\log\left(t^2 \cdot \frac{|\Sigma_E|}{|\Sigma_M|}\right) + \mathbf{v}^T \Sigma_M^{-1} \mathbf{v} \tag{F.14}$$

For threshold values that correspond to a relatively high likelihood ratio, $\mathbf{v}$ is relatively close to $\mathbf{c}_k$, and $\mathbf{v}$ can be replaced with $\mathbf{c}_k$ in Expression F.14. Then, β is independent of $\mathbf{v}$, and $A_{k,t}$ is an ellipsoid region with a Mahalanobis distance less than β from the class center $\mathbf{c}_k$.

Now, the error rates can be calculated by substituting Expression F.13 into 7.8 and 7.10. To transform the multidimensional integrals into one-dimensional integrals, we use the volume $Vol(r, \Sigma)$ of a ellipsoid that is defined by covariance matrix Σ and Mahalanobis distance r, which is given by [Gol97]:

$$Vol(r, \Sigma) = V_{\text{unit}} \cdot |\Sigma|^{1/2} \cdot r^d \tag{F.15}$$

with V_{unit} a constant that depends on the dimension d of the space

$$V_{\text{unit}} = \begin{cases} \dfrac{\pi^{1/2}}{(d/2)!} & d \text{ even} \\ \dfrac{2^d \pi^{(d-1)/2}(d-1)/2}{d!} & d \text{ odd} \end{cases} \tag{F.16}$$

Using

$$d\ Vol(r, \Sigma) = V_{\text{unit}} \cdot |\Sigma|^{1/2} \cdot d \cdot s^{d-1}\, ds \tag{F.17}$$

and Expression 7.8, the false rejection rate is given by:

$$\begin{aligned} FRR_k(t) &= 1 - \int_0^{\beta(t)} p(\mathbf{v}|w_k) d\ Vol(r, \Sigma) \\ &= 1 - \int_0^{\beta(t)} \frac{1}{(2\pi)^{d/2}\,|\Sigma_E|^{1/2}} \exp(-s^2/2) V_{\text{unit}} |\Sigma_E|^{1/2} d s^{d-1}\, ds \\ &= 1 - \frac{V_{\text{unit}} \cdot d}{(2\pi)^{d/2}} \cdot \int_0^{\beta(t)} \exp(-s^2/2) \cdot s^{d-1} ds \end{aligned} \tag{F.18}$$

which corresponds to a χ^2 distribution with d degrees of freedom.

Using Expression 7.10 and assuming $p(\mathbf{v})$ constant within class w_k, the false acceptance rate is given by:

$$FAR_k(t) = p(\mathbf{v}) \cdot V_{\text{unit}} \cdot |\Sigma_E|^{1/2} \cdot \beta^d(t) \tag{F.19}$$

These expressions for the theoretic error rates can be calculated relatively easily. Experiments show that they provide a very accurate estimate of the error rates for low dimensions ($d < 10$), while the experimental results deviate from the predicted values for higher dimensions. This can be explained by the fact that $p(\mathbf{v})$ cannot be assumed constant within class w_k anymore.

References

[Act01] S.T. Actor, D.P. Mukherjee, J.P. Havlicek, and A.C. Bovik. Oriented Texture Completion by AM-FM Reaction-Diffusion. *IEEE Trans. Image Processing*, 10(6):885–896, June 2001.

[Ara02] J.L. Araque, M. Baena, and P.R. Vizcaya. Synthesis of Fingerprint Images. In *Proc. ICPR 2002*, Quebec City, Canada, August 2002.

[Ash00] J. Ashbourn. *Biometrics: Advanced Identity Verification*. Springer-Verlag, London, 2000.

[Baz00a] A.M. Bazen and S.H. Gerez. Directional Field Computation for Fingerprints Based on the Principal Component Analysis of Local Gradients. In *Proc. ProRISC2000, 11th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2000.

[Baz00b] A.M. Bazen, G.T.B. Verwaaijen, L.P.J. Veelenturf, B.J. van der Zwaag, and S.H. Gerez. A Correlation-Based Fingerprint Verification System. In *Proc. ProRISC2000, 11th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2000.

[Baz01a] A.M. Bazen and S.H. Gerez. The Construction of an Intrinsic Coordinate System for Fingerprint Matching. In *Proc. ASCI Conference 2001*, pages 337–344, May 2001.

[Baz01b] A.M. Bazen and S.H. Gerez. Extraction of Singular Points from Directional Fields of Fingerprints. In *Mobile Communications in Perspective, Proc. CTIT Workshop on Mobile Communications*, pages 41–44, University of Twente, Enschede, The Netherlands, February 2001.

[Baz01c] A.M. Bazen and S.H. Gerez. An Intrinsic Coordinate System for Fingerprint Matching. In *Proc. 3rd Int. Conf. Audio- and Video-Based Biometric Person Authentication (AVBPA 2001)*, pages 198–204, June 2001.

[Baz01d] A.M. Bazen and S.H. Gerez. Segmentation of Fingerprint Images. In *Proc. ProRISC2001, 12th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2001.

[Baz01e] A.M. Bazen, M. van Otterlo, S.H. Gerez, and M. Poel. A Reinforcement Learning Agent for Minutiae Extraction from Fingerprints. In *Proc. BNAIC 2001*, pages 329–336, Amsterdam, October 2001.

[Baz02a] A.M. Bazen and S.H. Gerez. Achievements and Challenges in Fingerprint Recognition. In D. Zhang, editor, *Biometric Solutions for Authentication in an e-World*, pages 23–57. Kluwer, 2002.

[Baz02b] A.M. Bazen and S.H. Gerez. Elastic Minutiae Matching by means of Thin-Plate Spline Models. In *Proc. ICPR 2002*, Quebec City, August 2002.

[Baz02c] A.M. Bazen and S.H. Gerez. Fingerprint Matching by Thin-Plate Spline Modelling of Elastic Deformations. *Pattern Recognition*, 2002. Submitted.

[Baz02d] A.M. Bazen and S.H. Gerez. Systematic Methods for the Computation of the Directional Field and Singular Points of Fingerprints. *IEEE Trans. PAMI*, 24(7):905–919, July 2002.

[Baz02e] A.M. Bazen and S.H. Gerez. Thin-Plate Spline Modelling of Elastic Deformations in Fingerprints. In *Proc. SPS 2002*, pages 205–208, Leuven, Belgium, March 2002.

[Baz02f] A.M. Bazen and R.N.J. Veldhuis. Likelihood Ratio-Based Biometric Verification. In *Proc. ProRISC 2002, 13th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2002.

[Bha01] B. Bhanu and X. Tan. A Triplet Based Approach for Indexing of Fingerprint Database for Identification. In J. Bigun and F. Smeraldi, editors, *Proc. 3rd Int. Conf. Audio- and Video-Based Biometric Person Authentication (AVBPA 2001)*, pages 205–210, Halmstad, Sweden, June 2001.

[Big91] J. Bigun, G.H. Granlund, and J. Wiklund. Multidimensional Orientation Estimation with Applications to Texture Analysis and Optical Flow. *IEEE Trans. PAMI*, 13(8):775–790, August 1991.

[Boe01] J. de Boer, A.M. Bazen, and S.H. Gerez. Indexing Fingerprint Databases Based on Multiple Features. In *Proc. ProRISC2001, 12th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2001.

[Boo89] F.L. Bookstein. Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. *IEEE Trans. PAMI*, 11(6):567–585, June 1989.

[Can95] 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. Technical Report NISTIR 5647, NIST, April 1995.

[Cap99] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni. Fingerprint Classification by Directional Image Partitioning. *IEEE Trans. PAMI*, 21(5):402–421, May 1999.

[Cap00a] R. Cappelli, A. Erol, D. Maio, and D. Maltoni. Synthetic Fingerprint-Image Generation. In *Proc. ICPR2000, 15th Int. Conf. Pattern Recognition*, Barcelona, Spain, September 2000.

[Cap00b] R. Cappelli, D. Maio, and D. Maltoni. Combining Fingerprint Classifiers. In *Proc. First International Workshop on Multiple Classifier Systems (MCS2000)*, pages 351–361, Cagliari, June 2000.

[Cap00c] R. Cappelli, D. Maio, and D. Maltoni. Indexing Fingerprint Databases for Efficient 1:N Matching. In *Proc. Sixth Int. Conf. on Control, Automation, Robotics and Vision (ICARCV2000)*, Singapore, December 2000.

[Cap01] R. Cappelli, D. Maio, and D. Maltoni. Modelling Plastic Distortion in Fingerprint Images. In *Proc. ICAPR2001, Second Int. Conf. Advances in Pattern Recognition*, Rio de Janeiro, March 2001.

[Cho97] M.M.S. Chong, T.H. Ngee, and R.K.L. Gay. Geometric Framework for Fingerprint Image Classification. *Pattern Recognition*, 30(9):1475–1488, 1997.

[Chu00] H. Chui and A. Rangarajan. A New Algorithm for Non-Rigid Point Matching. In *Proc. CVPR*, volume 2, pages 40–51, June 2000.

[Dre99] G.A. Drets and H.G. Liljenström. Fingerprint Subclassification: A Neural Network Approach. In L.C. Jain, U. Halici, I. Hayashi, S.B. Lee, and S. Tsutsui, editors, *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, pages 109–134. CRC Press, Boca Raton, 1999.

[Far99] A. Farina, Z.M. Kovács-Vajna, and A. Leone. Fingerprint minutiae extraction from skeletonized binary images. *Pattern Recognition*, 32(5):877–889, 1999.

[Ger97] R.S. Germain, A. Califano, and S. Colville. Fingerprint Matching Using Transformation Parameter Clustering. *IEEE Computational Science and Engineering*, 4(4):42–49, 1997.

[Gol97] M. Golfarelli, D. Maio, and D. Maltoni. On the error-reject trade-off in biometric verification systems. *IEEE Trans. PAMI*, 19(7):786–796, July 1997.

[Hay99] S. Haykin. *Neural Networks, A Comprehensive Foundation*. Prentice Hall International, Inc., Upper Saddle River, NJ, 1999.

[Hen00] E.R. Henry. *Classification and Uses of Finger Prints*. Routledge, London, 1900.

[Ho94] T.K. Ho, J.J. Hull, and S.N. Srihari. Decision Combination in Multiple Classifier Systems. *IEEE Trans. PAMI*, 16(1):66–75, January 1994.

[Hon98] L. Hong, Y. Wan, and A. Jain. Fingerprint image enhancement: Algorithm and performance evaluation. *IEEE Trans. PAMI*, 20(8):777–789, August 1998.

[Hon99] L. Hong and A.K. Jain. Classification of Fingerprint Images. In *Proc. 11th Scandinavian Conference on Image Analysis*, Kangerlussuaq, Greenland, June 1999.

[Hun93] D.C.D. Hung. Enhancement and Feature Purification of Fingerprint Images. *Pattern Recognition*, 26(11):1661–171, 1993.

[Jai89] A.K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[Jai97a] A.K. Jain, L. Hong, and R. Bolle. On-line fingerprint verification. *IEEE Trans. PAMI*, 19(4):302–314, April 1997.

[Jai97b] A.K. Jain, L. Hong, S. Pankanti, and R. Bolle. An Identity-Authentication System Using Fingerprints. *Proc. of the IEEE*, 85(9):1365–1388, September 1997.

[Jai97c] A.K. Jain and N.K. Ratha. Object detection using Gabor filters. *Pattern Recognition*, 30(2):295–309, February 1997.

[Jai99a] A. K. Jain, S. Prabhakar, and L. Hong. A Multichannel Approach to Fingerprint Classification. *IEEE Trans. PAMI*, 21(4):348–359, April 1999.

[Jai99b] A.K. Jain, R. Bolle, and S. Pankanti. *Biometrics - Personal Identification in a Networked Society*. Kluwer Academic Publishers, Boston, 1999.

[Jai00a] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Trans. PAMI*, 22(1), January 2000.

[Jai00b] A.K. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Filterbank-Based Fingerprint Matching. *IEEE Trans. Image Processing*, 9(5):846–859, May 2000.

[Jai01] A.K. Jain, A. Ross, and S. Prabhakar. Fingerprint Matching Using Minutiae and Texture Features. In *Proc. Int. Conf. on Image Processing (ICIP)*, Greece, October 2001.

[Jai02] A.K. Jain and A. Ross. Fingerprint Mosaicking. In *Proc. Int. Conf. on Acoustic Speech and Signal Processing (ICASSP)*, Orlando, Florida, May 2002.

[Jia00] X. Jiang and W.Y. Yau. Fingerprint Minutiae Matching Based on the Local and Global Structures. In *Proc. ICPR2000, 15th Int. Conf. Pattern Recognition*, volume 2, pages 1042–1045, Barcelona, Spain, September 2000.

[Jia01] X. Jiang, W.Y. Yau, and W. Ser. Detecting the fingerprint minutiae by adaptive tracing the gray-level ridge. *Pattern Recognition*, 34(5):999–1013, May 2001.

[Kae96] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Kar96] K. Karu and A.K. Jain. Fingerprint classification. *Pattern Recognition*, 29(3):389–404, 1996.

[Kas87] M. Kass and A. Witkin. Analyzing Oriented Patterns. *Computer Vision, Graphics, and Image Processing*, 37(3):362–385, March 1987.

[Kaw84] M. Kawagoe and A. Tojo. Fingerprint Pattern Classification. *Pattern Recognition*, 17(3):295–303, 1984.

[Kim01] S. Kim, D. Lee, and J. Kim. Algorithm for Detection and Elimination of False Minutiae in Fingerprint Images. In J. Bigun and F. Smeraldi, editors, *Proc. AVBPA*, volume 2091 of *LNCS*, pages 235–240, Halmstad, Sweden, June 2001. Springer.

[Kit98] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. PAMI*, 20(3):226–239, March 1998.

[Kle02] S. Klein, A.M. Bazen, and R.N.J. Veldhuis. Fingerprint Image Segmentation Based on Hidden Markov Models. In *Proc. ProRISC 2002, 13th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2002.

[Köp99] M. Köppen and B. Nickolay. Design of Image Exploring Agent Using Genetic Programming. *Fuzzy Sets and Systems*, 103(2):303–315, April 1999.

[Kov00] Z.M. Kovács-Vajna. A Fingerprint Verification System Based on Triangular Matching and Dynamic Time Warping. *IEEE Trans. PAMI*, 22(11):1266–1276, November 2000.

[Koz92] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[Koz94] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge Massachusetts, May 1994.

[Koz99] John R. Koza, David Andre, Forrest H. Bennett III, and Martin Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufman, April 1999.

[Kum01] S. Kumar, M. Sallam, and D. Goldgof. Matching Point Features under Small Nonrigid Motion. *Pattern Recognition*, 34(12):2353–2365, December 2001.

[Li00] J. Li, A. Najmi, and R.M. Gray. Image Classification by a Two-Dimensional Hidden Markov Model. *IEEE Trans. Signal Proc.*, 48(2):517–533, February 2000.

[Lin92] L.J. Lin. Self-improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching. *Machine Learning Journal*, 8(3/4), 1992. Special Issue on Reinforcement Learning.

[Lin94] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Boston, 1994.

[Lum97] A. Lumini, D. Maio, and D. Maltoni. Continuous versus exclusive classification for fingerprint retrieval. *Pattern Recognition Letters*, 18(10):1027–1034, 1997.

[Mai97] D. Maio and D. Maltoni. Direct Gray-Scale Minutiae Detection in Fingerprints. *IEEE Trans. PAMI*, 19(1):27–39, January 1997.

[Mai99] D. Maio and D. Maltoni. Minutiae Extraction and Filtering from Gray-Scale Images. In L.C. Jain et al., editor, *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, pages 155–192. CRC Press LLC, 1999.

[Mai00] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, and A.K. Jain. FVC2000: Fingerprint Verification Competition. Biolab internal report, University of Bologna, Italy, September 2000. available from `http://bias.csr.unibo.it/fvc2000/`.

[Mai02] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, and A.K. Jain. FVC2000: Fingerprint Verification Competition. *IEEE Trans. PAMI*, 24(3):402–412, March 2002.

[Meh87] B.M. Mehtre, N.N. Murthy, S. Kapoor, and B. Chatterjee. Segmentation of Fingerprint Images Using the Directional Image. *Pattern Recognition*, 20(4):429–435, 1987.

[Meh89] B.M. Mehtre and B. Chatterjee. Segmentation of Fingerprint Images - a Composite Method. *Pattern Recognition*, 22(4):381–385, 1989.

[Meu00] P.G.M. van der Meulen, A.M. Bazen, and S.H. Gerez. A Distributed Object-Oriented Environment for the Application of Genetic Programming to Signal Processing. In *Proc. ProRISC 2000*, Veldhoven, The Netherlands, November 2000.

[Meu01] P.G.M. van der Meulen, H. Schipper, A.M. Bazen, and S.H. Gerez. PMDGP: A Distributed Object-Oriented Genetic Programming Environment. In *Proc. ASCI Conference 2001*, pages 484–491, May 2001.

[Moo00] T.K. Moon and W.C. Stirling. *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 2000.

[Nak82] O. Nakamura, K. Goto, and T. Minami. Fingerprint classification by directional distribution patterns. *Systems, Computers, Controls*, 13(5):81–89, 1982.

[O'G89] L. O'Gorman and J.V. Nickerson. An Approach to Fingerprint Filter Design. *Pattern Recognition*, 22(1):29–38, 1989.

[Pal93] N.R. Pal and S.K. Pal. A Review on Image Segmentation Techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.

[Pan01] S. Pankanti, S. Prabhakar, and A. K. Jain. On the Individuality of Fingerprints. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, Hawaii, December 2001.

[Pat01] M.S. Pattichis, G. Panayi, A.C. Bovik, and S.P. Hsu. Fingerprint Classification Using and AM-FM Model. *IEEE Trans. Image Processing*, 10(6):951–954, June 2001.

[Per98] P. Perona. Orientation diffusions. *IEEE Trans. Image Processing*, 7(3):457–467, March 1998.

[Pra00] S. Prabhakar, A.K. Jain, J. Wang, S. Pankanti, and R. Bolle. Minutia Verification and Classification for Fingerprint Matching. In *Proc. ICPR2000, 15th Int. Conf. Pattern Recognition*, Barcelona, Spain, September 2000.

[Pra01] S. Prabhakar. *Fingerprint Classification adn Matching Using a Filterbank*. PhD thesis, Michigan State University, 2001.

[Pro92] J.G. Proakis, C.M. Rader, F. Ling, and C.L. Nikias. *Advanced Digital Signal Processing*. Macmillan Publishing Company, NY, 1992.

[Rab89] L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE*, 77(2):257–286, February 1989.

[Rab93] L. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ, 1993.

[Rao92] A.R. Rao and R.C. Jain. Computerized Flow Field Analysis: Oriented Texture Fields. *IEEE Trans. PAMI*, 14(7):693–709, July 1992.

[Rat95] N. Ratha, S. Chen, and A. Jain. Adaptive flow orientation based feature extraction in fingerprint images. *Pattern Recognition*, 28:1657–1672, Nov. 1995.

[Rat98] N.K. Ratha, J.H. Connell, and R.M. Bolle. Image mosaicing for rolled fingerprint construction. In *Proc. 14th ICPR*, pages 1651–1653, vol. 2, 1998.

[Rat00] N.K. Ratha, R.M. Bolle, V.D. Pandit, and V. Vaish. Robust fingerprint authentication using local structural similarity. In *Proc. 5th IEEE Workshop Appl. Comp. Vision*, pages 29–34, 2000.

[Rod99] A.R. Roddy and J.D. Stosz. Fingerprint Feature Processing Techniques and Poroscopy. In L.C. Jain, U. Halici, I. Hayashi, S.B. Lee, and S. Tsutsui, editors, *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, pages 37–105. CRC Press, 1999.

[Roh99] K. Rohr, M. Fornefett, and H.S. Stiehl. Approximating Thin-Plate Splines for Elastic Registration: Integration of Landmark Errors and Orientation Attributes. In *Proc. 16th Int. Conf. Information Processing in Medical Imaging*, LNCS 1613, pages 252–265, Hungary, June 1999.

[Ros02a] A. Ross, A.K. Jain, and J. Reisman. A Hybrid Fingerprint Matcher. In *Proc. ICPR 2002*, Quebec City, Canada, August 2002.

[Ros02b] A. Ross, J. Reisman, and A.K. Jain. Fingerprint Matching Using Feature Space Correlation. In M. Tistarelli, J. Bigun, and A.K. Jain, editors, *Biometric Authentication, Int. ECCV 2002 Workshop*, volume 2359 of *LNCS*, pages 48–57, Copenhagen, Denmark, June 2002. Springer-Verlag.

[Rum94] G.A. Rummery and M. Niranjan. On-Line Q-Learning using Connectionist Systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University, Engineering Department, 1994.

[Sch00] M. Schrijver, A.M. Bazen, and C.H. Slump. On the Reliability of Template Matching in Biomedical Image Processing. In *Proc. SPS2000, IEEE Benelux Signal Processing Chapter*, Hilvarenbeek, The Netherlands, March 2000.

[Sen01] A.W. Senior and R. Bolle. Improved Fingerprint Matching by Distortion Removal. *IEICE Trans. Inf. and Syst., Special issue on Biometrics*, E84-D(7):825–831, July 2001.

[She93] B.G. Sherlock and D.M. Monro. A Model for Interpreting Fingerprint Topology. *Pattern Recognition*, 26(7):1047–1055, 1993.

[She94] B.G. Sherlock, D.M. Monro, and K. Millard. Fingerprint Enhancement by Directional Fourier Filtering. *IEE Proc.-Vis. Image Signal Process.*, 141(2):87–94, April 1994.

[Sri92] V.S. Srinivasan and N.N. Murthy. Detection of Singular Points in Fingerprint Images. *Pattern Recognition*, 25(2):139–153, 1992.

[Sue00] C.Y. Suen and L. Lam. Multiple Classifier Combination Methodologies for Different Output Levels. In J. Kittler and F. Roli, editors, *Proc. MCS 2000*, pages 52–56, 2000.

[Sut98] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Camebridge, Massachusetts, 1998.

[Sut00] R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.

[The92] C.W. Therrien. *Discrete Random Signals and Statistical Signal Processing*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 1992.

[Tre68] H.L. Van Trees. *Detection, estimation, and modulation theory*. Wiley, New York, 1968.

[Tur91] M. Turk and A. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[Vel90] R.N.J. Veldhuis. *Restoration of Lost Samples in Digital Signals*. Prentice Hall International, UK, 1990.

[Ver00] G.T.B. Verwaaijen. Fingerprint Authentication. Master's thesis, Faculty of Electrical Engineering, University of Twente, Enschede, The Netherlands, May 2000. EL-S&S-002N00.

[Viz96] P. Vizcaya and L. Gerhardt. A Nonlinear Orientation Model for Global Description of Fingerprints. *Pattern Recognition*, 29(7):1221–1231, 1996.

[Way99] J.L. Wayman. Error rate equations for the general biometric system. *IEEE Robotics and Automation Magazine*, 6(1):35–48, March 1999.

[Wil94] C.L. Wilson, G.T. Candela, and C.I. Watson. Neural Network Fingerprint Classification. *J. Artificial Neural Networks*, 1(2):203–228, 1994.

[Wil01] A.J. Willis and L. Myers. A cost-effective fingerprint recognition system for use with low-quality prints and damaged fingertips. *Pattern Recognition*, 34(2):255–270, 2001.

[Xia91] Q. Xiao and H. Raafat. Fingerprint Image Postprocessing: a Combined Statistical and Structural Approach. *Pattern Recognition*, 24(10):985–992, 1991.

[Zha02] D. Zhang. *Biometric Solutions for Authentication in an e-World*. Kluwer Academic Publishers, 2002. To be published.

[Zho01] J. Zhou, D. He, G. Rong, and Z.Q. Bian. Effective Algorithm for Rolled Fingerprint Construction. *Electronics Letters*, 37(8):492–494, April 2001.

Summary

Recognition of persons on the basis of biometric features is an emerging phenomenon in our society. The use of features physically connected to a person's body significantly decreases the possibility of fraud. Furthermore biometry can offer user-convenience in many situations, as it replaces cards, keys, and codes. The fingerprint is considered one of the most practical features, since it is user friendly, provides good performance, and uses sensors that are relatively inexpensive and that can be integrated easily in wireless hardware.

A fingerprint is a pattern of curving line structures, where the skin has a higher profile than its surroundings. These structures are called ridges and valleys. Commonly used features that are extracted from the fingerprint image are the directional field, the singular points and the minutiae. The directional field is defined as the local orientation of the ridge-valley structures. It describes the coarse structure, or basic shape, of a fingerprint. The singular points are the discontinuities in the directional field. A fingerprint contains at most four singular points, which can be used for registration purposes. The minutiae provide the details of the ridge-valley structures. Two elementary types of minutiae exist: ridge endings and bifurcations. A typical fingerprint contains 20 to 50 minutiae, which can be used for matching purposes.

Two types of fingerprint recognition systems can be distinguished, being verification and identification systems. Verification systems use fingerprint technology to verify the claimed identity of a user. The user offers his identity and a test fingerprint to the system, and the test fingerprint is matched against a reference fingerprint that is retrieved from a database. The verification decision is based on the resulting measure of similarity, or matching score. Identification systems, on the other hand, require only a query fingerprint as input. The person is identified by searching a database for a matching fingerprint.

The distributions of the matching scores of genuine and impostor attempts overlap to some extent. This results in two kinds of errors, that together measure the performance of a fingerprint recognition system. The false acceptance rate is the probability that the system outputs *'match'* for fingerprints that are not from the same finger. The false rejection rate is the probability that the system outputs *'non-match'* for fingerprints that originate from the same finger.

This thesis focusses on three principal challenges in fingerprint recognition: robust feature extraction from low-quality fingerprints, matching elastically deformed fingerprints, and efficient search of fingerprints in a database. Research into these topics has yielded improvements for most of these challenges, resulting in a system that provides increased fingerprint recognition performance.

A fingerprint recognition system involves several phases. In the acquisition phase, the fingerprint is scanned using a fingerprint sensor that is in general based on optical or capacitive principles. This results is a digital gray-scale image that is processed in the subsequent phases. The quality and characteristics of the fingerprint image are highly dependent on the exact type of sensor that is used. Therefore, the choice of the sensor directly affects the recognition performance, and the recognition algorithms have to be adapted to the specific fingerprint sensor that is used.

The feature extraction phase involves the calculation of various characteristics of the fingerprint that are used for matching and database search. In this work, new methods are proposed for accurate estimation of the directional field, for the consistent extraction of singular points, and estimation of their orientation, for more accurate segmentation of fingerprint images, for the efficient enhancement of fingerprint images, and for the extraction of the minutiae.

In the matching phase, the features of the test fingerprint are compared to a template that is retrieved from a database. This thesis presents the first elastic minutiae matching algorithm that explicitly deals with elastically deformed fingerprints. Additionally, a very efficient matching algorithm is proposed that uses fixed length feature vectors instead of minutiae sets.

Identification systems have to compare the query fingerprint to all fingerprints in their database. However, the computational time and the probability of false acceptance increase with larger numbers of matches. To reduce the number of matches that have to be performed, identification systems use some form of classification. After classifying the query fingerprints and all fingerprints in the database, the query fingerprint is only matched against the fingerprints of the corresponding class. In this work, an alternative database indexing method is used that considers each fingerprint as a distinct class. Candidate fingerprints are selected in decreasing order of probability from the database, for matching against the query fingerprint. By combining multiple types of features, this method is able to reduce the effective size of the database to be searched to less than 0.5% of its original size. This enables the use of databases that are 100 times larger than in the situation without indexing.

Samenvatting

De herkenning van personen op basis van biometrische kenmerken is een steeds belangrijker fenomeen in onze samenleving. Het gebruik van kenmerken die fysisch met het lichaam verbonden zijn vermindert de mogelijkheden voor fraude. Daarnaast kan biometrie gebruiksvriendelijkheid bieden door pasjes, sleutels and codes te vervangen. De vingerafdruk is een van de meest praktische kenmerken omdat deze gebruiksvriendelijk is, een goede prestaties levert, en relatief goedkope sensoren gebruikt die eenvoudig in mobiele hardware kan worden geintegreerd.

Een vingerafdruk bestaat uit een patroon van gekromde lijnstrukturen waar de huid iets hoger ligt dan direct er naast. Deze strukturen worden ridges en valleys genoemd. Veel gebruikte kenmerken die uit een plaatje van een vingerafdruk kunnen worden afgeleid zijn het richtingsveld, de singuliere punten en de minutiae. Het richtingsveld geeft de lokale orientatie van de lijnen weer. Hiermee wordt de globale struktuur of basisvorm van de vingerafdruk beschreven. De singuliere punten zijn de discontinuiteiten in het richtingsveld. Een vingerafdruk bevat maximaal vier singuliere punten die kunnnen worden gebruikt voor het uitlijnen. De minutiae geven de details van de ridge-valley strukturen. Er bestaan twee elementaire typen minutiae: eindpunten en splitsingen. Meestal bevat een vingerafdruk 20 tot 50 minutiae, welke worden gebruikt voor het vergelijken.

Er kunnen twee typen vingerafdrukherkenningssystemen worden onderscheiden: verificatie- en identificatiesystemen. Verificatiesystemen gebruiken vingerafdrukkerkenningstechnieken om de geclaimde identiteit van een gebruiker te controleren. De gebruiker biedt zijn identiteit en een test-vingerafdruk aan aan het systeem, en de test-vingeradruk wordt vergeleken met een referentie-vingerafdruk die uit een database wordt gehaald. De beslissing is gebaseerd op de resulterende gelijkheidsmaat of matching score. Identificatiesystemen vragen alleen om een vingerafdruk. Een persoon wordt geidentificeerd door een database to doorzoeken naar een gelijke vingerafdruk.

De verdelingen van de matching scores van geautoriseerde personen en indringeres overlappen enigszinds. Hierdoor ontstaan twee soorten fouten, die samen de prestatie van een vingerafdrukherkenningssysteem bepalen. De false acceptance rate is de kans dat een indringer wordt geaccepteerd, en de false rejection rate is de kans dat een geautoriseerd persoon wordt afgewezen.

Dit proefschrift richt zich op de drie belangrijkste uitdagingen in de vingerafdrukherkenning: robuste kenmerkextractie uit vingerafdrukken van slechte kwaliteit, het vergelijken van elastisch vervormde vingerafdrukken, en het efficient doorzoeken van databases. Het

onderzoek heeft verbeteringen voor het merendeel van de uitdagingen opgeleverd, wat kan leiden tot een beter presterend systeem.

In een vingerafdrukherkenningssystem kunnen verschillende fasen worden onderscheiden. In de opname fase wordt de vingerafdruk gescanned met behulp van een sensor die meestal is gebaseerd op optische of capacitieve principes. Dit resulteert in een digitaal grijswaarden-plaatje waar verder mee wordt gewerkt in de volgende fasen. De kwaliteit en karakteristieken van het plaatje zijn erg afhankelijk van het type sensor dat wordt gebruikt. Daarom beinvloedt de sensor de prestatie direct, en moeten de verdere fasen worden afgestemd op de specifieke sensor eigenschappen.

De kenmerkextractie fase omvat de berekening van de verschillende kenmerken van de fingerafdruk die bij het vergelijken en zoeken worden gebruikt. In dit proefschrift worden nieuwe methoden voorgesteld voor de nauwkeurige schatting van het richtingsveld, voor het constistent detecteren van de singuliere punten en hun orientatie, voor de segmentatie van fingerafdrukken, voor het verbeteren van de kwaliteit van de plaatjes, en voor het extraheren van de minutiae.

In de vergelijkingsfase worden de kenmerken van de test-afdruk vergeleken met de template die uit een database wordt verkregen. Dit proefschrift presenteert het eerste elastische minutiae matching algoritme, dat expliciet rekening houdt met elastisch vervormde vingerafdrukken. Daarnaast wordt een erg efficient algoritme gepresenteerd dat kenmerk vectoren van vaste lengte gebruikt in plaast van minutiae.

Identificatiesystemen moeten een vingerafdruk vergelijken met alle vingerafdrukken in hun database. Echter, de benodigde rekentijd en de foutkans worden groter bij het doen van meer vergelijkingen. In dit proefschrift wordt indexing methode gebruikt die een beter alternatief biedt voor de veelgebruikte Henry classificatie. Elke vingerafdruk wordt gebruikt als een aparte classe, en deze worden met dalende waarschijnlijkheid vergeleken met de gevraagde vingerafdruk. Door verschillende typen kenmerken te combineren, wordt het mogelijk databases te doorzoeken die 100 maal groter zijn dan zonder het gebruik van deze techniek mogelijk is.

Biography

Asker M. Bazen was born in The Netherlands in 1973. He received his M.Sc. degree in Electrical Engineering from the University of Twente in 1998 for his research on high-resolution parametric radar processing, which he continued for one more year at Thomson-CSF Signaal. In the summer of 2002, he finished his Ph.D. thesis at the chair of Signals and Systems of the University of Twente on various topics in fingerprint recognition, including robust minutiae extraction from low-quality fingerprints, matching elastically deformed fingerprints and indexing large fingerprint databases. Research interests include biometrics, signal and image processing, pattern recognition, and computational intelligence.

List of Publications

[1] A.M. Bazen and R.N.J. Veldhuis. Likelihood ratio-based biometric verification. In *Proc. ProRISC 2002, 13th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2002.

[2] S. Klein, A.M. Bazen, and R.N.J. Veldhuis. Fingerprint image segmentation based on hidden Markov models. In *Proc. ProRISC 2002, 13th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2002.

[3] A.M. Bazen and S.H. Gerez. Fingerprint matching by thin-plate spline modelling of elastic deformations. *Pattern Recognition*, 2002. Submitted.

[4] A.M. Bazen and S.H. Gerez. Achievements and challenges in fingerprint recognition. In D. Zhang, editor, *Biometric Solutions for Authentication in an e-World*, pages 23–57. Kluwer, 2002.

[5] A.M. Bazen and S.H. Gerez. Systematic methods for the computation of the directional field and singular points of fingerprints. *IEEE Trans. PAMI*, 24(7):905–919, July 2002.

[6] A.M. Bazen and S.H. Gerez. Elastic minutiae matching by means of thin-plate spline models. In *Proc. ICPR 2002*, Quebec City, August 2002.

[7] A.M. Bazen and S.H. Gerez. Thin-plate spline modelling of elastic deformations in fingerprints. In *Proc. SPS 2002*, pages 205–208, Leuven, Belgium, March 2002.

[8] A.M. Bazen and S.H. Gerez. Segmentation of fingerprint images. In *Proc. ProRISC2001, 12th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2001.

[9] J. de Boer, A.M. Bazen, and S.H. Gerez. Indexing fingerprint databases based on multiple features. In *Proc. ProRISC2001, 12th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2001.

[10] A.M. Bazen, M. van Otterlo, S.H. Gerez, and M. Poel. A reinforcement learning agent for minutiae extraction from fingerprints. In *Proc. BNAIC 2001*, pages 329–336, Amsterdam, October 2001.

[11] A.M. Bazen and S.H. Gerez. An intrinsic coordinate system for fingerprint matching. In *Proc. 3rd Int. Conf. Audio- and Video-Based Biometric Person Authentication (AVBPA 2001)*, pages 198–204, June 2001.

[12] A.M. Bazen and S.H. Gerez. The construction of an intrinsic coordinate system for fingerprint matching. In *Proc. ASCI Conference 2001*, pages 337–344, May 2001.

[13] P.G.M. van der Meulen, H. Schipper, A.M. Bazen, and S.H. Gerez. PMDGP: A distributed object-oriented genetic programming environment. In *Proc. ASCI Conference 2001*, pages 484–491, May 2001.

[14] A.M. Bazen and S.H. Gerez. Extraction of singular points from directional fields of fingerprints. In *Mobile Communications in Perspective, Proc. CTIT Workshop on Mobile Communications*, pages 41–44, University of Twente, Enschede, The Netherlands, February 2001.

[15] A.M. Bazen and S.H. Gerez. Directional field computation for fingerprints based on the principal component analysis of local gradients. In *Proc. ProRISC2000, 11th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2000.

[16] A.M. Bazen, G.T.B. Verwaaijen, L.P.J. Veelenturf, B.J. van der Zwaag, and S.H. Gerez. A correlation-based fingerprint verification system. In *Proc. ProRISC2000, 11th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2000.

[17] P.G.M. van der Meulen, A.M. Bazen, and S.H. Gerez. A distributed object-oriented environment for the application of genetic programming to signal processing. In *Proc. ProRISC 2000*, Veldhoven, The Netherlands, November 2000.

[18] A.M. Bazen. De complexiteit van vingerafdrukidentificatie. In *Proc. Keys of the Future*, pages 33–38, Enschede, The Netherlands, September 2000.

[19] A.M. Bazen and S.H. Gerez. Computational intelligence in fingerprint identification. In *Proc. SPS2000, IEEE Benelux Signal Processing Chapter*, Hilvarenbeek, The Netherlands, March 2000.

[20] M. Schrijver, A.M. Bazen, and C.H. Slump. On the reliability of template matching in biomedical image processing. In *Proc. SPS2000, IEEE Benelux Signal Processing Chapter*, Hilvarenbeek, The Netherlands, March 2000.

[21] M. Heskamp, C.H. Slump, A.M. Bazen, and M.J. Bentum. On signal processing algorithms to mitigate the influence of wireless communication on the performance of the wsrt. In *Proc. ProRISC2000, 11th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, December 2000.

[22] A.M. Bazen and C.H. Slump. Generalizations of the normalized prediction error. In *Proc. ProRISC99, 10th Annual Workshop on Circuits, Systems and Signal Processing*, pages 631–642, Nov. 1999.

[23] A.M. Bazen. Hoge resolutie radar met adaptieve clutter onderdrukking. *De Vonk*, 17(8):14–20, 1999.

[24] H.E. Wensink and A.M. Bazen. On automatic clutter identification and rejection. In *Proc. 5th Int. Conf. on Radar Syst. (Radar 99)*, Brest, France, May 1999.

[25] H.E. Wensink and A.M. Bazen. On radar detection with automatic clutter rejection. In *Proc. High Resolution Radar Techniques*, pages 18–1 – 18–10, Granada, Spain, March 1999.

[26] H.E. Wensink and A.M. Bazen. On stochastic parametric modelling of radar sea clutter for identification purposes. In *Proc. Physics in Signal and Image Processing (PSIP '99)*, pages 80–85, Paris, France, January 1999.