UNIVERSITY OF OSLO
Department of Informatics

Mapping Fingerprints to Unique Numbers

Ian Michael Trotter
`iant@ifi.uio.no`

29th October 2007

Abstract

As automated fingerprint recognition systems gain popularity, the proliferation of information about unchangeable biometric characteristics causes serious privacy and security concerns. This information may enable an impostor to create a matching fingerprint, and the stored information should therefore be considered extremely sensitive.

This thesis explores a novel method for generating cancellable fingerprint templates that will impede the reproduction of a fingerprint from the stored template, and at the same time allow the same fingerprint to be reused in the case of a compromise.

During enrollment, the proposed method aligns the minutiae points of a fingerprint to a reference coordinate system using the core and principal direction, and creates a hash value based on the set of minutiae points. It then generates Reed-Solomon error correction codes which enable the reproduction of the full set of minutiae points if a certain number of minutiae points are known. It then performs an irreversible Cartesian block transformation on the minutiae points.

During the matching process, the minutiae points of the candidate print are similarly aligned, and transformed using the same Cartesian transformation. A standard matching algorithm is performed on the minutiae sets in the transformed space, which allows the Cartesian transformation to be reversed for the matching minutiae points in the enrolled template. Using the Reed-Solomon error-correction codes generated during enrollment, the entire minutiae point set of the enrolled print can be recreated, provided enough minutiae points could be correctly reversed.

Thus, a matching candidate fingerprint allows an otherwise irreversible transformation on the enrolled print to be reversed. The same hash value created for the fingerprint during enrollment can thus be re-generated when a matching fingerprint is presented.

A proof-of-concept implementation of the method is presented and tested. Although the recognition accuracy of the proposed method was found to be inferior to comparable traditional fingerprint recognition methods, the method nonetheless shows promise as it allows for improved security and privacy of the enrolled data.

Contents

Chapter 1

Introduction

1.1 Background

Biometrics is the study of methods for uniquely recognising humans based on one or more intrinsic physical or behavioural characteristics. A wide variety of systems, business processes, services and applications require a positive verification of the identity of the persons involved, for instance financial transactions, physical access to buildings and areas, and secure access to computer systems. By using biometrics, it is possible to identify a person based on intrinsic traits of the person, rather than by what is in their possession, such as a key or an access card, or some knowledge that is assumed to be exclusive to the person, for instance a password or his mother's maiden name.

There are many biometrics that have been proposed for identification and verification purposes, with varying degrees of success. These include fingerprints, voice, iris, retina, signature, face, gait, hand geometry, ear shape, keystroke patterns, skin reflectance, lip motion, body odor and DNA.

Particularly, the use of fingerprints for biometric identification has attracted a significant amount of interest. The inside surfaces of the hands and feet of humans contain small ridges of skin which facilitate perspiration, enhance the sense of touch, and provide a gripping surface. Fingerprints are part of an individual's phenotype, and are only weakly determined by genetics [15]. Fingerprints are considered to be distinctive to a person, and remain largely unchanged throughout adult life [57].

Fingerprints have been used systematically in forensic science since the early 20th century [15], although there is evidence that the Chinese were aware of the individuality of fingerprints well over 5,000 years ago [15]. The advent of several ink-less fingerprint scanning technologies coupled with the recent increase in processor performance has taken fingerprint recognition beyond criminal identification applications, and into the realm of non-criminal civilian applications such as access control, time and attendance tracking and computer user login.

An automatic biometric recognition system usually consists of four distinct parts: sensor, feature extraction, database, and matcher. The sensor acquires a sample of the biometric, information about certain distinguishing features is extracted from the sample during the feature extraction phase, and the features are then either enrolled in a database for future reference, or matched with a previously stored biometric sample from the database.

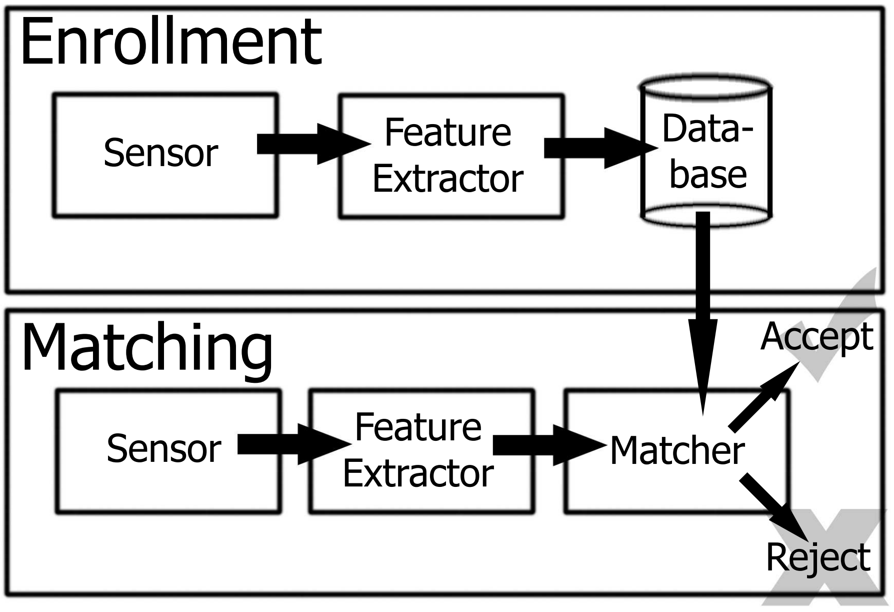

Figure 1.1: A biometric authentication system

The analysis of fingerprints for identification and verification purposes generally requires the comparison of several central features of the print pattern. Matching algorithms are used to compare previously acquired samples of fingerprints against candidate fingerprints for authentication or identification. In order to do this, either the original image must be directly compared to the candidate image [12, 64], or certain features must be extracted and compared [43, 33, 13, 28, 47, 76, 72]. Most of the automatic fingerprint matching systems rely on matching minutiae points across fingerprint images [57]. Minutiae points are the points where there are local discontinuities in the ridge-valley structure of the fingerprint, of which the most commonly used ones are ridge bifurcations and ridge endings. During enrollment, the locations of these points, as well as their orientation and type, are normally extracted and the resulting fingerprint template is stored in a form of database [60]. This information is assumed to represent the fingerprint uniquely, and during verification or identification, the stored template is compared against a newly acquired sample.

1.2 Problem statement

Due to the permanence of the fingerprint, the fingerprint template containing information about central, distinguishing features of the fingerprint should be considered highly sensitive. This information may allow an attacker to reconstruct a matching fingerprint [17, 26], and allow for database cross-referencing. Storing fingerprint templates in a manner that mitigates such illicit use is thus of great importance.

This thesis explores a novel approach to the issue of secure digital storage of fingerprint data. The approach attempts to ensure that a fingerprint template may be revoked and reissued in the case of a compromise, and that the stored fingerprint template will be effectively useless for fingerprint reconstruction and for database cross-referencing, whilst still enabling matching.

1.3 Method

Applying a one-way function to the fingerprint data before storing it in a database could make the details of the fingerprint unrecoverable from the stored representation. Such functions are familiar in the field of cryptography, where they are used as cryptographic hash functions [55, 63]. These functions turn input data into a relatively small number that may serve as a relatively unique identifier for the input data, and are designed with the intent that there should be no practical way of reproducing the input data from the output value of the function. This is often used for password storage, to ensure that passwords cannot be guessed even when given access to the password database.

This idea cannot, however, be directly transferred to fingerprint recognition systems. The image of a fingerprint will look different every time it is scanned due to changes in the reading conditions. The factors that will affect the fingerprint image include residue on the scanner, finger placement (which may introduce rotation and translation), skin elasticity (causing non-linear deformation of the image), skin condition (cuts and scrapes), and humidity (dilating or eroding the fingerprint ridges). This causes different sets of features to be extracted by the algorithms for different scans of the same finger, and processing the extracted features with a regular cryptographic hash function will naturally yield different results. Nonetheless, a vauable lesson can be learned from the field of cryptography.

The proposed method starts by subjecting the fingerprint image to a standard feature extraction process: the locations and orientations of the ridge endings and bifurcations are found.

Despite the reading conditions, there are certain features of a fingerprint that can be located quite reliably across acquisitions, for instance the cores and deltas of the fingerprint [38, 19, 40, 46]. Using a stable reference point and a principal direction, it is possible to define a coordinate system for the fingerprint that will be relatively stable across acquisitions, somewhat offsetting the effects of finger rotation and translation.

During enrollment, a cryptographic hash value is calculated based on the locations and orientations of the minutiae points relative to this coordinate system. This value will be unique for the particular acquisition of the fingerprint due to the differing reading conditions.

In addition, Reed-Solomon error correction codes are calculated, so that the entire set of minutiae points may be recreated if only a given number are known exactly.

The minutiae points that were detected are then transformed with a Cartesian transformation: the coordinate system is divided into blocks which are randomly shuffled, so that the original location of a single minutiae point cannot be guessed by looking at the location of it in the transformed space [59, 58]. This step is analogous to the applying a cryptographic hash function, in the respect that it cannot be easily reversed.

The transformed minutiae points and the Reed-Solomon error correction codes can safely be stored, since they do not reveal the original whereabouts of the minutiae points in the fingerprint, in much the same way that a cryptographic hash of a password does not reveal the actual password.

In the matching process, the same feature extraction is performed, the same coordinate system is generated, and the same Cartesian transformation is performed on the minutiae points. By looking at which minutiae points from the two acquisitions are close in the transformed space, it is possible to deduce which block the minutiae point from the stored template originated from, as it is likely to be the same block from which the minutiae point from the candidate image came.

The matching process thereby provides partial reversibility of the Cartesian transformation performed on the enrolled fingerprint. Combining the subset of reversed minutiae points with the Reed-Solomon error correction codes generated during enrollment, the entire set of minutiae points found during enrollment can be recreated. Having retrieved the entire set of minutiae points that were detected during enrollment, it is now trivial to regenerate the cryptographic hash value that was created for the minutiae set during enrollment.

If the cryptographic hash value generated during the matching process is the same as the one created during enrollment, the fingerprints can be considered mates.

The method is implemented using well-known image enhancement and minutiae detection techniques [30], as well as the famous Poincare index method for detecting stable points [38]. The Cartesian transformation performed on the features is one of many known methods for creating cancellable templates [58].

The novel concept in this thesis, however, is that of using information from matching in the transformed feature space to partially reverse the Cartesian transformation, and combining the reversed information with Reed-Solomon error correction codes in order to obtain the full original feature set of the enrolled print. From the entire original feature set, a key can be generated. For a given enrolled print, this key will be the same each time a matching fingerprint is presented.

The outlined algorithm is subjected to the rigorous testing procedures used in the Fingerprint Verification Competition (FVC) [44], using the fingerprint image databases used in the competition in 2000. This makes it easy to compare the accuracy of the outlined algorithm and state-of-the-art fingerprint recognition algorithms. Although this particular testing procedure only measures recognition accuracy, and not security per se, it will nevertheless be interesting to compare the results obtained by this algorithms with other algorithms that participated in the FVC in 2000.

Using the same test methodology and databases, the method presented above is compared to a reduced version of the method which only involves feature extraction, alignment and matching - skipping the Cartesian transformation and Reed-Solomon error correction coding phases. If those stages are skipped, the entire algorithm is reduced to a fairly standard fingerprint recognition algorithm, using the same foundation of feature extraction and matching algorithms as the proposed algorithm. Comparing the results of these will provide valuable insight into how the transformation and error correction phases, which are the most innovative and particular features of this approach, affect the fingerprint recognition accuracy.

1.4 Organisation of the Thesis

Chapter 2 sums up background and related work.

Chapter 3 presents the details of the proposed algorithm.

Chapter 4 discusses the proof-of-concept implementation of the method, along with the chosen run-time parameters.

Chapter 5 presents the testing framework and the criteria the proposed method will be evaluated against.

Chapter 6 presents the results from the testing and discusses the implications of these.

Chapter 7 summarises the findings of this thesis, presents ideas for further improvement, and outlines related work that may provide additional insight into the proposed approach.

Chapter 2

Background and Related Work

2.1 Person Authentication

Being able to recognise and tell apart individuals is fundamental to modern society. Gaining access to your home, picking up the right child from kindergarten and performing financial transactions with a credit card are ordinary everyday occurrences that all require some sort of reliable verification of ones identity; in these cases either proven by exclusive possession (house keys and credit card), exclusive knowledge (PIN number of the credit card), biometric recognition (a childs intuitive recognition of its parent) or a combination thereof.

The applications for recognition and identity verification can be coarsely divided into three categories [15]:

Physical access control For instance controlling access to office buildings and controlled areas such as airports.

Logical access control For instance the right to manage a bank account, or the right to use computer resources.

Ensuring uniqueness For instance ensuring that a person is only enrolled once into the social security system.

The purpose of each of these is to ensure that a given application or service is used solely by legitimate users.

In general, there are considered to be three different ways to determine the identity of an individual [49]:

Exclusive possession The subject possesses a physical object that is assumed to be exclusive to authorised users that can be used as a token for identification or verification of the user's permissions. Examples of such an object include an access card, a key, or a passport.

Exclusive knowledge The subject knows or has access to information which is exclusive to that user. Upon request, the user can present the information as evidence for a claimed identity. This could include passwords and passphrases, PIN numbers or ones mother's maiden name.

Biometrics This is an attribute that is considered intrinsic and inseparable from the subject. The subject can present the attribute for identification purposes, or as evidence for a claimed identity. Such an attribute may for example be DNA, a fingerprint, a birthmark or simply the appearance of ones face.

Many applications combine several of these in order to deliver a higher level of authentication assurance - for example an ATM card with an associated PIN number combines both possession and exclusive knowledge. When more than one of these methods is used, it is usually termed multifactor authentication in order to reflect the use of multiple authentication factors.

In short, each restricted application follows a strict protocol, using one or more authentication factors in order to determine whether a subject should be granted access to the application or not. The authentication protocol describes the interaction between the system and the subject that is necessary for the system to grant the subject access to the application.

2.2 Biometrics

Biometrics is the study of using intrinsic biological traits to uniquely identify individuals. For humans, identifying individuals is an apparently simple intuitive process that engages the entirety of the perceptory system, and applies mental processes which are not fully understood. However, creating systematic approaches that can be performed reliably by automated systems has proven a venerable challenge. Within computer science, automated biometrics is often considered to be within the field of pattern recognition.

It is common to distinguish between the use of biometrics for identification and the use of biometrics for verification [15]. When using biometrics to perform identification, the biometric measurement is used to select an individual from a list of individuals. The identity of the individual is thus determined solely by matching the presented biometric against a number of samples of known origin. However, when performing verification, the subject presents both a biometric sample and a claimed identity. The biometric sample is then compared against a single sample known to belong to the claimed identity in order to verify

Static Biometrics
DNA
Fingerprint
Face
Iris
Retina
Hand geometry
Ear shape

Behavioural Biometrics
Signature
Voice
Gait

Table 2.1: List of Biometrics

that the presented biometric sample matches that of the claimed identity. The former process is naturally more demanding, but establishes the identity of the individual regardless of the subject's claimed identity, whereas the latter process simply confirms or refutes a single identity claim.

Many different biometric characteristics have been suggested for recognition purposes, some of which are listed in table 2.1. The table distinguishes between static biometrics, where the acquisition of the sample is performed at one instant, and behavioural biometrics, where the characteristics are measured over time. Some of the biometrics in the table have been researched more thoroughly than others, but most of them have not yet reached a level of maturity where wide application is feasible [15].

Evaluation of a Biometric

Not every biometric would be suitable for every application. An important question that arises is how to evaluate a biometric, and how to select the most appropriate biometric for a particular application. There are four basic properties that must be taken into account, and are exhibited to a different degree by each biometric [32]:

Universal It is desirable that the biological characteristic is exhibited by the entire population, so the biometric system does not categorically exclude any significant group of users.

Unique The biological characteristic must exhibit enough unique features so that each individual in the entire population may be differentiated by it.

Permanent The characteristic should not change significantly throughout the lifetime of the individuals - it should allow for recognition regardless of how long ago the reference sample was obtained.

Collectable The acquisition of the biometric sample should not be too cumbersome or intrusive, and should not unreasonably inconvenience the subject.

These represent some of the most important properties that can be used to evaluate the suitability of a biometric for a specific application. Although most biometrics fulfill the criteria to a certain extent, there is no biometric that perfectly fulfills all of these criteria - it is necessary to compromise and select which considerations are most important for a particular application.

2.3 Fingerprints

A Short History of Fingerprints

The skin on the inside of the hands contains structures whose purpose appear to be to improve grip, increase sensitivity in the hands and allow for easier perspiration [15]. More specifically, the tips of the fingers contain skin structures consisting of ridges and valleys, each of which displays a distinctively unique pattern [57].

There is evidence that the Chinese were aware of the individuality of fingerprints already 5,000 years ago [60]. However, in more recent times, the systematic use of the fingerprint for identification purposes was pioneered by forensic science and law enforcement in the 19th century. Elaborate and advanced manual classification schemes gradually evolved, such as the Henry system [15], which was adopted by the Federal Bureau of Investigaion in the United States of America, and the Vucetich system [27], for efficiently determining the identity of individuals, mainly criminals, by using fingerprints.

Fuelled by these developments, the use of fingerprints for identification purposes gained traction. By the early 1960s, the use had increased to such an extent that the labour associated with the manual systems had become impractical, and automated systems were commissioned [15]. The proliferation of advanced digital technology ushered yet further developments in the area; cheaper sensors and more advanced recognition algorithms has caused widespread use of fingerprints both for identification and verification in applications outside law enforcement, such as physical access control, computer login, and attendance tracking [7, 1, 6].

Evaluation of Fingerprints as a Biometric

The universality of fingerprints is often taken for granted. All primates with fingers are assumed to have fingerprints [15], although there have been a small number of documented cases where persons have no ridge-valley pattern on the tip of their fingers [25].

Although the individuality of fingerprints has been contested in legal processes [3], a number of studies present credible estimations supporting their uniqueness [57]. The fingerprint pattern is part of an individual's phenotype, and thus determined by a complex interaction of genetic and environmental factors. This means that identical twins can be told apart by their fingerprint pattern, although they share the same DNA [66].

A person's fingerprint remains largely unchanged throughout adult life [51]. Even though scratches and small scars may appear on the fingerprint, the pattern usually remains in a recognisable condition, such as shown in figure 2.1. Should the fingerprint disappear entirely, such as in a burn, the original pattern will normally grow back [15, 60].

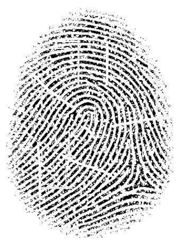

Figure 2.1: Fingerprint image with scratches

Fingerprints are easily collectable and can be gathered in a very non-intrusive manner - a visible imprint may be collected using only some ink and a piece of paper. Indeed, such a collection technique has persisted for many years, and is to this day still used in a number of systems, for instance by Brazilian immigration control. Many applications these days are utilising inkless sensors [69, 11, 24, 39], which are becoming increasingly affordble and reliable. There are a number of competing technologies to select from: optical, CMOS capacitance, thermal and ultrasound. However, sensing introduces distortion to the fingerprint image, as the skin deforms under pressure. Another downside of this easy collectability, is that we inadvertently leave our fingerprints all over the place.

All in all, the fingerprint scores fairly high on all four of the biometric assessment criteria presented earlier.

2.4 Biometric System Design

Although a wide variety of biometrics have been proposed, each suitable for different applications, any biometric system can be viewed as essentially a pattern recognition system. As such, it is usually considered to consist of four distinct loosely-coupled parts, each performing an essential and well-defined function [32, 15]:

Sensor The sensor acquires a sample of the presented biometric. For instance, a tissue sample, a photography of the face, an image of a fingerprint, an image of a signature, a thermogram of a face, an image of an iris or an image of a retina.

Feature Extractor The feature extractor extracts the data from the raw biometric signal from the sensor that will be necessary in order to compare it to other acquired samples. The result of the feature extraction phase is a biometric template, assumed to include all the key information necessary to uniquely identify the individual.

Database When the relevant information has first been extracted, it must be stored for future reference. The biometric template is thus stored in some sort of database. Examples of such a database are the archives of ten-print fingerprint cards used by the Henry system, or passports, which normally contain a photo and the signature of the holder. Automated biometric recognition systems may for example store a digital representation of a fingerprint on a smart-card [50, 54, 77] or in a central database [33].

Matcher The matcher component compares two biometric templates, and attempts to determine whether the templates represent the same individual. In some cases, this process may be performed by a human examiner, for instance by a fingerprint expert or a shopkeeper comparing the signature just provided with a signature on a credit card. Yet in other cases, computer-implemented algorithms may perform the matching, such as in automatic fingerprint recognition systems [60].

Each biometric recognition system consists of these four loosely connected components. A general such system is illustrated in figure 2.2.

2.5 Automatic Fingerprint Recognition Systems

Automatic fingerprint recognition systems have been in large-scale operation since the 1960s [60], and has been a popular research topic since.

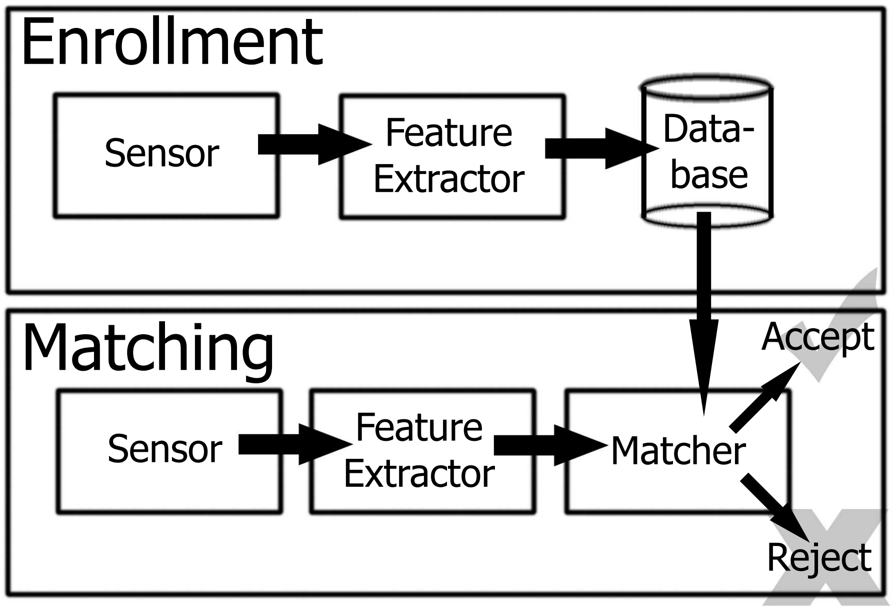

Figure 2.2: A biometric authentication system

Although such automatic systems were originally used in forensic science and law enforcement, recent advances in technology has stimulated a number civilian applications of the technology. Some countries have discussed the use of digital representations of fingerprints in passports [7], laptop computers are increasingly delivered with embedded fingerprint sensors for login [6], and immigration control in several countries, such as the United States of America, require that aliens enroll their fingerprint into a central database.

Being a particular kind of biometric recognition system, an automatic fingerprint recognition system also consists of four distinct parts, as discussed earlier; a sensor, a feature extractor, a database and a matcher.

Sensors

In the first part of the 20th century, fingerprint systems were usually based on ten-print ink cards, and the acquisition process entailed dipping each finger in ink and rolling the finger from one side of the nail to the other onto the fingerprint card. Through the last few decades, however, reliable and affordable inkless sensors have surfaced, based on a number of different technologies:

Optical This sensor type usually uses a CCD or CMOS camera to acquire a digital image of the fingertip, which is rested on a prism [15, 11]. This technology has been in use for many decades - it was, for example, used in medical studies as early as 1966 [9].

Thermal Thermal sensors measure the changes in temperature due to the ridge-valley structure of the fingerprint as the finger is drawn across a thin sheet of pyroelectric material [39].

Capacitance Capacitance sensors use the difference in charge between the ridges and the valleys of the fingerprint as the finger is drawn across a CMOS grid chip in order to acquire a fingerprint image [69].

Ultrasound Ultrasound sensors scan the tip of the finger by the reflection of an ultrasonic beam, thereby creating an accurate depth-map of the finger [24].

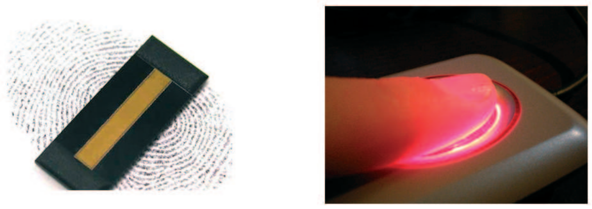

Figure 2.3: A capacitance swipe sensor and an optical sensor

Increasingly affordable sensors will facilitate the proliferation of fingerprint technology. Lately, a large number of laptop computers have been produced with integrated capacitance sensors intended for user login [6], and border control in the USA enrolls aliens using optical sensors.

The output from this stage of the process is usually a greyscale image, from which the feature extraction module can extract the distinguishing features.

Feature Extraction

The feature extraction process is concerned with processing the image from the sensor and extracting features which can be used to uniquely identify the subject. A fingerprint contains many distinctive features, and many different features have been exploited for automatic recognition purposes.

Local discontinuities in the fingerprint ridge structure, which have been designated minutiae points, are used in a large number of algorithms [43, 33, 13, 28, 47, 76, 72]. Although there are many different types of minutiae, the most commonly used minutiae types are ridge endings, where a single ridge line abruptly ends, and ridge bifurcations, where a single ridge line divides into two lines. Some of these minutiae types are illustrated in figure 2.4. The overwhelming majority of biometric systems extract these minutia points, and ISO has even created a standard format for fingerprint templates in order to facilitate interoperability between systems [56].

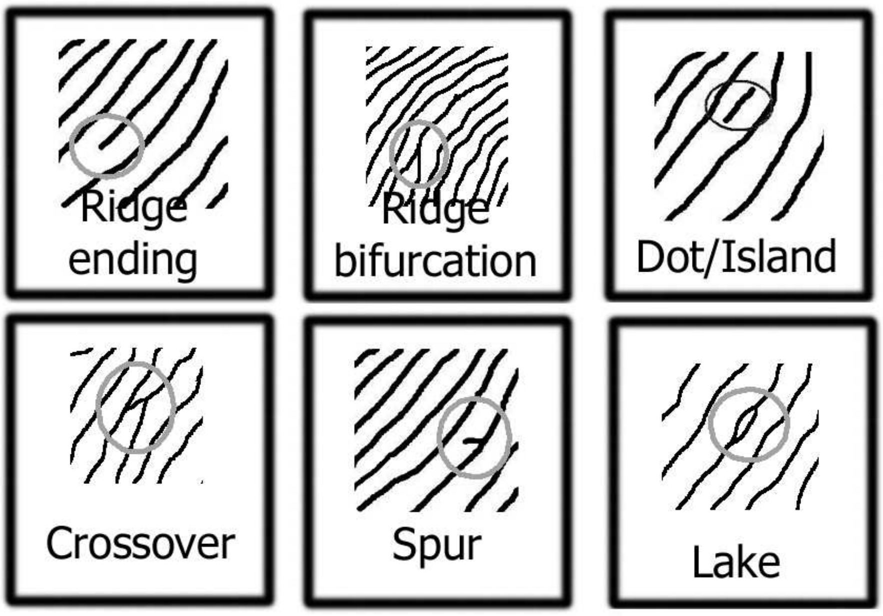

Figure 2.4: Examples of various minutiae types

Many algorithms also detect singularities in the fingerprint image, such as the core and the delta [34, 18, 19], which are global features of the fingerprint. The core is normally defined as the point with the highest ridge curvature, and the delta is normally the area where there is a triangulation or a dividing of the ridges. Some examples of these singularities in different fingerprint types are shown in figure 2.5.

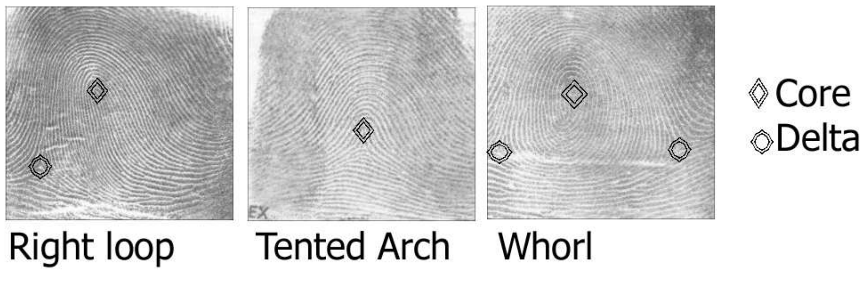

Figure 2.5: Location of core and delta in fingerprints

Although these are the most common features that are extracted in the feature extraction process, they only represent a fraction of the uniquely identifiable information contained in a fingerprint. There are a number of other features that could be extracted for recognition purposes, such as the location of sweat pores on the ridges [31], entire ridge lines [47], or simply the entire image so that an image correlation operation may be performed during the matching phase [12, 64].

The feature extraction process usually combines general image processing methods and more specialised fingerprint image processing methods in order to extract the necessary data. Minutiae detection algorithms often enhance the image and binarise it, before thinning the ridges and detecting the minutiae points [29], although there are some minutiae detection algorithms that work directly on the greyscale fingerprint image [45]. The precursors of a basic minutiae extraction process, composed of enhancement, binarisation and thinning is illustrated in figure 2.6. Detecting ridge endings and ridge bifurcations in the thinned representation is a simple matter of traversing the image and looking for locations where the one pixel wide ridge either ends or forks.

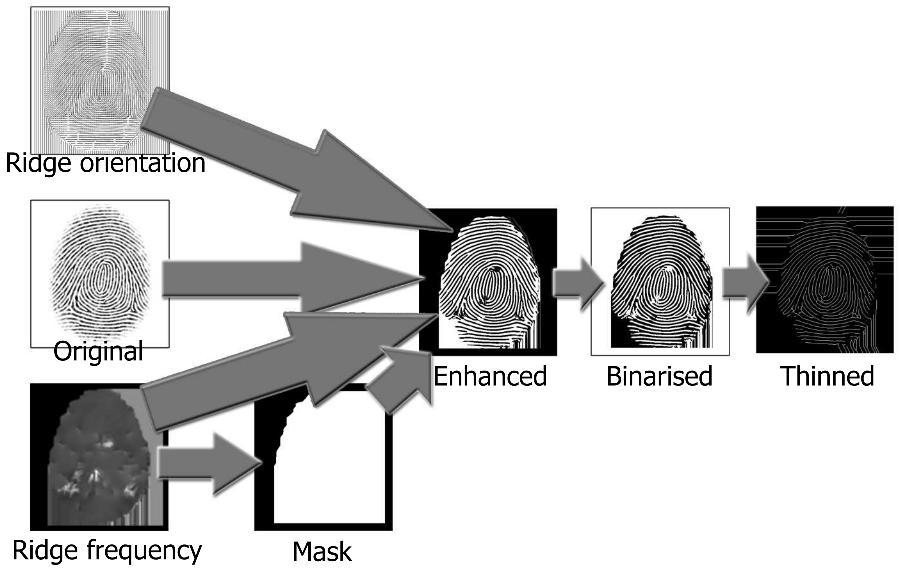

Figure 2.6: Image enhancement overview

Database

When a fingerprint is first enrolled into the system, the fingerprint template is usually associated with an individual and stored in some kind of database as a reference template. Fingerprint templates from subsequent acquisitions may be compared to the reference template, and the identity of the subject can be determined by considering the similarity of the templates.

Manual systems have traditionally used filing cabinets with ten-print fingerprint cards [15, 60]. Modern systems, however, often store the templates in a central database, such as a server on a network [33], although various types of distributed databases have also been suggested. For instance, smart-cards [50, 54, 77] and passports containing biometric data [7] represent some examples of distributed databases for fingerprint template storage.

The chosen database type also affects the requirements to the fingerprint template. A smart-card or passport, for instance, may have a very limited storage space, meaning a very compact representation of the fingerprint must be chosen. Central databases may impose yet other restrictions on the fingerprint template, for instance regarding the security and privacy of data retainment and transmission.

Matcher

The matching process compares two fingerprint templates and decides whether or not they represent the same individual. Since it uses the templates from the feature extraction process to determine this, the matching process is closely connected to the feature extraction process - essentially, the matcher must operate on the data extracted during the feature extraction phase, be it minutiae points [72], ridge lines [47] or filterbank responses [34].

Matching of fingerprints is a fairly demanding process - the matcher must take into account a number of effects during the acquisition that may produce large differences in the samples and attempt to consolidate them. These effects may include rotation and translation of the finger with respect to the sensor, humidity and other environmental factors that can cause dilution or erosion of the ridge lines, scars and bruises on the finger and non-uniform distortion of the elastic finger skin as it makes contact with the sensor. Some of these effects are illustrated in figure 2.7. The output from the feature extraction process may therefore be very different for each collected sample, even for the same finger, and must be taken into account when performing the matching.

A very common algorithm used during the matching process is the Hough transform [23]. A number of minutiae matching algorithms and ridge line matchers are based on this transform.

The output of a matcher is usually a score indicating the similarity of the samples, which is then compared to a threshold value to determine whether the subject should be accepted or rejected.

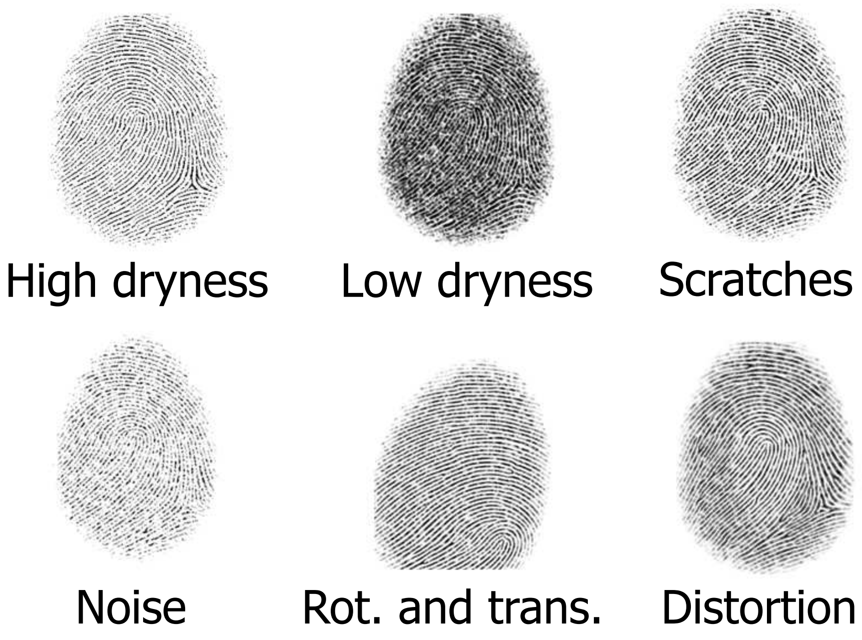

Figure 2.7: Acqutitions emphasising various effects

2.6 Fingerprint Recognition System Security

Many automatic fingerprint recognition systems have been demonstrated to be easily subvertible. One particularly popular television program has foiled some of the allegedly most advanced fingerprint recognition systems commercially available with a simple photocopy of a finger [8], and several other similar demostrations exist [48, 26].

The four general criteria for a biometric discussed earlier; universality, uniqueness, permanence and collectability, can be used to evaluate the suitability of a biometric for a particular application, but do not guarantee the security of a system based on the biometric per se. A system based on the chosen biometric has additional requirements, such as security of the entire system and privacy of the information used by the system.

Thus the question arises of how to evaluate the security of a fingerprint recognition system.

A commonly accepted approach is to attempt to estimate the accuracy of the system by running the system on a large number of fingerprint images, and check that it accepts and rejects the fingerprints correctly [20]. A fingerprint image that should be accepted, but is wrongfully rejected by the system is called a False Reject. A fingerprint that is wrongfully accepted by the system when it should be rejected, is called a False Accept. It is common for a system to report the rate at which these occur, giving rise to the accuracy figures termed False Reject Rate (FRR) or False Non-Match Rate (FNMR), and False Accept Rate (FAR) or False Match Rate (FMR).

In essence, these figures are only representative of the security when considering zero-effort attacks; where the impostor makes no effort to subvert the system except presenting his own biometric on the off chance that he may be mistaken by the system as a legitimate user. Although the FRR and FAR are useful and necessary figures, they are not representative for the security of the system as such, and really only reflect the accuracy of the recognition system.

To deal with other threats, it is necessary to focus on other measures than increasing system accuracy.

A scenario commonly portrayed in mass media is where an impostor has access to a latent fingerprint present on an object that a legitimate user has touched - this fingerprint is lifted and presented to the system [8]. Fingerprints are particularly vulnerable to this kind of attack, as latent fingerprints are difficult to avoid, and fairly easy to lift and employ. A threat of similar nature, although more macabre, is that of somehow separating the fingerprint from the legitimate owner, and subsequently presenting it to the system.

Numerous measures have been suggested to prevent the success of such attacks, mainly reading vital signs during acquisition, such as heat, heartbeat and exudiation of perspiration [68], to ensure that the fingerprint is presented by the legitimate possessor. Other solutions suggest that the system be augmented by other authentication factors, such as possession or knowledge [70, 50].

Such attacks, however, are difficult to model, and, similarly, the resistance to such attacks is difficult to quantify, which is why the biometrics community often only reports accuracy figures for fingerprint systems.

With the proliferation of digital fingerprint systems, another variant of the threat that an illegitimate user may present a legitimate fingerprint has recently surfaced. By gaining access to the database component of a fingerprint recognition system, or to the communications channel linking the database with the rest of the system, an impostor may potentially have access to all the fingerprint templates contained therein. For a long time, researchers assumed that the feature extraction processes which generate the fingerprint templates were largely one-way functions. It has recently been demonstrated, however, that it is possible to recreate a fingerprint based on a standard fingerprint template [17].

The potential damage of an attacker gaining access to a database of fingerprint templates is much larger than that of an attacker gaining access to a latent fingerprint, since the digital template does not gradually erode, such as a latent fingerprint, or require the physical presence of the attacker. Furthermore, a single compromised database may give the attacker access to a large number of fingerprint templates at once. This is analogous to a threat faced in the more general computer security field where an attacker gaining access to a cleartext password database is considered more serious than an attacker looking over someones shoulder while a password is typed. The compromise of a fingerprint database, however, is even slightly more serious than the compromise of a password database: passwords may easily be revoked and reissued, whereas fingerprints are immutable and any compromise is permanent.

Thus, the database component of a fingerprint system must be thoroughly secured in order to ensure the security and privacy of the system. A collection of approaches have been suggested to deal with this issue.

The first type of solution implements various means of securing the information, physically or logically. A large number of suggested solutions are based on tamper-resistant smart cards [50, 54, 77]. The fingerprint template can be stored on the card, and the matching process may be performed by the card itself, such that no fingerprint information must be exchanged with the rest of the system - only the decision of the authentication process. The use of a physical token - the smart card - implies that this effectively constitutes a two-factor authentication process, where the card physically protects the sensitive data. There are also examples of systems that use traditional encryption methods to conceal the fingerprint template [41].

The second type of solution combines a lesson from password management in the more general computer security field, where one-way functions have already been employed for password storage for many decades [55], with the concern that biometrics are irrevocable. It was stipulated that by processing each password in a password list with a one-way function before storing it, an intruder would be unable to obtain the actual passwords even though he may have access to the password list. Similarly, performing a one-way function on a fingerprint template may transform the data so that the original fingerprint cannot be recreated from the stored representation. Furthermore, if the one-way function is somehow parametrised, a fingerprint template may be changed by modifying the parameters, and thus a compromised template can be revoked and a different template issued by simply changing the parameters of the one-way function. As opposed to the first type of solution, which involved physically or logically implementing means of protection for the template, these kinds of approaches involve actually modifying the template until it is unusable for any potential attackers, yet still usable for the system. This idea has spawned the field of cancellable biometrics.

2.7 Cancellable Biometrics

Cancellable Biometrics is a relatively young field that has arisen as a response to privacy and security concerns within biometrics [14, 61]. Essentially, this field deals with the issue that a biometric is traditionally not revokable and any compromise is permanent. Similar to how passwords often are treated [55], the biometric can be processed with a potentially parametrised one-way function in order to create a representation of the biometric which is changeable, since one can change the parameters, and which cannot be used to reconstruct a copy of the original biometric.

There are four basic requirements for such a cancellable template [46]:

Irreversibility As pointed out earlier, it should be impossible, or at least infeasible, to recover the original biometric from the cancellable template.

Reusability Different cancellable templates can easily be generated, both in order to use different templates for different applications, and in the event of revocation and reissue.

Diversity Separately generated cancellable templates should not match each other, or the original template.

Unaffected performance The cancellable template should be as unique as the fingerprint itself, and should not deteriorate the entropy of the fingerprints. If it does, the performance of the matching process will suffer, returning more false matches simply because distinguishing features have been removed from the original template.

Furthermore, the cancellable template must be as tolerant as the original template to intrauser variability, or the matching process will reject a larger number of genuine matches because templates that should match have been transformed such that they no longer match.

For fingerprints in particular, a plethora of different methods have been proposed, which can be loosely divided into four categories [58]:

Biometric Salting

In the wider computer security field, passwords are often "salted": the password is combined with a pseudorandom string before being hashed and stored in the database. The use of a pseudorandom string increases the entropy of the stored value, and thus impedes dictionary attacks. Similarly, biometric salting combines the biometric template with user-specific pseudorandom information to increase the entropy of the result.

A particularly well-known example of this technique is BioHashing [71]. BioHashing uses iterated inner-products between tokenised random data and a rotation- and offset-invariant representation of the fingerprint generated using the wavelet Fourier-Mellin transform on the fingerprint image. When subjected to this process, two imprints of the same finger will generate highly correlated bitstrings, whereas two different fingerprints will generate very different bitstrings. A fingerprint can be revoked by simply changing the random data, and the features of the fingerprint are not evident from the bitstring stored during enrollment, as the inner-products are one-way functions.

This approach, however, does not operate solely on the biometric - it also requires tokenised random data - and thus effectively constitutes a multifactor authentication protocol. Comparing this method with traditional biometric recognition systems that operate exclusively on the biometric is therefore not entirely trivial.

Biometric Key Generation

Solutions in this category attempt to generate a key directly from the biometric signal. During verification, it is simply checked if the newly generated key matches the stored key. However, computing a robust binary representation from a noisy signal without additional information is a venerable challenge.

One particularly interesting such solution, based on the iris biometric, uses a complex interactive process in order to obtain as pure an acquisition as possible. A short 256-byte code is computed from the sample. This code is close to other acquisitions of the same iris, when measuring the Hamming distance. Using a large number of samples during enrollment, a canonical biometric is created by a voting mechanism, along with error correction codes that allow reconstruction of the canonical biometric from an imperfect sample, provided it matches closely enough [21].

Many aspects of this solution are intriguing, particularly the use of error correction codes, yet it is not entirely clear how to accomplish the same feat using other biometrics than the iris, especially those which are by nature more volatile and exhibit larger intrauser variation.

Solutions based on other biometrics, such as typing pattern, voice and signature, have also been proposed [53, 52, 75].

Fuzzy Schemes

Another approach in early development, so-called fuzzy schemes, generates a public string and a private string at the time of enrollment. During verification the public string is combined with the new biometric measurement in a manner that will reproduce the private string if the measurements match each other closely enough.

Fuzzy schemes have a robust general framework [22, 36, 35, 42, 73], which is subsequently specialised for different biometrics. In such a scheme, the private string need never be stored, as it can be generated by combining the public string and a matching biometric. The public string reveals no information about the features of the biometric, and often contains error correction codes which provide adequate error tolerance during the matching phase.

A particular fuzzy scheme devised for fingerprints encodes a secret into the stored minutiae data that can only be retrieved when a matching print is presented [74]. This type of scheme has many advantages, and has been validated in early prototype.

Noninvertible Transforms

This class of techniques involves performing an irreversible transformation on the biometric signal or features prior to storage. The transformation is often constructed such that traditional feature extraction and matching algorithms can continue to work unaffected, so as to provide backwards compatibility with existing recognition methodologies.

Since the transformation is noninvertible, information about the original features about the biometric remain secret even if the transformed features are known.

A number of such functions have been suggested and investigated [10, 67].

One of the earliest examples of such a transformation is the Cartesian transformation [59], which simply splits the two-dimensional Cartesian feature space into regular, rectangular blocks and shuffles them. The minutiae points are thus effectively shuffled around based on which block they are in. If several blocks in the input are mapped to the same block in the output, there is no telling where a given minutiae point originally came from, even if the transformation parameters are known. Although others exist, this particular transformation is one of the most rudimentary transformations that have been suggested, and is known to disrupt the recognition accuracy slightly.

2.8 New Method for Generating Cancellable Fingerprint Templates

In this thesis, I explore a novel approach to generating cancellable fingerprint templates which combines elements of these four categories and error correction codes. In short, a key is generated at enroll time, error-correction codes are generated, and a noninvertible transform is performed on the fingerprint features. During the matching process, pieces of information gleaned from matching the candidate template with the master template in the transformed feature space provides partial reversibility of the noninvertible transform - information which, in the case of a match, recreates the original features when combined with the error-correction codes. Having recreated the original features of the master fingerprint again, the same key that was created at enroll-time can be regenerated.

Effectively, the method described herein gives each fingerprint a unique number at the time of enrollment, and recreates this number when subsequently presented with a matching fingerprint. At each enrollment this number will naturally be different due to differing reading conditions, and, furthermore, no sensitive biometric information is stored in plain text at enrollment, thus impeding fake fingerprint generation, providing revocability, and prohibiting database cross-referencing.

Chapter 3

Method

The method presented in this thesis uses information gleaned from the candidate image during matching to partially reverse an otherwise irreversible transformation that was performed on the minutiae points at the time of enrollment. The information gathered through the partial reversal is used in conjunction with Reed-Solomon error-correction codes to provide a full reversal of the transformation performed on the minutiae points during enrollment, and thus a foundation for generating the same key in the matching process that could be generated in the enrollment process. A match can then easily be evaluated by simply checking whether or not the same key was generated during matching as during enrollment.

3.1 Overview

Upon enrollment, the fingerprint image is first enhanced, and the locations of ridge endings and bifurcations are detected. The core of the fingerprint, together with an estimated principal direction, is used to align the minutiae points to a reference coordinate system. Reed-Solomon error correction codes are generated for the minutiae points such that the entire set can be reproduced if the exact information of only a certain amount of the minutiae points is available. A key, or rather a cryptographic hash value, is generated based on the entire set of minutiae points. Subsequently, the space in which the minutiae points reside is subjected to a Cartesian block transformation, which shuffles the minutiae points around such that the original location of any minutiae cannot be known. The template that is stored contains the list of transformed minutiae points, the checksums from the Reed-Solomon coding, and the information required to perform the same Cartesian transformation on a candidate fingerprint during the matching process.

During the matching stage, the same image enhancement and minutiae detection algorithms are used, and the resulting minutiae points are transformed using the same Cartesian transformation that was performed during enrollment. A matching algorithm matches the minutiae points of the candidate fingerprint with those of the master fingerprint in the transformed feature space. The matching process will reveal the original location of the matched minutiae points in the master fingerprint, and thus allows the Cartesian block transformation to be reversed for these minutiae. If the number of minutiae points for which the transformation can be reversed by this procedure is sufficient, the recovered minutiae points can be combined with the Reed-Solomon codes generated during enrollment in order to reproduce the entire original set of minutiae points that were found in the master fingerprint. Having generated the same set of minutiae points as during enrollment, we can now use these to generate the very same key, or hash value, as was done during enrollment. If this newly generated key matches the one that was produced during enrollment, the fingerprints can be considered mates, as it indicates that enough minutiae points matched between the fingerprints to entirely reverse the Cartesian block transformation in conjunction with the Reed-Solomon error correction codes.

The enrollment and the matching processes are illustrated in figures 3.1 and 3.2, respectively.

The image enhancement and minutiae detection is based on a well-known and common approaches [29, 30]. The core detection is based on the Poincare index method [38], and the Cartesian transformation performed on the features has also been thoroughly investigated [58].

The concept pioneered in this thesis, however, is using information from matching in the transformed feature space to partially reverse the Cartesian transformation, and combining the reversed information with Reed-Solomon error correction codes in order to obtain the full original feature set of the enrolled print. From the entire original feature set, a key can be generated. For a given enrolled print, this key will be the same each time a matching fingerprint is presented.

3.2 Fingerprint Image Enhancement

The fingerprint images acquired by the sensors are often prone to noise, scratches and various other artifacts and effects due to changing acquisition conditions. By using image processing methods specifically tailored for fingerprint image enhancement, it is usually possible to improve the quality of the image by filtering out some of the noise and compensating for certain acquisition effects. The end result of such a fingerprint image enhancement technique is a binary image that emphasises the ridge-valley structure of the fingerprint, and as accurately as possible

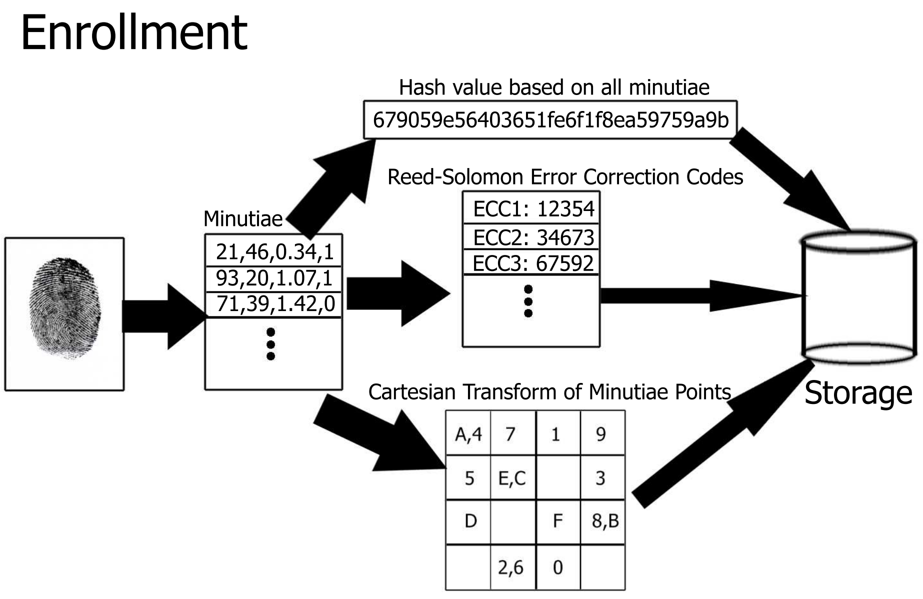
Enrollment
Hash value based on all minutiae
679059e56403651fe6f1f8ea59759a9b
Reed-Solomon Error Correction Codes
ECC1: 12354
ECC2: 34673
ECC3: 67592
Minutiae
21,46,0.34,1
93,20,1.07,1
71,39,1.42,0
Cartesian Transform of Minutiae Points
A,4 7 1 9
5 E,C 3
D F 8,B
2,6 0
Storage

Figure 3.1: Overview of the enrollment process

reflects the actual ridge structure.

There has been much discussion on fingerprint image enhancement, and a lot of effort has been devoted to the challenge of discovering enhancement algorithms that provide useful enhancements, even with limited computing resources [29, 65]. The image enhancement stage used here is based on a fairly common approach [30], which in essence filters each pixel of the fingerprint image using a Gabor filter based on estimates of the local ridge flow direction and frequency.

The image enhancement is performed in several stages. The preprocessing stage prepares the image for further analysis, before the direction of the ridge flow and the frequency of the ridge lines normal to the ridge flow direction are estimated for every pixel on the image. A mask is then created that excludes the unrecoverable regions of the fingerprint image.

The enhanced image is generated by applying a Gabor-filter, specific to the local ridge direction and frequency, to each pixel in the image. The enhanced image is subsequently binarised with a simple thresholding algorithm and thinned by elementary morphological operations until each ridge line is a single pixel wide. The resulting image, called a fingerprint skeleton, appears highly stylised, and ridge endings and bifurcations can easily be detected by traversing it looking for ridge pixels with only one neighbouring ridge pixel (ridge ending) or more than two neighbouring

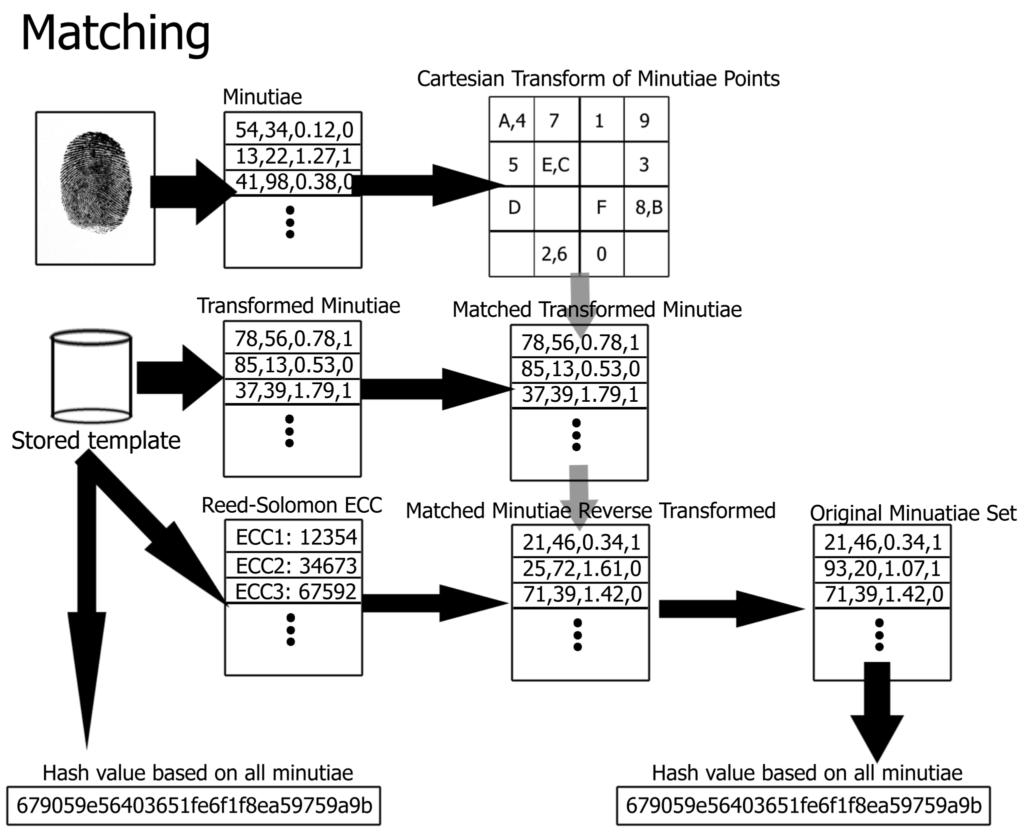

Figure 3.2: Overview of the matching process

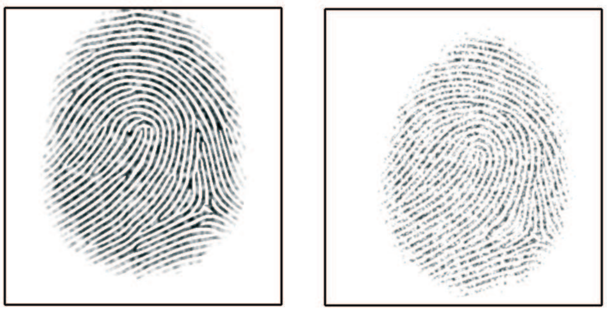

Figure 3.3: Better and worse acquisitions of a fingerprint

ridge pixels (ridge bifurcation).

A rough step-by-step overview of the fingerprint enhancement algorithm is shown in figure 3.4.

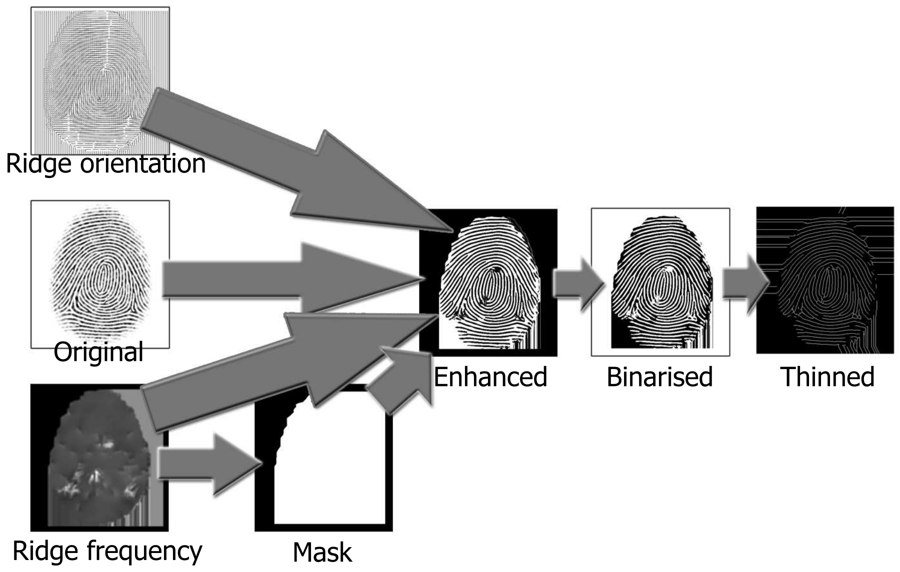

Figure 3.4: Image enhancement overview

Preprocessing

Prior to the image enhancement process, the fingerprint image is blurred slightly using a 3×3 mean filter in order to reduce the effect of rough edges, artifacts and noise. Given that $P_i(x, y)$ represents the pixel value at location (x, y) in the image, the value of the pixel at location (x, y) in the blurred image, $P_f(x, y)$, can be expressed as

$$P_f(x, y) = \frac{1}{9} \sum_{m=-1}^{1} \sum_{n=-1}^{1} P_i(x + m, y + n) \tag{3.1}$$

The result of this blurring stage is hardly noticeable, as shown in figure 3.5.

Local Ridge Orientation Estimation

The local ridge orientation is loosely defined as the direction of the ridge flow pattern of a fingerprint at one particular point. In order to determine the parameters for the Gabor filter to process a given pixel with, it is necessary to take into consideration the orientation of the ridge flow at that particular pixel.

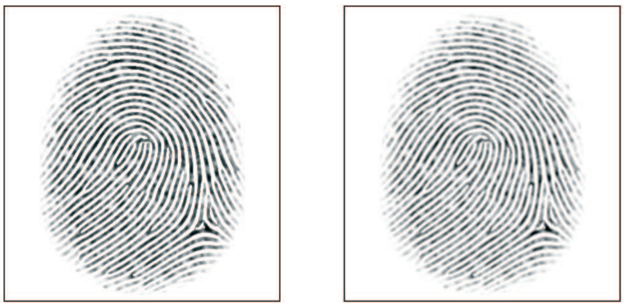
Figure 3.5: Before and after preprocessing

Estimating the local ridge orientation at each pixel is a computationally demanding task, and many different algorithms have been devised to estimate the local ridge orientation across the image. The algorithm used in this paper is largely based on a fairly common approach [33], and appears to generate quite pleasing results in a relatively modest amount of time.

The input fingerprint image is first divided into blocks of size $W \times W$. Then the gradients G_x and G_y are calculated for each pixel in the block in a simple way:

$$G_x(i,j) = B(i,j) - B(i-1,j) \tag{3.2}$$

$$G_y(i,j) = B(i,j) - B(i,j-1) \tag{3.3}$$

where $B(i,j)$ represents the value of the pixel at (i,j) in the block.

Using these gradients, the local orientation of each block is estimated using the following operations, essentially based on elementary geometry:

$$\theta = \frac{1}{2}\arctan\left(\frac{\sum_{i=1}^{W}\sum_{j=1}^{W} 2G_x(i,j)G_y(i,j)}{\sum_{i=1}^{W}\sum_{j=1}^{W} G_x(i,j)^2 + G_y(i,j)^2}\right) \tag{3.4}$$

where θ is estimate of the local ridge orientation of the block.

Noise, artifacts, high curvature, singular points and minutiae points in the image may cause this estimate to be inaccurate. Since local ridge orientation ordinarily varies slowly in a local neighbourhood, except close to singular points, a low pass filter can be used to smoothen the estimate of the local ridge orientation. In order to perform the low-pass filtering, the orientation image needs to be converted into a continuous vector field:

$$\phi_x(i,j) = cos(2 \times \theta(i,j)) \tag{3.5}$$

$$\phi_y(i,j) = sin(2 \times \theta(i,j)) \tag{3.6}$$

A low-pass filter can then be applied by simply convoluting ϕ_x and ϕ_y with a regular low-pass kernel, H. Re-calculating the angle gives the final ridge orientation image, O:

$$O(i,j) = \frac{1}{2}\arctan\left(\frac{H * \phi_y(i,j)}{H * \phi_x(i,j)}\right), \tag{3.7}$$

with $*$ denoting the convolution operator.

Using this algorithm, a fairly smooth orientation field estimate, such as that shown in figure 3.6, can be created in a reasonable amount of time.

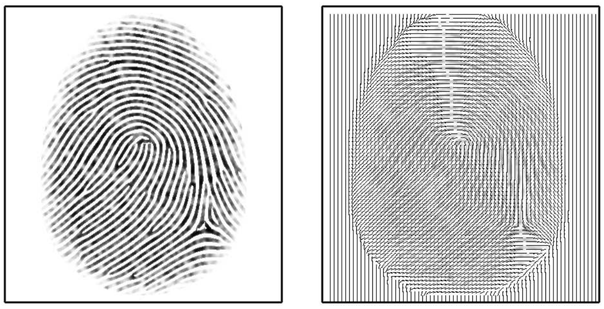

Figure 3.6: Local Ridge Orientation estimate

Local Ridge Frequency Estimation

The local ridge frequency describes the frequency of ridge lines along the direction normal to the ridge line flow, in a small neighbourhood around a given pixel. In order to create suitable Gabor filters for use on the fingerprint image, as described earlier, the local ridge frequency must be estimated for every pixel in the image.

In the local neighbourhood of a pixel, the pixel intensities along the direction normal to the local ridge orientation can be expected to exhibit a wave-like pattern, as illustrated in figure 3.7. The following procedure attempts to estimate the frequency component of that wave. The presence of, for instance, singularities and minutiae points may disrupt this pattern, so the frequency for certain blocks may have to be interpolated using the frequencies of neighbouring blocks.

First, the image, P is divided into blocks of size $W \times W$. For each block, an oriented window of size $w \times l$ is defined with the x-direction normal to the local ridge flow direction, and the y-direction along the local ridge flow direction. The signature along the x-axis of this window is then computed by summing up the intensity values of the pixels along the y-axis. If no minutiae or singular points appear in the oriented window, the

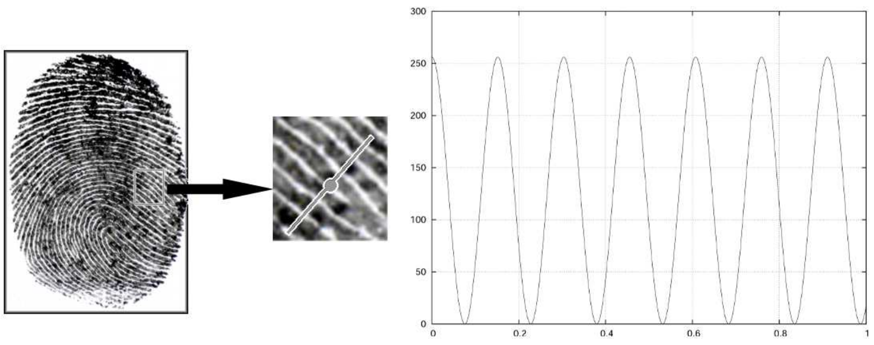

Figure 3.7: Local ridge structure exhibiting sinusoidal shape

calculated x-signature of the block should form a sinusoidal-like wave, which has the same frequency as that of the ridges and valleys in the oriented window. Therefore the signature can be used to estimate the local frequency of the ridges and valleys in the block. This process is illustrated in figure 3.8.

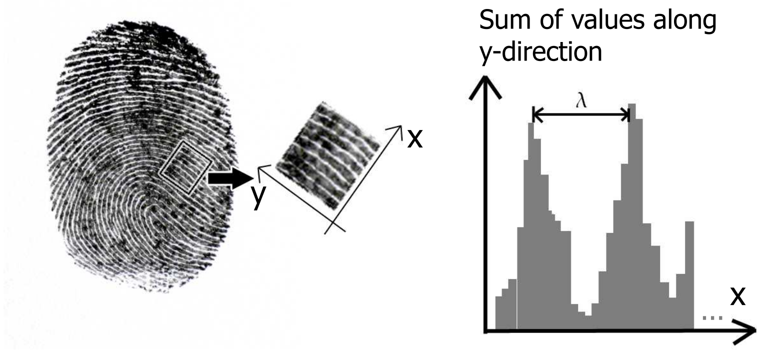

Figure 3.8: Oriented window with signature

The peaks of the signature are those values which are larger than both the left and right neighbours. The average number of pixels between consecutive peaks in the signature can be viewed as an approximation to the wavelength, λ, from which an estimate of the frequency may be easily obtained; $f = \frac{1}{\lambda}$.

However, if no consecutive peaks can be detected from the x-signature, then the frequency cannot be estimated with this method. The blocks may contain singular points, minutiae points, or other artifacts that result in difficulties for estimating the local ridge frequency. The frequencies of the concerned pixels are then found by interpolation, using those of the neighbouring estimates that were meaningful and well-defined.

Since inter-ridge distances can be expected to change slowly in a local neighbourhood, a low-pass filter can finally be used to remove potential outliers.

The result of the local frequency estimation is shown in figure 3.9, where lighter tone indicates higher frequency and darker tone indicates lower frequency.

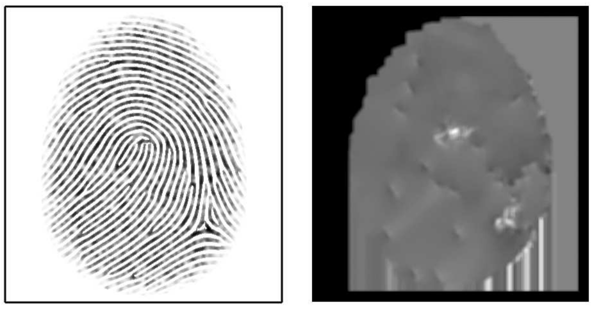

Figure 3.9: Local frequency estimates of a fingerprint

Region Masking

Fingerprint images are subject to various acquisition effects that may result in certain regions of the image being useless for recognition purposes. Some of these effects may be amended during the image enhancement process, for instance where some humidity has caused slight dilation of a ridge line, whilst other areas of the image may be entirely unrecoverable, for instance when large portions of the finger have been placed outside the sensor. The purpose of the masking process is to distinguish between the regions of the image which are good enough for using in the recognition process, and those which are not.

The image is masked using a very simple algorithm. The frequency of the ridges and valleys in a fingerprint ordinarily lies within a certain range. Therefore, if the estimated frequency is outside the expected range, the region can be considered unrecoverable. This algorithm is simple, fast, and appears quite reliable, although the actual performance may be debatable since it relies heavily on the local ridge frequency estimate. The effect of the masking process on a fingerprint image is illustrated in figure 3.10.

Bandpass and Directional Filtering

The image is then filtered using the direction and frequency fields estimated earlier in conjunction with Gabor filters, in order to obtain a greatly

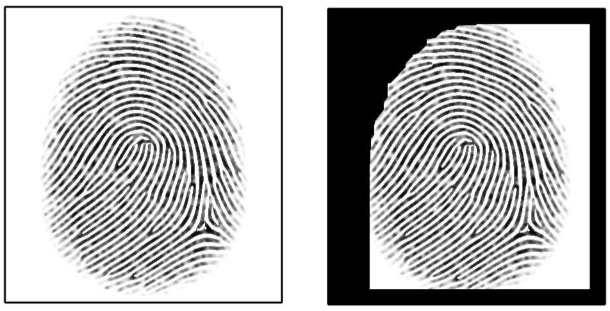

Figure 3.10: Mask of unrecoverable regions

enhanced fingerprint image. A Gabor filter combines both frequency and directional filtering, so that only effects that exhibit a certain frequency and direction will be preserved.

The Gabor filter used for a pixel (x, y) has the form

$$h(x, y; \phi, f) = \exp\left(-\frac{1}{2}|\frac{x'^2 + y'^2}{r^2}|\right)\cos(2\pi f x') \tag{3.8}$$

where ϕ is the local ridge orientation, f is the local ridge frequency, r is the radius parameter for the filter, and

$$x' = x\cos(\phi) + y\sin(\phi) \tag{3.9}$$

$$y' = -x\sin(\phi) + y\cos(\phi). \tag{3.10}$$

Some examples of such a filters with different orientations are shown in figure 3.11. The neighbourhood of a given pixel is convolved with a $q \times q$ representation of this filter, with the local value of ϕ and f, to determine the resulting intensity of the pixel.

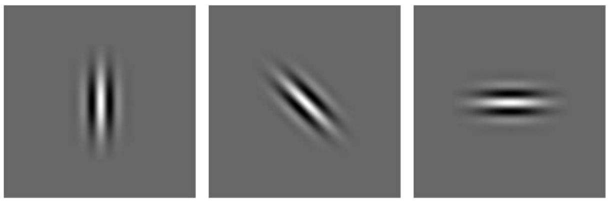

Figure 3.11: Gabor filters for three different directions

If a pixel is beneath the mask, the convolution is not performed for that pixel, and if the filter response exceeds a given threshold, the value can be clamped to a more reasonable value.

Binarisation

The resulting image is then binarised by simply comparing each pixel in the filtered image with a preselected threshold value, shown in 3.12.

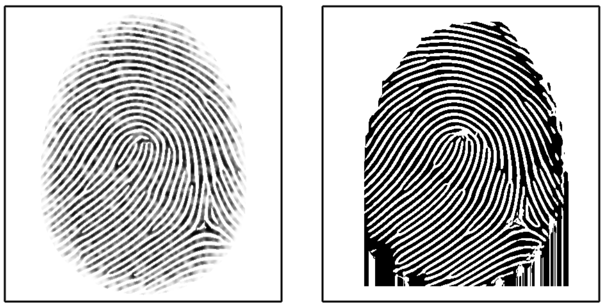

Figure 3.12: Before and after binarisation by thresholding with a value of 128

Thinning

The binarised image is then thinned using a hit and miss transform with the 3×3 structuring elements shown in 3.13. The result of the thinning process is illustrated in 3.14, where the ridge pixels are white and the ridge is only a single pixel wide.

0	0	0
x	1	x
1	1	1

x	0	0
1	1	0
x	1	x

Figure 3.13: Structure elements for thinning by hit and miss transform

3.3 Minutiae Detection

The reliability of the minutiae detection phase relies heaviliy on the quality of the image, and the performance of the image enhancement algorithms.

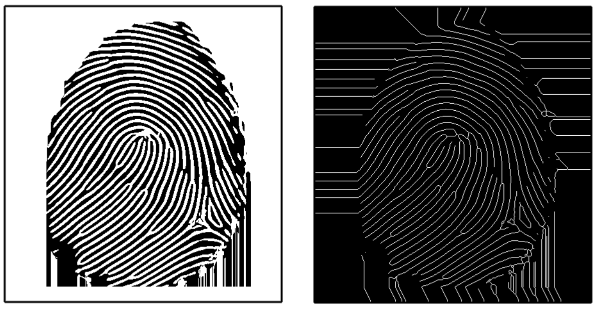

Figure 3.14: Before and after thinning by a hit and miss transform

The binarised and thinned fingerprint image, the skeleton, is traversed top-to-bottom and left-to-right to detect the minutiae points. At each pixel a collection of very simple operations are performed in order to determine whether that pixel represents a minutiae point. If the pixel is not on a ridge, it apparently cannot represent a minutiae point. If it is on a ridge, however, the number of pixels in the eight-neighbourhood which also lie on the ridge are counted. If the number of such neighbours is three or more, there is a bifurcation occurring at this pixel, and if the number of such neighbours is one, the ridge ends at this pixel. Otherwise, if the examined ridge pixel has two neighbours on the ridge, it is not considered a minutiae point. Examples of minutiae points detected by this procedure are shown in figure 3.15, with a birds-eye view shown in figure 3.16.

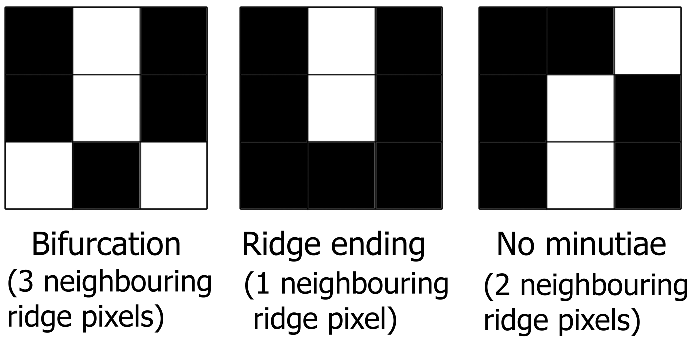
Bifurcation
(3 neighbouring ridge pixels)
Ridge ending
(1 neighbouring ridge pixel)
No minutiae
(2 neighbouring ridge pixels)

Figure 3.15: Minutiae detection by counting neighbours

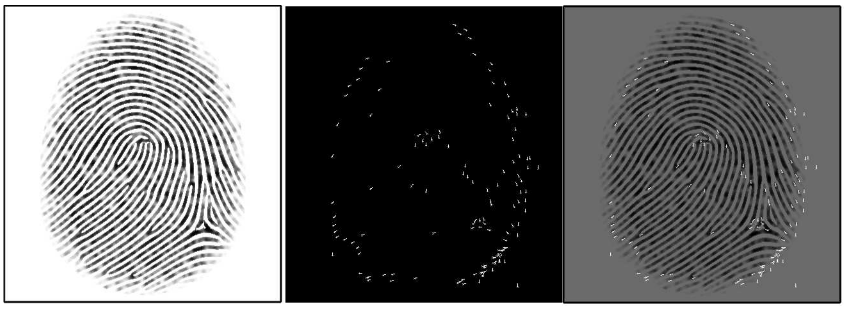

Figure 3.16: Fingerprint with the detected minutiae points marked

3.4 Core Detection

The core of the fingerprint is "the north most point of the innermost ridge line" [46]. Some examples of cores are shown in figure 3.17. If it can be detected reliably, the core can be used as a landmark for pre-aligning the fingerprint images.

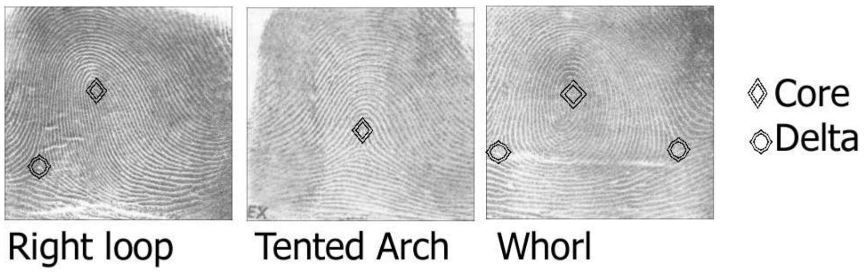

Figure 3.17: Location of core and delta in fingerprints

There have been many methods devised to detect the core [38, 19, 16], and one of the most common and elegant solutions involve using the Poincare index computed over a closed curve in the direction field, the local ridge orientation estimate, of the fingerprint [38].

The Poincare index of the pixel (i, j) is found by summing the difference between the local ridge orientation estimate for adjacent pixels in the eight-neighbourhood of (i, j). If the ridge orientation estimates of each pixel in the eight-neighbourhood of (i, j) are numbered from zero to seven in a clockwise manner, starting from the bottom left pixel, as shown in 3.18, the Poincare index can be computed as

$$\chi(i, j) = \sum_{k=0}^{7} d_{(k+1) mod 8} - d_k, \tag{3.11}$$

where d_k represents the local ridge orientation estimate in neighbour k of the pixel (i, j). Furthermore, since angles are periodic, the value of the angle d_{k+1} is taken as the interpretation which is closest to d_k.

2	3	4
1		5
0	7	6

Figure 3.18: Numbering of neighbours for Poincare index method

The Poincare index calculated over a closed curve in the direction field can only give a few discrete values, $\{0, \pi, -\pi, \pm 2\pi\}$, which mean that (i, j) represents a non-singular region, a loop, a delta, and a whorl, respectively. A case of some of these are illustrated in 3.19.

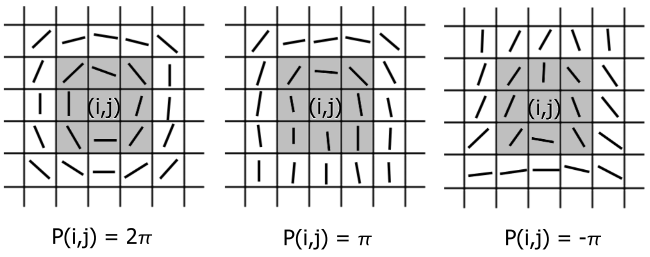

Figure 3.19: LRO in the eight-neighbourhood of (from left to right) a whorl, loop and delta

In this method, the image is traversed in a left-to-right and top-to-bottom manner, and the first detected singularity point is used as the core of the fingerprint.

Arch-type fingerprints, such as the one shown in 3.20, do not have a distinct core, and the Poincare index method may be unable to locate a stable reference point for these prints. There are, however, methods that can identify stable reference points also on this type of fingerprint [16].

In addition to a single stable reference point, such as the core found here by the Poincare index method, it is also necessary to extract a stable direction in order to align two fingerprint images. This stable direction is computed by simply averaging the local ridge orientation estimates for a block of pixels around the core.

The cores and principal directions found by this method for some fingerprints of different types is illustrated in figure 3.21.

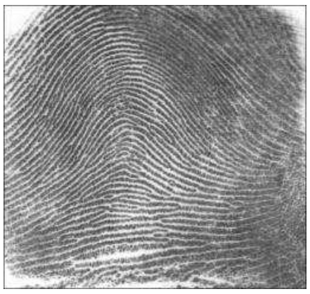

Figure 3.20: Typical arch-type fingerprint

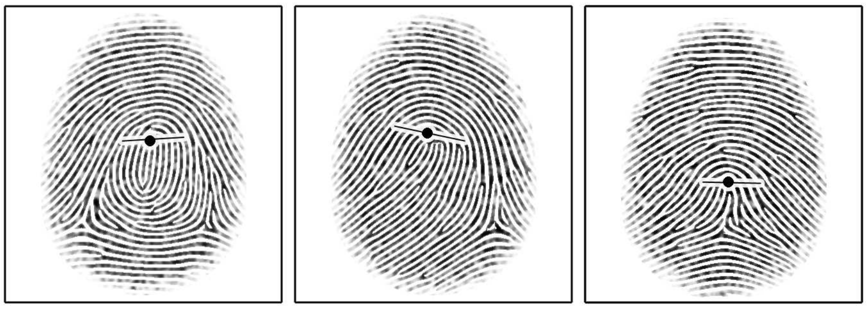

Figure 3.21: Cores and principal directions of fingerprints of types whorl, loop and tented arch

3.5 Minutiae Alignment

Having found the position of a stable reference point and a stable direction, it is possible to align two fingerprint images with each other. More interestingly, though, it is possible to align a fingerprint image to an absolute position if the reference point is taken as the origin and the principal direction is taken as the x-axis.

Rather than aligning the entire fingerprint image, however, it is beneficial for this purpose to only align the set of minutiae points given after the image enhancement and minutiae detection phase - a minutiae point at (i, j) is aligned as if the core, (c_x, c_y), was the origin and principal direction, α, was along the x-axis by

$$\begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{pmatrix} i - c_x \\ j - c_y \end{pmatrix} \tag{3.12}$$

After this operation, assuming that the core and principal direction could be estimated accurately by the process described earlier, the minutiae points that were detected in both images should be translated and rotated to roughly the same coordinates, disregarding skin elasticity effects. This process is illustrated in figure 3.22.

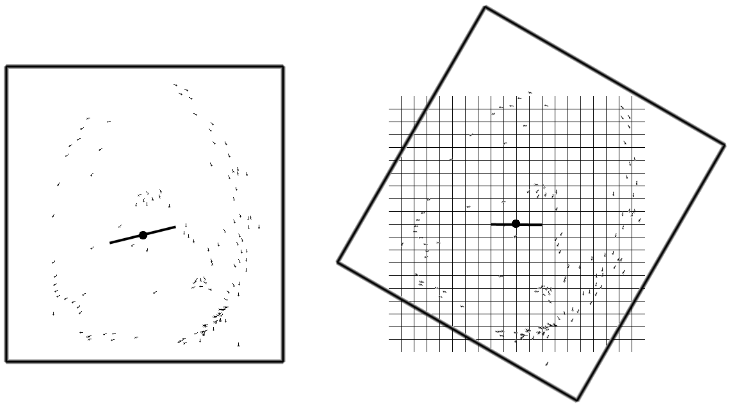

Figure 3.22: Alignment of minutiae points

3.6 Hashing

Upon enrollment, a hash value is calculated for the set of detected and aligned minutiae points, using the Message-Digest algorithm 5 [63] on the position, the angle and the type of each minutiae point. The hash value is intended to represent the set of minutiae points uniquely, and should be prohibitively difficult to reproduce if the exact specifications of the minutiae points are unknown.

Taking into account all the effects that affect the acquisition and minutiae extraction processes - rotation, translation, nonlinear deformation, noise, scratches, humidity, etc. - the probability of the hash value for the template being the same for two different acquisitions, even of the same fingerprint, is negligible. The hash value for the entire template is stored in the template file during enrollment, for reference during the matching process.

3.7 Reed-Solomon Error Correction

Reed-Solomon error correction is an error correction code that works by oversampling a polynomial constructed from the data, in this case the set of minutiae points. Since the polynomial is oversampled, and hence overdetermined, it is possible to recreate the entire data set even if some of the values are missing or corrupt. If there is so much data missing that the polynomial is undetermined, the original data set cannot be recovered [62].

The set of aligned minutiae points are processed using Reed-Solomon coding, so that the entire set of minutiae points may be recovered if only a given number of minutiae points are known exactly. The reason for this is that the matching process may only recover a subset of the given minutiae points that were enrolled, and in order to generate the hash value for the entire set of minutiae points, as was done during enrollment, the whole set of minutiae points are required. Once the entire original set of minutiae points from the master fingerprint is recovered, it is possible to re-calculate the hash value, which, in the case of a match, should be the same as that stored in the template file during enrollment.

By engineering the Reed-Solomon coding process on the set of aligned minutiae points, it is possible to guarantee that the entire set of minutiae points can be recovered if we know only m of them - in addition to the checksums generated during the Reed-Solomon coding process.

A Reed-Solomon code is determined by two parameters, n and k, where n is the total number of symbols in a block, and k is the number of data points. The entire set of minutiae points are encoded as one such a block, with each minutiae point being considered one symbol or data point.

Each minutiae point is represented by three floating point numbers, (x, y) coordinate and an orientation θ, and an integer, t denoting the minutiae point type. For simplicity, all these members are assumed to be of 32 bit length, and a single minutiae point is thus represented by $4 \times 32 bits = 128 bits$. Hence the length of a symbol used in this instance of Reed-Solomon error correction codes should be 128 bits.

The number of errors that can be corrected in a block is given by $\frac{n-k}{2}$. The number of data points is the total number of minutiae points in the set. This determines the parameter k for the Reed-Solomon coding process. Since the entire data set should be reconstructable from m minutiae points, the number of erroneous data points the code must be able to correct is $k - m$. This enables us to calculate the total number of symbols required in the block. Setting $n = 2(k - m) + k$, where k is the number of minutiae points in the entire original set and m is the number of minutiae points that must be known in order to recreate the set by Reed-Solomon decoding, allows the error correction code to recreate the entire set of minutiae points if only m are known.

The block contains $n = 2(k - m) + k = 3k - 2m$ symbols in total, where k is the number of minutiae points in the set, and m is the number of minutiae points required to regenerate the entire set. If each minutiae point is represented by 128 bits, the total length of the block becomes $(3k - 2m) \times 128 bits$.

The error correction codes for the set of minutiae points is generated in an encoding phase during enrollment, and are then stored in the template file so they may be used during the matching process. The original set of minutiae points, however, are subjected to a Cartesian transformation before being stored, so only the part of the block containing the error correction codes are stored from this stage.

3.8 Cartesian Transformation of the Feature Set

After Reed-Solomon error correction codes have been created for the entire aligned minutiae set, the set of minutiae points will undergo a particular transformation before being stored. The intent of this transformation is primarily to conceal the original locations of the minutiae points. However, the transformation is required to still allow reliable fingerprint comparison to take place in the transformed space, and the matching process must provide enough information to reverse the transformation for the matched minutiae.

Characteristics of the Transformation

The anatomy of this transformation is crucial to the security of the method. One way to conceal the original locations of the minutiae points, is to ensure that the transformation is non-injective. Furthermore, given both the transformation and a transformed minutiae point, the original location of the minutiae point should still not be obvious. The larger number of possible original locations for a minutiae point under a specific transformation, the more difficult it will be for an attacker to actually guess the original location of the minutiae point, and hence the template will be more secure. So not only is it a requirement that the transformation is non-injective, it is required that each transformed minutiae point may have originated in a large number of possible original minutiae locations.

The next requirement for this transformation is that it must allow fingerprint comparison in the transformed feature space. This entails that fingerprints that are close matches in the untransformed space, should also be close matches in the transformed space. Features from the two prints which are close together before the transformation, should also be close after the transformation. By performing the comparison in the transformed feature space, the matching feature should enable the program to select the original minutiae point from the list of possible minutiae points whence it came.

A simple example will highlight the desired characteristics of this transformation.

Imagine that a fingerprint feature, λ_1, is transformed by such a transformation, f: $f(\lambda_1) = \Lambda$.

The first requirement for the transformation is that there are other features which also map to the same output location: $f(\lambda_i) = \Lambda, i = 2, ..., n$, so that someone who knows Λ and f will still not know which of the $\lambda_i, i = 1, ..., n$ represents the original feature.

Now imagine that feature matching is represented by an equivalence relation $\sim$, and that the corresponding feature from a matching print, $\gamma_1 \sim \lambda_1$, is also transformed by the same transformation: $f(\gamma_1) = \Gamma$.

The second requirement concerns matching in the transformed space: the transformation f behaves such that if $\lambda \sim \gamma$, then $f(\lambda) \sim f(\gamma)$.

Knowing that $\Lambda \sim \Gamma$, $f(\gamma_1) = \Gamma$ and $\lambda_1 \sim \gamma_1$ makes it possible to select λ_1 as the original feature from the set $\lambda_i, i = 1, ..., n$.

Thus a transformation with these characteristics is hardly reversible, unless presented with a matching fingerprint.

Cartesian Block Transformation

A Cartesian block transformation of the space of the fingerprint feature set is a relatively simple and straight-forward operation [58]: the Cartesian coordinate system in which the aligned minutiae points reside is divided into blocks of regular size, whose locations are then shuffled - thus shuffling all the minutiae points around. If several blocks in the input can be mapped to a single block in the output, each output block is the result of a many-to-one transformation, and the original locations of the minutiae points cannot be known, in principle, even if the transformation itself is known. A simple illustration of this is shown in figure 3.23.

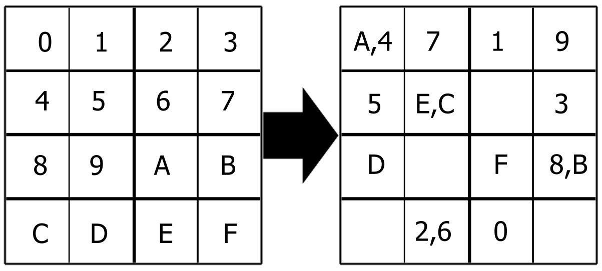

Figure 3.23: A Cartesian transformation shuffles the blocks

This transformation does not entirely fulfill the requirements discussed earlier. It is indeed non-injective, but it does not ensure that all transformed minutiae points have several possible origins. The transformation also only to a certain degree ensures that points close to one another will be transformed in a similar manner. Points that are close to the edge of a block may fall into different blocks for different acquisitions, and thus points that may have matched in the untransformed space may conceivably be transformed so that they do not.

Despite these shortcomings, however, this transformation is a well-known method for generating cancellable fingerprint templates, and displays a certain degree of the desired qualities. Although it is far from perfect for this application, it is conceptually simple and it may still serve well for illustrative purposes. It may be swapped for a more appropriate transformation in the future, adhering more strongly to the discussed principles, but will suffice at the present time.

During enrollment, the transformed minutiae points are stored in the template, along with the parameters used for the transformation. This should keep the original positions of the minutiae points hidden, whilst enabling minutiae sets from subsequent acquisitions to undergo the same transformation during the matching process.

3.9 The Fingerprint Template

At the end of fingerprint enrollment, the template includes the following data:

- Transformed minutiae points: (x, y, θ) and type.
- Cartesian block transformation parameters

- Hash value of the pre-transform aligned minutiae set
- Reed-Solomon error correction codes

This template does not reveal the actual whereabouts of the minutiae points in the fingerprint, and each enrollment will, due to minute changes in the set of minutiae points, generate an entirely different template. Revoking and reissuing a template is thus simply a matter of acquiring another fingerprint image and generating a new template using the same process, though alternatively also changing the transformation parameters.

3.10 Matching

The matching process loads the template stored during enrollment and a candidate fingerprint image, and attempts to determine whether or not they represent the same fingerprint.

The candidate fingerprint image is first enhanced, and the minutiae points are extracted and aligned to the reference coordinate system, as explained earlier. The Cartesian transformation that was performed on the minutiae points of the enrolled fingerprint, is repeated on the minutiae points of the candidate print. Having stored the transformation parameters in the template file, this process is trivial.

A standard matching algorithm, essentially checking whether minutiae points from each of the sets are closer to each other than a given threshold, is performed in the transformed feature space. If two minutiae points match, the source block of the minutiae point in the stored template, i.e. the block it was in before the Cartesian block transformation, is assumed to be the same as that of the minutiae point from the candidate fingerprint. This means that the Cartesian transform that was performed on the minutiae points of the enrolled fingerprint can be reversed for the matching minutiae points.

When at least m such minutiae points have been matched correctly and the Cartesian block transformation on these has been reversed, they can be combined with the Reed-Solomon error correction codes, so that the entire set of minutiae points from the enrolled fingerprint prior to transformation can be obtained. The entire pre-transformed set of minutiae points from the enrolled image can then be used to calculate a hash value, as during enrollment, which should be exactly the same as the one stored in the template.

If the hash value generated by this process matches the hash value that was stored in the template during enrollment, the fingerprints are considered mates. An overview of the matching process is given in figure 3.24.

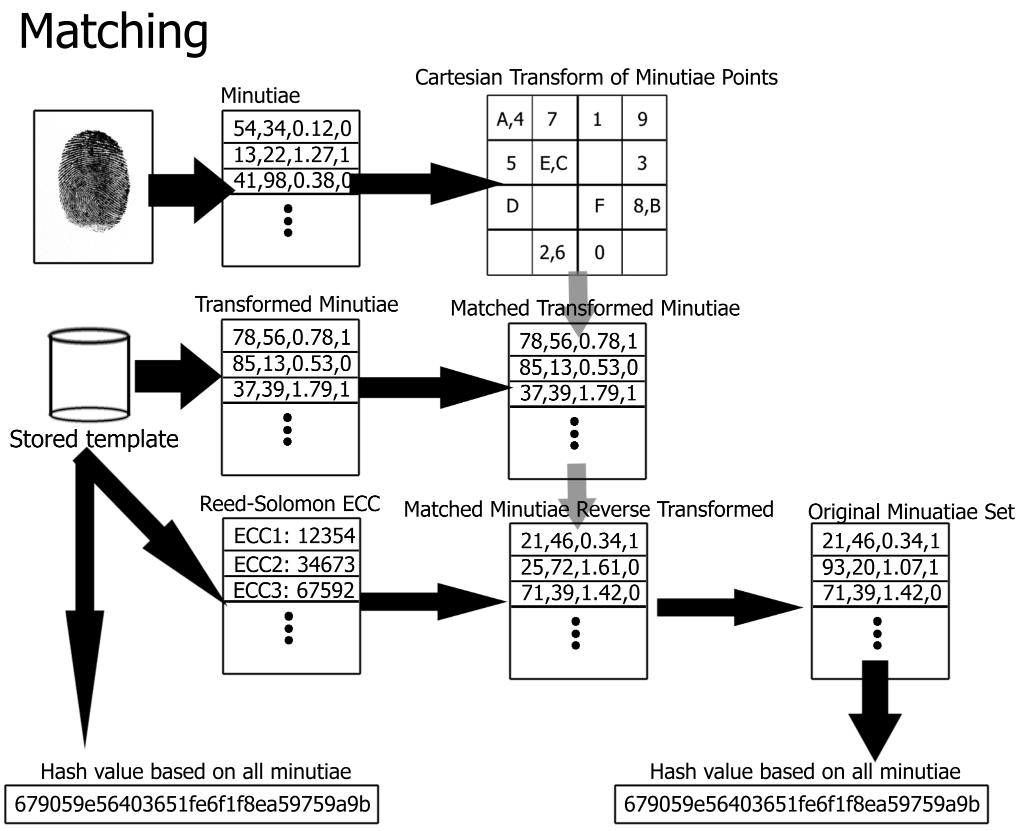

Figure 3.24: Overview of the matching process

Chapter 4

Implementation

4.1 Overview

A sample implementation of the method was created in order to verify and explore the concepts presented in this thesis. The implementation is intended only as a proof-of-concept.

Two separate programs were created: one for enrolling fingerprint images and one for matching the enrolled fingerprint images with images of candidate fingerprints.

The method was implemented using a combination of the programming languages C and Python. The C programming language is known to perform well for computationally demanding tasks, and the Python programming language is largely known for its convenience and ease of use. C was used for computationally demanding tasks, such as image enhancement, and Python was used for control and flexibility where the inferior computational speed of Python was not found prohibiting.

After the feature extraction stage, the aligned minutiae set is stored in a file for future reference, since this is a very time-consuming process. This process really only needs to be performed once for every image, and since the testing process will encounter an image several times during testing, this simple technique was used to reduce the time spent on testing.

Although the description of the implementation given here is fairly complete, it only covers the most crucial parts, and some details have knowingly been omitted in the name of brevity.

4.2 Enrollment

The enrollment program accepts a fingerprint image as input and outputs a fingerprint template of the type described in this thesis - containing minutiae points that have been transformed with a Cartesian block transformation, the transformation parameteres used, a hash value calculated from the untransformed minutiae set and Reed-Solomon error correction codes.

The different stages of the enrollment program are, in order:

- Image enhancement and feature extraction
- Core detection and principal direction estimation
- Minutiae alignment
- Minutiae set hashing
- Reed-Solomon error correction coding
- Cartesian block transformation of minutiae points

Image Enhancement and Feature Detection

The implementation used the *Fingerprint Verification System* (FVS) software libraries for performing image enhancement and minutiae detection [2]. This library implements the image enhancement methods presented in this thesis, which are largely based on the approaches in [29] and [30]. It also provides necessary data structures for manipulating images and fingerprint data.

The FVS libraries are written in C, and using them for the enhancement and minutiae detection algorithms presented in this paper, assuming the variables used are already defined, requires only a few essential lines of code:

```
/* Preprocessing - apply blur filter */
ImageSoftenMean(image, 3);
/* Estimate local ridge orientation */
FingerprintGetDirection(image, direction, 5, 8);
/* Estimate local ridge frequency */
FingerprintGetFrequency(image, direction, frequency);
/* Create mask of unrecoverable regions */
FingerprintGetMask(image, direction, frequency, mask);
/* Gabor filter each pixel */
ImageEnhanceGabor(image, direction, frequency, mask, 4.0);
/* Binarise by thresholding */
ImageBinarize(image, (FvsByte_t)0x80);
/* Perform thinning */
ImageThinHitMiss(image);
/* Find minutiae points */
MinutiaSetExtract(minutia, image, direction, mask);
```

This code extracts all the minutiae points detected in the fingerprint image into a specialised data structure for minutiae points, and may be used further in the program.

Core Detection and Principal Direction Estimation

Using the local ridge orientation estimtate created in the image enhancement phase, detecting the core with the Poincare index method is a simple matter of traversing the direction field looking for locations where the sum of the differences of the direction estimates in the eight-neighbourhood is close to π:

```
for(i = 1; i < height; i++) {
  for(j = 1; j < width; j++) {
    d0 = GetFFVal(direction, i+1, j-1,0);
    d1 = GetFFVal(direction, i, j-1, d0);
    d2 = GetFFVal(direction, i-1, j-1, d1);
    d3 = GetFFVal(direction, i-1, j, d2);
    d4 = GetFFVal(direction, i-1, j+1, d3);
    d5 = GetFFVal(direction, i, j+1, d4);
    d6 = GetFFVal(direction, i+1, j+1, d5);
    d7 = GetFFVal(direction, i+1, j, d6);
    d8 = GetFFVal(direction, i+1, j-1, d7);
    pind =
(d1-d0)+(d2-d1)+(d3-d2)+(d4-d5)+(d6-d5)+(d7-d6)+(d8-d1);

    if (abs(PI - pind) < eps) {
      corey = i; corex = j;
    }
  }
}
```

where the `GetFFVal(dirfield, i, j, dir)` function returns the angle at point `(i,j)` of the direction field `dirfield`, taken as the interpretation which is closest to the angle `dir`.

The principal direction is then estimated by averaging the angles in a small neighbourhood around the located core:

```
float dir = 0.0;
for(m = corex - 2; m < corex - 3; m++)
  for(n = corey - 3; n < corey - 1; n++)
    dir += FloatFieldGetValue(direction, m, n);
dir /= 20.0;
```

Minutiae Alignment

Alignment of the minutiae points is performed by iterating through each minutiae point that was found during feature extraction and performing the rotation and translation according to the detected core, as follows:

```
for(i = 0; i < numminutiaes; i++) {
  x = minutia[i].x - corex;
  y = minutia[i].y - corex;
  minutia[i].x = x*cos(dir) - y*sin(dir)
  minutia[i].y = x*sin(dir) + y*cos(dir)
}
```

After this operation, the minutiae points have been aligned with the core as origin and with the estimated principal direction along the x-axis. The list of minutiae points is then written to a temporary file, ready to be used by the Python part of the enrollment program, which will process the data further.

Minutiae Set Hashing

The file with the aligned minutiae set is then read into the Python program, where it is trivial to calculate an MD5 hash value for the string from the file using the built-in `md5` module:

```
import md5
minutiae_set_hash = md5.new(filecontents).hexdigest()
```

Using the hexadecimal representation, given by the `hexdigest` function, makes the calculated MD5 hash value a little easier on the eyes during inspection.

Reed-Solomon Error Correction Coding

The *Reed-Solomon Python Extension Module* is used for the Reed-Solomon coding [4], and is based on Phil Karn's *DSP and FEC Library* [37].

If `minutiaelist` is a list containing one entry for each of the 128-bit representations of the minutiae points, using the extension module to create the Reed-Solomon error correction codes is fairly trivial:

```
import reedsolomon as rs

# Setting the RS coding parameters
k = len(minutiaelist)
n = 3*k - 2*m

c = rs.Codec(n, k)
encoded = c.encodechunks(minutiaelist)
checksums = encoded[k:]
```

The list `checksums` will now contain only the Reed-Solomon checksums, which subsequently will be stored in the template file. The value for `m`, the number of minutiae points required to regenerate the entire list at the time of matching, was chosen to be eight, as eight matching minutiae points represents a reasonable match.

Cartesian Block Transformation of Minutiae Points

The implementation of the Cartesian transformation first finds a bounding box for the set of minutiae points. The bounding box is then divided into a 10×10 regular grid.

In this implementation, the Cartesian block transformation was simplified to a one-to-one transformation rather than a many-to-one transformation. This was done for simplicity, as this allows the implementation to create a list of block numbers and simply shuffle that list. Although this allows a potential attacker to easily reverse the transformation, and is thus a transgression of one the core principles in the presented method, this is simply a proof-of-concept implementation, and this transgression will be permitted. The implementation will still suffice to illuminate the key principles of the method, even with this deficiency.

Next, the position of each minutiae point is translated to its new block. Subsequently, the list of transformed minutiae points will be ready for saving to the template file, along with the transformation parameteres - namely `blocks` and the bounding box, which describes the mapping in its entirety.

```
# Find extremities of bounding box
xmax = max(minutiaexpositions)
ymax = max(minutiaeypositions)
xmin = min(minutiaexpositions)
ymin = min(minutiaeypositions)

# Find the step lengths for the grid
dx = (xmax-xmin)/10
dy = (ymax-ymin)/10

# Create the transformation by shuffling numbered blocks
blocks = range(0,10*10)
random.shuffle(blocks)

for i in range(minutiaexpositions):
  # Find which block this minutiae point is in
  x = minutiaexpositions[i]
  y = minutiaeypositions[i]
  blocknum = int((x-xmin)/dx)+10*int((y-ymin)/dy)

  # Find which block to transport it to, and do it
  newblock = blocks[blocknum]
  minutiaexpositions[i] = x+dx*(newblock%10-int((x-xmin)/dx))
  minutiaeypositions[i] = y+dy*(newblock/10-int((y-ymin)/dy))
```

At the end of this stage, the transformed minutiae points and the transformation parameters are ready to be stored in the template file. The transformation parameters that are stored are the extremities of the bounding box, and the list of block locations.

Template File

After these stages have been performed, the fingerprint template is stored in a file. The template includes the following data:

- Transformed minutiae points: (x, y, θ) and type.
- Cartesian block transformation parameters, specifically the shuffled list of blocks and the corners of the bounding box.
- Hash value of the pre-transform aligned minutiae set.
- Reed-Solomon error correction codes.

4.3 Matching

The matching program accepts a fingerprint template, of the format output by the enrollment program, and a fingerprint image as input. The output from the program is either a zero, indicating that the template and image do not represent the same fingerprint, or a one, indicating that the template and image do represent the same fingerprint.

The different stages of the matching program are, in order:

- Image enhancement and feature extraction
- Core detection and principal direction estimation
- Minutiae alignment
- Cartesian block transformation of minutiae points
- Minutiae matching
- Cartesian block transformation reversal
- Reed-Solomon decoding
- Minutiae set hashing

The image enhancement, feature extraction, core detection, principal direction estimation, minutiae alignement and Cartesian blocks transformation of the minutiae points are performed exactly as during enrollment, except that the array `blocks` and the bounding box used for Cartesian block transformation is read from the template file instead of randomly generated, as during enrollment.

Minutiae Matching

When the minutiae points of the candidate fingerprint have been extracted, aligned and transformed, they are matched against the set of minutiae points stored in the template file. As the minutiae sets are both pre-aligned using a common reference point and direction, then subsequently transformed with the same transformation, the minutiae points which match each other in the sets should presumably be spatially close to each other.

In this implementation, two minutiae points are considered a match if they are closer than 15 pixels. This constraint appears to be quite relaxed - it will cause many matches, a portion of which will be true and a portion of which will be false. However, as long as the number of correct matches is great enough to perform the Reed-Solomon decoding, it makes no difference how many false matches there are. Therefore even a potentially high ratio of falsely matched minutiae points causes little concern for this method and can be tolerated. A similarly relaxed constraint applies to the angles of the minutiae points.

The matching is performed by simply comparing the position and angle each minutiae point from the template to the positions and angles of the minutiae points from the candidate image:

```
matches = []
for p in minutiaelist1:
  for q in minutiaelist2:
    if abs(p.x - q.x) < 15:
      if abs(p.y - q.y) < 15:
        if abs(p.angle - q.angle) < 0.3:
          p.destinationblock =
q.sourceblock
          matches.append(p)
          break
```

After this procedure, the list `matches` will contain a list of the minutiae points from the template that matched minutiae points from the candidate image. In addition, the block whence the minutiae point in the candidate image came is set as a destination block for the matched minutiae point, so that the Cartesian block transformation may easily be reversed for it. This assumes that the variable `q.sourceblock` was set for the minutiae points in the candidate image when the transformation was performed.

Cartesian Transformation Reversal

Having a list of transformed minutiae points with their original block attached to each one, as created in the last step, reversing the Cartesian transformation is relatively simple. For each matching minutiae point, m, the following procedure reverses the Cartesian block transform:

```
m.x = xmin + dx*(m.destinationblock%10)
m.y = ymin + dy*(m.destinationblock/10)
```

xmin, ymin, dx and dy are all easily available, as the extremities of the bounding box was stored during enrollment.

After performing this procedure on each matching minutiae point from the fingerprint template, the matching minutiae points should all have recovered their original location from before the transformation was performed during enrollment.

Reed-Solomon Decoding

Having regained the data of some of the original minutiae points from the enrolled template, it is now possible to recreate the entire pre-transform enrolled minutiae set using the Reed-Solomon error correction codes, provided enough minutiae points were reversed correctly.

Using the same Python extension module for the decoding as was used for the encoding, the decoding is very simple:

```
k = len(templateminutiaelist)
n = 3*k - 2*m
c = rs.Codec(n,k)
originalminutiaelist = c.decodechunks(matchedlist + checksums)[0]
```

This piece of code assumes that templateminutiaelist is the list of minutiae points read from the template file, that matchedlist contains the set of minutiae points that matched and whose transformation was reversed, and that checksums is the list of Reed-Solomon error correction codes that were generated during enrollment and stored in the template file. The variable m contains the same number as during the error correction coding at enrollment, namely eight.

At the end of this phase, the list originalminutiaelist contains the entire set of aligned minutiae points that were located in the enrolled fingerprint.

Minutiae set hashing

Having recreated the entire set of aligned minutiae points, it is now possible to calculate the hash value for it in exactly the same manner as during enrollment.

If the hash value that was saved in the template file during enrollment is the same as the hash value generated by this process during matching, the fingerprints can be considered mates. The hash value thus created is therefore a unique number for this fingerprint template, and the same number can be created during matching as was created during enrollment.

4.4 Run-time Parameters

Since a lot of the complexity of the implementation is hidden due to the use of pre-built software libraries, certain settings and parameters may not be entirely obvious from the discussion earlier, and deserve a small remark.

The region masking process removes areas where the local ridge frequency is lower than $\frac{1}{25}$ or higher than $\frac{1}{3}$, based on empirical estimates for 500dpi fingerprint images.

Binarisation of the enhanced image is performed by a simple thresholding process, and the value used as a threshold in this process was 128.

Eight minutiae points are used as the limit for the Reed-Solomon process, since eight correctly matching minutiae points can be considered a reasonable threshold for a match.

Although the description of the implementation given here is fairly complete, some details have knowingly been omitted in the name of brevity.

Chapter 5

Testing Procedures

5.1 System Accuracy Testing

The accuracy of the devised recognition system will be estimated by running the implemented program on the fingerprint databases used in the Fingerprint Verification Competition (FVC) 2000 [44]. The FVC is a bi-annual event where contestants - industrial, academic and independent - submit fingerprint recognition systems, whose accuracies are thoroughly tested, and the systems with the best recognition accuracies are awarded.

The FVC constructs four databases for each competition. Each database consists of 100 fingers, with eight impressions of each finger, resulting in a total of 800 fingerprint images per database. Three of the databases are collected using different types of sensors, and one of the databases is synthetically generated [5].

The figures that will be reported from this testing process follow the same testing protocol as in the FVC, such that a meaningful foundation for comparison of system performance with the participants of the competitions will be immediately available.

Performance evaluation of the fingerprint systems is performed by first matching each fingerprint image against the remaining images of the same finger, avoiding symmetric matches - if print a has already been matched against print b, then b is not matched against a - in order to avoid correlation. The rate at which the matching process rejects these fingerprints, which should match, provides an estimate of the False Reject Rate (FRR), or the False Non-Match Rate (FNMR).

For a database containing 100 fingers with 8 impressions of each finger, the number of false reject tests performed using this testing procedure is $\frac{8\times7}{2} \times 100 = 2800$. Having four such databases altogether, the number of false reject tests reaches a total of $4 \times 2800 = 11200$, which is a reasonable amount of tests from which to estimate the FRR.

Secondly, the first fingerprint image of each finger is matched against the first fingerprint image of every other finger in the database, still avoiding symmetric matches. The rate at which these fingerprints, which should not match, are accepted by the recognition system provides an estimate of the False Accept Rate (FAR), or the False Match Rate (FMR).

For a database containing 100 fingerprints with 8 impressions of each finger, the number of false accept tests performed using this testing procedure is $\frac{100 \times 99}{2} = 4950$. With four such databases, the total number of false accept tests reaches $4 \times 4950 = 19800$, which provides a reasonable amount of tests from which to estimate the FAR.

5.2 Evaluation Criteria

Does it work at all?

The first point of consideration is whether or not the algorithm presented here is at all able to distinguish between matching and non-matching fingerprints.

In order to establish the validity of the method, the Genuine Accept Rate (GAR) is compared to the False Accept Rate. A Genuine Accept is when the algorithm correctly decides that two fingerprint imprints come from the same finger. The rate at which this occurs is termed the Genuine Accept Rate, and is easily calculated using the False Reject Rate: $GAR = 1 - FRR$.

If the algorithm indeed does work, the GAR should turn out to be higher than the FAR. If the FAR and GAR are equal, it indicates that the system would be equally likely to accept an impostor as a genuine user. If the GAR is higher than the FAR, the system is more likely to accept a genuine user than an impostor, and the system is thus validated.

Performing the two-sample t-test with the null hypothesis stating that the GAR and FAR are equal will give an indication of whether the differences found in the GAR and FAR are statistically significant. If this null hypothesis can be rejected, the GAR and FAR are likely to be unequal, thus in principle validating the algorithm.

What causes the system errors?

The second point of consideration is what portion of the error rate can be attributed to which part of the system. Specifically, it is of great interest how the novelties of this method, namely the Cartesian block transformation, the inversion of it, and the Reed-Solomon coding, affect the performance of the more traditional algorithm that is used as a basis - the feature extraction and matching algorithm.

Therefore, the method presented here is not only compared to other recognition algorithms that have participated in the FVC, but is also compared to a reduced version of itself with all the novelties stripped out. This simplified version uses the same image enhancement, feature extraction and matching algorithm as the full method, but does not perform the Cartesian block transformation or the Reed-Solomon coding.

Comparing the performance of the full algorithm with this simplified algorithm will give an idea of how the novelties of the procedure in this thesis affect recognition accuracy, and what part of the errors can be attributed to deficiencies in the underlying feature extraction and matching algorithms, and what portion of the errors must be attributed to, primarily, the Cartesian transformation.

How does it compare to state-of-the-art recognition methods?

The third point of consideration is how the method compares to other modern recognition algorithms. Luckily, since the fingerprint databases and testing protocol from the FVC 2000 are used for performance estimation, the results are in a format which can be easily compared to fingerprint recognition algorithms that participated in the FVC 2000.

Special care must be taken during this comparison, because whereas the output from conventional recognition procedures is usually a score between zero and one indicating the strength of the match, the output from the algorithm presented herein is strictly binary; either the fingerprints match, or they do not. The Receiver Operator Curve (ROC) is a curve showing the FAR on the vertical axis and the FRR on the horizontal axis as a threshold t for considering two subjects a match varies, and is the de facto method of reporting recognition performance for fingerprint recognition systems. Although a similar parametrisation to the t-threshold of traditional matchers is conceivable, constructing an ROC for the particular method presented here is currently meaningless. This algorithm can therefore be compared to traditional thresholded algorithms only at specific operating points.

The operating point that will be used for comparison is the Equal Error Rate - the point at which the FAR and FRR are equal.

Since the FVC is a fairly prestigious competition, there is no reason to believe that the algorithm presented here will offer much competition when it comes to recognition performance. This, however, was never the intention of the algorithm either - first of all, recognition performance is only a secondary concern of this algorithm, and, secondly, the implementation that is tested can only be considered a crude proof-of-concept. Thus, the performance degradation due to supplementing a standard recognition method with the particular additional steps described here is a much more interesting measure.

5.3 Shortcomings of the Testing Procedure

The testing procedure used here, and indeed in most studies of fingerprint recognition systems, effectively simulate a zero-effort attack: the impostor is making no effort whatsoever to circumvent the system, except presenting his own fingerprint and hoping the software will fail to reject his fingerprint. As such, the testing procedure only estimates recognition accuracy, not system security per se. Other types of attacks are complicated to simulate, and are beyond the scope of this thesis.

However, a simple model estimating the feasibility of regaining the original minutiae set from the stored template is presented and discussed. This will provide some insight into the characteristics that arise when combining Reed-Solomon error correction coding and the Cartesian block transformation.

Nor does the testing procedure reflect the increased privacy and security, which are the main concern of this algorithm. Although the performance figures may turn out to be sub-par, the method may still be useful because it adds a different dimension of security and privacy that is not normally, or easily, evaluated quantitatively for fingerprint recognition algorithms.

Chapter 6

Results and Analysis

6.1 Experimental Results

Table 6.1 summarises the results from the experiments. In the table, the caption New refers to the method explored in this thesis, Simpl refers to the simplified method on which the new method is based - with the Reed-Solomon coding and Cartesian block transformation left out - and FVCM and FVCW refer to the median entry and winning entry of the Fingerprint Verification Competition, respectively. The False Reject Rate (FRR) and False Accept Rate (FAR), measuring the recognition accuracy, are given for each of the recognition algorithms. For the entries in the FVC, a specific operating point has been chosen, namely the point of Equal Error Rate (EER), since recognition algorithms generally output a number between zero and one indicating the strength of the match, and not a binary result, as the algorithm discussed in this paper does.

Does it work at all?

Considering the new method separately, it appears to exhibit an alarmingly high False Reject Rate for all the four databases, and a relatively modest False Accept Rate. With an average False Reject Rate over the

	New		Simpl		FVCM		FVCW	
	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR
DB1	94%	5.5%	41%	52%	10.66%	10.66%	0.67%	0.67%
DB2	96%	3.7%	19%	77%	8.83%	8.83%	0.61%	0.61%
DB3	100%	0%	3.3%	92%	12.20%	12.20%	3.64%	3.64%
DB4	74%	17%	31%	50%	12.08%	12.08%	1.99%	1.99%
Avg.	91%	6.5%	24%	68%	10.94%	10.94%	1.73%	1.73%

Table 6.1: Results from testing on the FVC2000 datasets

four databases of 91%, the current implementation is useless for any practical application as is. The False Accept Rate appears more reasonable, though still higher than can be accepted for most practical applications, with an average over the four databases of 6.5%.

Looking closer at the results, all the images in the third test database, DB3, appear to have been rejected by the new algorithm. This may indicate that there might be a problem with the format of the images in the database or the parameters used in the image enhancement stage that causes the algorithm to categorically reject images of this format.

The Genuine Accept Rate (GAR), $GAR = 1 - FRR$, for the four databases are 6%, 4%, 0% and 26%, respectively. For all but the last database, these figures are very close to the FAR, which implies that the algorithm is having difficulties distinguishing between matching and non-matching images, since it is accepting those that it shouldn't at approximately the same rate it is accepting those that it should. At the same time, it appears to be having more success, although still limited, on the fourth database, giving reason to believe that the method can achieve lower error rates if thoroughly tweaked.

Using the two-sample t-test, it is possible to determine whether the differences between the GAR and FAR are statistically significant, and thus determine the confidence with which the algorithm can be said to be working.

The null hypothesis, H_0, is taken to be that the FAR and GAR are equal:

$$H_0 : FAR = GAR. \tag{6.1}$$

The alternative hypothesis, H_a, is that the FAR is less than the GAR:

$$H_a : FAR < GAR. \tag{6.2}$$

In total, the entire testing procedure ran $n_1 = 10,913$ false reject tests and $n_2 = 19,701$ false accept tests, a little below the numbers estimated in last chapter due to some failed enrollments. The number of genuine accepts in the false reject tests were $X_1 = 989$ and the number of false accepts in the false accept tests were $X_2 = 1,317$. Considering that these are actually binomial distributions, we can estimate the mean and standard deviation of the sample proportions:

$$GAR = \mu_1 = \frac{X_1}{n_1} = \frac{989}{10913} = 0.0906258590672 \tag{6.3}$$

$$FAR = \mu_2 = \frac{X_2}{n_2} = \frac{1317}{19701} = 0.0668493985077 \tag{6.4}$$

$$\sigma_1 = \sqrt{\frac{\mu_1(1-\mu_1)}{n_1}} = 0.00278405417919 \tag{6.5}$$

$$\sigma_2 = \sqrt{\frac{\mu_2(1-\mu_2)}{n_2}} = 0.00177942827272 \tag{6.6}$$

Using this, it is now possible to compute the two-sample t statistic

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} = 814.223407604. \tag{6.7}$$

The degrees of freedom k is conservatively estimated as the smaller of the $n_1 - 1$ and $n_2 - 1$:

$$k = \min(n_1 - 1, n_2 - 1) = 10912. \tag{6.8}$$

Checking any t-table, it can be seen that $P[t(10912) > 814]$ is a really tiny number. Indeed, for all practical purposes, it is reasonable to say $P[t(10912) > 814] \approx 0$. The large sample size has ensured that it is possible to reject the null hypothesis, $FAR = GAR$, with well over 99.99% confidence.

Having rejected the null hypothesis with such confidence, it can be confidently asserted that the method presented here is successful at discriminating between matching and non-matching fingerprints, despite the large false accept rate and false reject rate reported from the testing procedure. Although the GAR is only slightly larger than the FAR, approximately 2.5%, the large sample size ensures the difference between the FAR and the GAR is statistically significant.

Comparison with a simplified method

When comparing the newly conceived method to the more traditional method on which it is based - one which uses the same image enhancement, alignment and matching algorithms with similar parameters, but avoids Reed-Solomon coding and Cartesian block transformation - the most notable differences are a great decrease in the False Reject Rate, averaging at 24% rather than 91%, and a great increase in the False Accept Rate, averaging at 68% rather than 6.5%.

The same of number of matching minutiae were required for the simplified algorithm to consider two fingerprints mates as was necessary to reproduce the entire minutiae set with the Reed-Solomon codes in the full algorithm. The high FAR and low FRR suggest that it is easier to match the minutiae points in the untransformed space than the transformed space. The Cartesian block transformation thus appears to be adversely affecting the recognition performance, and minutiae points that would have matched in the untransformed space do not necessarily match in the transformed space.

The genuine accept rates for the four databases are 59%, 81%, 96.7% and 69% respectively. As with the full algorithm, the GARs are in the vicinity of the FARs, also indicating that this simplified algorithm is doing only slightly better than blind guessing. However, the difference between the GARs and FARs do indeed show that this simplified algorithm is doing considerably better than the full algorithm in this respect.

Therefore, there is reason to believe that although a part of the error rate exhibited by the full algorithm can be blamed on the foundation of feature extraction and minutiae alignment algorithms on which it was built, the stages of Cartesian transformation and error-correction coding, and their inversions, do indeed adversely affect the recognition accuracy.

Since the Reed-Solomon error correction codes are only instrumental in recovering the full original minutiae set after matching, and are not actually involved in the minutiae matching process, the Cartesian transformation can likely be blamed for the majority of the error rate differences between the new method and the simplified method.

The Cartesian transformation shuffles minutiae points around based on which block they are in in a rectangular grid. One possible source of error is that the block a given minutiae point resides in varies across acquisitions, and therefore is transformed differently each time. If the grid is very fine, even small disturbances in the minutiae point may cause it to change the block it is in, and thus it is transformed to an entirely different place in two acquisitions. Two minutiae points that would have matched in the untransformed space can reside in different blocks, and their transformed locations therefore could be further apart, so that they will not match in the transformed space.

These small disturbances may be caused by a number of factors, from inaccuracies in the estimation of the core location and principal direction, to skin deformation and environmental effects in the acquisition process.

Making the blocks larger may counteract some of these effects, making the matching in the transformed space more resistant to small variations in the minutiae positions. However, this will affect the difficulty of inverting the transformation: fewer and larger blocks make it much easier to guess the original minutiae locations. Evidently, a balance must be chosen for how fine to make the grid of blocks for the Cartesian transformation, balancing the concerns of matching performance and noninvertibility.

In order to decrease the error-rate of the method presented here, it therefore appears to be necessary to enhance the feature extraction algorithm, as well as take a closer look at the Cartesian transformation that is performed on the aligned minutiae points.

Comparison with FVC median and winner entries

Comparing the accuracy of the recognition algorithm presented here with the median entry and winning entry of the FVC, the recognition accuracy of the algorithm presented here is clearly inferior to either by a large margin.

6.2 Irreversability of the Transformation

Since the fingerprint template in the end contains a reasonable amount of information, a question of great importance is whether it would be feasible for a potential attacker to use the information to reverse the transformation and recreate the original minutiae map. A simple model is considered here, which will illuminate certain aspects of the transformation, and a few numbers will be calculated to estimate the irreversibility of the transformation, both when using data from the template in the implementation presented in this thesis, and when considering requirements for other potential transformations.

Since the Reed-Solomon error-correction codes are in the template, the attacker needs only to reverse the transformation for a subset of the minutiae points. Imagine that each transformed minutiae point has on average n possible original positions - so given the transformed location of a minutiae point, it could possibly come from any of those n locations. Assume that the location of m minutiae points are needed for the Reed-Solomon decoding.

Considering an average transformed minutiae point, Λ, its original configuration may be any of the $\lambda_i, i = 1, ..., n$. So the probability of guessing the original location of the minutiae point correctly is therefore $\frac{1}{n}$. Assume, for the sake of the argument, that the locations of the minutiae points are independent, such that guessing the original location for each minutiae point is, similarly, $\frac{1}{n}$. An attacker must then make m correct such guesses simultaneously to be able to perform the Reed-Solomon decoding. The probability of independently guessing m minutiae locations correctly thus becomes $\left(\frac{1}{n}\right)^m = \frac{1}{n^m}$.

In the implementation tested in this thesis, a simple permutation, a one-to-one transformation, was used as the transformation for the sake of simplicity. Being only a permutation, the original location of each transformed minutiae may easily be found since the transformation parameters are given.

When not simplified to a permutation, the Cartesian transformation discussed here shows a bit more resistance, yet it is still fairly weak. For example, assuming that each transformed minutiae point has two possible origins on average, and that eight minutiae points are required to reverse the Reed-Solomon coding, it only requires $2^8 = 256$ guesses to deplete the entire search space and guarantee a reversal of the transformation.

However, an interesting question is what multiplicity the transformation must provide for a minutiae point in order to be considered secure. Given a transformed minutiae point, how many options for the original minutiae point is needed in order to create a secure solution, and how many such reversed minutiae points should be required to reverse the Reed-Solomon coding? Answering these questions should shed more light on the required qualities of the transformation.

If the requirement is, for instance, that it should take at least one year of CPU time to reverse the transformation, it is possible to estimate the parameters n and m required in order to accomplish this. One year consists of approximately 3.2×10^7 seconds, and a generous assumption is that a CPU can make 10^7 guesses each second. The CPU thus makes 3.2×10^{14} guesses in a year. On average, an attacker could be expected to succeed after guessing through half the search space, so the entire search space should be twice as large: 6.4×10^{15}. So one must therefore choose n and m such that $n^m > 6.4 \times 10^{15}$.

Fortunately, n^m grows very fast, both in n and m. By setting $n = 10$ and $m = 16$ the expected CPU time to invert the transformation would exceed one year. This would mean that each transformed minutiae point would have ten possible origins, and that sixteen minutiae points were required for the Reed-Solomon decoding. Although the current transformation does not provide the required accuracy to match sixteen minutiae points in the transformed space nor the required number of possible original locations for a transformed minutiae point, there is a great possibility that such a transformation may be designed.

6.3 Notes on Feature Extraction

Recognition accuracy relies very heavily on the performance of the feature extraction phase. Although feature extraction constitutes an essential part of this algorithm, the method used herein is by no means unique, and can easily be replaced with other image enhancement, minutiae detection and core detection algorithms without changing the essence of the method presented herein - the combination of Cartesian block transformation and Reed-Solomon coding.

Since the performance of the matching process is so sensitive to the performance of the feature extraction algorithms, a critical look at the feature extraction algorithm used in the implementation can be useful.

Several strange effects appear during the feature extraction process that should be amended. The local ridge frequency estimates, for instance, appear correct at some places yet unreliable at others. The local ridge frequency estimate in figure 3.9 can be seen to contain low frequencies in the upper right part of the image, although it is clear from the image that there is indeed no ridge-valley structure present. This also taints the calculation of the mask of unrecoverable regions, as in figure 3.10.

During minutiae detection on the fingerprint skeleton, many minutiae points are detected at the edge of the fingerprint. These are false minutiae points, as they are not actually ridge endings - they only represent the border of the fingerprint, and should be excluded from the minutiae set. However, the algorithm used here detects an overwhelming amount of minutiae points in these unreliable areas, which could explain some of the poor performance of this algorithm. This effect is clearly present in figure 3.16, where an overwhelming majority of the minutiae points detected in the image are along the edge of the fingerprint, and hence do not represent real minutiae points.

The Poincare index method that is used to locate the core of the fingerprint image is unable to locate stable reference points in arch-type fingerprints, which excludes a whole class of fingerprints from the current procedure. This could also help explain the high error rate, as this is a very essential part of the algorithm. There are other algorithms that are able to detect stable reference points even in arch-type fingerprints, which perhaps may be used in this algorithm with more success.

Considering the number and severity of adverse feature extraction effects in the algorithms used here, there is reason to believe that a significant portion of the high error rate can be explained by the faults of the feature extraction algorithms used, and can similarly be eliminated by choosing a more optimal set of algorithms.

Chapter 7

Conclusion

The method for generating cancellable templates presented in this thesis is showing a certain degree of success, although it is severely hampered by the limited recognition accuracy shown in the test scenarios. The accuracy of the method is clearly not competitive with the state-of-the-art methods tested in the Fingerprint Verification Competition 2000.

However, results from testing a comparable method with the Reed-Solomon coding and Cartesian transformation left out suggests that quite a large portion of the error rates are caused by deficiencies in the underlying feature extraction and matching algorithms, and further investigation has supported this. Furthermore, the Cartesian transformation is also suspected to deteriorate the matching accuracy considerably.

Although the accuracy of this particular implementation is much too low for any practical applications, the results do suggest that the general concept actually does work, as the difference between the Genuine Accept Rate and the False Accept Rate is statistically significant, with the Genuine Accept Rate being consistently higher than the False Accept Rate.

The general framework of performing the matching in a transformed feature space in order to reverse an otherwise irreversible transformation, and subsequently applying error correction codes to fully reverse the transformation, can conceivably be used with other feature extraction algorithms and with other transformations than those presented here.

Further work should therefore concentrate on improving the feature extraction algorithm in order to improve recognition accuracy, as this appears to be a large source of error. Specific changes that should be made to the feature extraction algorithms are changing the algorithm used for estimating the local ridge frequency, removing the false minutiae points that appear at the edge of the fingerprint and replace the core detection algorithm with one that can detect a stable reference point and principal direction in all fingerprint types.

Theoretical estimates suggest that the irreversibility of the Cartesian block transformation chosen here is weak, and should be exchanged for a much more appropriate transformation for improved security and privacy. In addition, the difference between the results when the transformation was and was not included suggests that the chosen transformation currently interferes with the matching process to a great extent.

Overall, the proof-of-concept implementation tested here shows that the proposed method is promising and represents a fingerprint matching methodology that impedes fake fingerprint generation, provides revocability, and prohibits database cross-referencing, although the high error-rates are prohibitive of practical use, as is. A lot of work remains to be done with regards to choosing optimal feature extraction algorithms and more appropriate transformations before more definite conclusions about the success of the proposed method can be drawn.

Bibliography

[1] Bioenable: Identify, automate, track, http://www.bioenabletech.com/, accessed: 23rd of october, 2007.

[2] Fingerprint verification system: http://fvs.sourceforge.net/, accessed: 23rd of october, 2007.

[3] Forensic evidence.com: Identification evidence/court excludes fingerprint critic's testimony as junk science, http://forensic-evidence.com/site/id/cole_junksci.html, accessed: 23rd of october, 2007.

[4] Reed-solomon python extension module: http://hathawaymix.org/software/reedsolomon, accessed: 23rd of october, 2007.

[5] Sfinge: Synthetic fingerprint generator. biometric system laboratory, university of bologna, http://biolab.csr.unibo.it/, accessed: 23rd of october, 2007.

[6] Pc world - laptop to offer fingerprint id, http://www.pcworld.com/article/id,35415-page,1/article.html, accessed: 23rd of october, 2007, 2000.

[7] Eu passport will contain two biometrics. In *Biometric Technology Today*, volume 13, page 12. Elsevier B.V., January 2005.

[8] Mythbusters, season 4 - episode 16: Crimes and myth-demeanors 2, 2006.

[9] R. Achs, R.G. Harper, and N.J. Harrick. Unusual dermatoglyphics associated with major congenital malformations. *New England Journal of Medicine*, pages 275:1273–1278, 1966.

[10] Russell Ang, Rei Safavi-Naini, and Luke McAven. Cancelable key-based fingerprint templates. In *Information Security and Privacy*, volume 3574, pages 242–252. 2005.

[11] R. D. Bahuguna and T. Corboline. Prism fingerprint sensor that uses a holographic optical element. *Applied Optics*, 35:5242–5245, 1996.

[12] A. M. Bazen, G. T. B. Verwaaijen, S. H. Gerez, L. P. J. Veelenturf, and B. J. van der Zwaag. A correlation-based fingerprint verification system. In *Proceedings of the ProRISC Workshop on Circuits, Systems and Signal Processing, Veldhoven, the Netherlands*, pages 205–213, Netherlands, November 2000. STW Technology Foundation.

[13] Asker M. Bazen and Sabih H. Gerez. Elastic minutiae matching by means of thin-plate spline models. In *International Conference on Pattern Recognition*, volume 2, pages 985–988. University of Twente, Dept. of Electrical Engineering, 2002.

[14] Ruud Bolle, J. H. Connell, and N. K. Ratha. Biometric perils and patches. *Pattern Recognition*, 35(12):2727–2738, 2002.

[15] Ruud Bolle, Jonathan Connell, Sharath Pankanti, Nalini Ratha, and Andrew Senior. *Guide to Biometrics.* Springer-Verlag New York, Inc., 2004.

[16] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni. Fingerprint classification by directional image partitioning. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 21, pages 401–421, 1999.

[17] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni. Can fingerprint be reconstructed from iso templates? In *International Conference on Control, Automation, Robotics and Vision*, pages 1–6, December 2006.

[18] K.C. Chan, Y.S. Moon, and P.S. Cheng. Fast fingerprint verification using subregions of fingerprint images. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 14, pages 95–101, January 2004.

[19] Byoung-Ho Cho, Jeung-Seop Kim, Jae-Hyung Bae, In-Gu Bae, and Kee-Young Yoo. Core-based fingerprint image classification. In *Internation Conference on Pattern Recognition*, volume 2, pages 859–862. Dept. of Computer Engineering, Kyoungpook National University, 2000.

[20] Sarat C. Dass, Yongfang Zho, and Anil K. Jain. Validating a biometric authentication system: Sample size requirements. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, pages 1902–1913, December 2006.

[21] G.I. Davida, Y. Frankel, and B.J. Matt. On enabling secure applications through off-line biometricidentification. In *Proceedings to the 1998 IEEE Symposium on Security and Privacy*, pages 148–157, 1998.

[22] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027, pages 523–540. Springer Berlin / Heidelberg, 2004.

[23] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

[24] B. Wicz et. al. Fingerprint structure imaging based on an ultrasound camera. *Instrumentation science and technology*, 27(4):295–303, 1999.

[25] Jennie Lugassy et. al. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *The American Journal of Human Genetics*, 79(4):724–730, October 2006.

[26] J. Galbally-Herrero, J. Fierrez-Aguilar, J.D. Rodriguez-Gonzalez, F. Alonso-Fernandex, Javier Ortega-Garcia, and M. Tapiador. On the vulnerability of fingerprint verification systems to fake fingerprints attacks. In *Proceedings 2006 40th Annual IEEE International Carnahan Conferences Security Technology*, pages 130–136, October 2006.

[27] Mary Hanson. Fingerprint-based forensics identify argentina's desaparecidos. In *IEEE Computer Graphics and Applications*, volume 20, pages 7–10, 2000.

[28] Ying Hao, Tienio Tan, and Yunhong Wang. An effective algorithm for fingerprint matching. In *TENCON 02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, volume 1, pages 519–522. National Lab of Pattern Recognition, CAS, Institute of Automation, Beijing, October 2002.

[29] Lin Hong, Anil K. Jain, Sharath Pankanti, and Ruud Bolle. Fingerprint enhancement. pages 202–207. Department of Computer Science, Michigan State University, Exploratory Computer Vision Group, IBM T.J. Watson Research Center, 1996.

[30] Lin Hong, Yifei Wan, and Anil Jain. Fingerprint image enhancement: Algorithm and performance evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):777–789, 1998.

[31] Anil Jain, Yi Chen, and Meltem Demirkus. Pores and ridges: Fingerprint matching using level 3 features. *icpr*, 4:477–480, 2006.

[32] Anil K. Jain. Biometric recognition: How do i know who you are? 3540/2005:1–5, June 2005.

[33] Anil K. Jain, Lin Hong, and Ruud Bolle. On-line fingerprint verification. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19, pages 302–314, April 1997.

[34] Anil K. Jain, Salil Prabhakar, Lin Hong, and Sharath Pankanti. Fingercode: A filterbank for fingerprint representation and matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 187–193, 1999.

[35] Ari Juels and Madhu Sudan. A fuzzy vault scheme. *Des. Codes Cryptography*, 38(2):237–257, 2006.

[36] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*, pages 28–36, New York, NY, USA, 1999. ACM Press.

[37] Phil Karn. Dsp and fec library: http://www.ka9q.net/code/fec/, accessed: 23rd of october, 2007.

[38] Masahiro Kawagoe and Akio Tojo. Fingerprint pattern classification. *Pattern Recogn.*, 17(3):295–303, 1984.

[39] J. Klett. Thermal imaging fingerprint technology. In *Proceedings of Biometric Consortium 9th Meeting*, April 1997.

[40] C. Klimanee and D.T. Nguyen. Classification of fingerprints using singular points and their principal axes. In *International Conference on Image Processing*, volume 2, pages 849–852. School of Engineering, University of Tasmania, October 2004.

[41] J.K. Lee, S.R. Ryu, and K.Y. Yoo. Fingerprint-based remote user authentication scheme using smartcards. In *Electronics Letters*, volume 38, pages 554–555, June 2002.

[42] Jean-Paul Linnartz and Pim Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *Audio- and Video-Based Biometric Person Authentication*, volume 2688, page 1059. Springer Berlin / Heidelberg, 2003.

[43] Xiping Luo, Jie Tian, and Yan Wu. A minutia matching algorithm in fingerprint verification. In *International Conference on Pattern*

Recognition, volume 4, pages 833–836. AILAB, Institute of Automation, The Chinese Academy of Sciences, 2000.

[44] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, and A.K. Jain. Fvc2002: Second fingerprint verification competition.

[45] Dario Maio and Davide Maltoni. Direct gray-scale minutiae detection in fingerprints. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(1):27–40, 1997.

[46] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. *Handbook of Fingerprint Recognition.* Springer Science+Business Media, Inc., 2003.

[47] Aparecido Nilceu Marana and Anil K. Jain. Ridge-based fingerprint matching using hough transform. In *IEEE Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing*, pages 112–119, 2005.

[48] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino. Impact of artificial gummy fingers on fingerprint systems. In *Proc. of SPIE*, volume 4677, pages 275–289, 2002.

[49] B. Miller. Vital signs of identity. In *IEEE Spectrum*, volume 31, pages 22–30, 1994.

[50] M. Mimura, S. Ishida, and Y. Seto. Fingerprint verification system on smart card. In *International Conference on Consumer Electronics, 2002. ICCE. 2002 Digest of Technical Papers.*, volume 1 of *All IEEE Conferences*, pages 182–183. Syst. Dev. Lab., Hitachi Ltd., Yokohama, 2002.

[51] Shimon K. Modi, Stephen J. Elliott, Jeff Whetsone, and Hakil Kim. In *IEEE Workshop on Automatic Identification Advanced Technologies*, pages 19–23. Industrial Technology, College of Technology Purdue University, INHA University, June 2007.

[52] Fabian Monrose, Michael K. Reiter, Qi Li, and Susanne Wetzel. Cryptographic key generation from voice. *sp*, 00:0202, 2001.

[53] Fabian Monrose, Michael K. Reiter, and Susanne Wetzel. Password hardening based on keystroke dynamics. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*, pages 73–82, New York, NY, USA, 1999. ACM Press.

[54] Y. S. Moon, K. L. Ng, S. F. Wan, and S. T. Wong. Collaborative fingerprint authentication by smart card and atrusted host. In *2000 Canadian Conference on Electrical and Computer Engineering*, volume 1

of *All IEEE Conferences*, pages 108–112. Dept. of Comput. Sci. and Eng., Chinese Univ. of Hong Kong, Shatin, 2000.

[55] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[56] International Organization of Standardization. *ISO/IEC 19794-2: Information technology – Biometric data interchange formats – Part 2: Finger minutiae data*.

[57] Sharath Pankanti, Salil Prabhakar, and Anil K. Jain. On the individuality of fingerprints. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 24, pages 1010–1025. IBM T.J. Watson Research Center, DigitalPersona Inc., Dept. of Comp. Sci. and Eng. Michigan State University, August 2002.

[58] Nalini Ratha, Sharat Chikkerur, Jonathan Connell, and Ruud M. Bolle. Generating cancelable fingerprint templates. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 29, pages 561–572, April 2007.

[59] Nalini Ratha, Jonathan Connell, Ruud M. Bolle, and Sharat Chikkerur. Cancelable biometrics: A case study in fingerprints. In *The 18th International Conference on Pattern Recognition*, pages 370–373, 2006.

[60] Nalini Ratha and Ruud Bolle et. al. *Automatic Fingerprint Recognition Systems*. Springer-Verlag New York, Inc., 2004.

[61] N.K. Ratha, J.H. Connell, and R.M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634, 2001.

[62] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, June 1960.

[63] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), April 1992.

[64] Danny Roberge, Colin Soutar, and Bhagavatula Vijaya Kumar. Optimal correlation filter for fingerprint verification. volume 3386, pages 123–133. SPIE, 1998.

[65] B.G. Sherlock, D.M. Monro, and K. Millard. Fingerprint enhancement by directional fourier filtering. In *IEEE Proceedings - Vision, Image and Signal Processing*, volume 141, pages 87–94, 1994.

[66] Sargur N. Srihari, Harish Srinivasan, Gang Fang, and Arvindakshan Ravichandran. Discriminability of fingerprints of twins.

[67] Yagiz Sutcu, Husrev Taha Sencar, and Nasir Memon. A secure biometric authentication scheme based on robust hashing. In *MM&Sec '05: Proceedings of the 7th workshop on Multimedia and security*, pages 111–116, New York, NY, USA, 2005. ACM Press.

[68] Kiyoaki Takiguchi. Uspto app. no 20070116330: Living-tissue pattern detecting method, living-tissue pattern detecting device, biometric authentication method, and biometric authentication device, 2005.

[69] M. Tartagni and R. Guerrieri. A fingerprint sensor based on the feedback capacitive sensingscheme. In *IEEE Journal of Solid State Circuits*, volume 33, pages 133–142, January 1998.

[70] Andrew BJ Teoh and David CL Ngo. Biophasor: Token supplmented cancellable biometrics. In *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*, All IEEE Conferences, pages 1–5. Faculty of Information Science and Technology, Multimedia University, December 2006.

[71] Andrew BJ Teoh, David CL Ngo, and Alwyn Goh. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern Recognition*, 37(11):2245–2255, November 2004.

[72] Yu Tong, Hui Wang, Daoying Pi, and Qili Zhang. Fast algorithm of hough transform-based approaches for fingerprint matching. In *IEEE Proceedings of the 6th World Congress on Intelligent Control and Automation*, volume 2, pages 10425–10429, June 2006.

[73] Pim Tuyls, Anton H.M. Akkermans, Tom A.M. Kevenaar, Geert-Jan Schrijen, Asker M. Bazen, and Raimond N.J. Veldhuis. Practical biometric authentication with template protection. In *Audio- and Video-Based Biometric Person Authentication*, volume 3546, pages 436–446. Springer Berlin / Heidelberg, 2005.

[74] Umut Uludag, Sharath Pankanti, and Anil K. Jain. Fuzzy vault for fingerprints. *Lecture notes in computer science*, 3546:310–319, 2005.

[75] C. Vielhauer, R. Steinmetz, and A. Mayerhofer. Biometric hash based on statistical features of online signatures. In *Proceedings. 16th International Conference on Pattern Recognition*, volume 1, pages 123–126, 2002.

[76] Chengfeng Wang, Marina Gavrilova, Yuan Luo, and Jon Rokne. An efficient algorithm for fingerprint matching. In *IEEE Proceedings of the 18th International Conference on Pattern Recognition*, pages 1034–1037. University of Calgary, Canada, 2006.

[77] Shenglin Wang and Ingrid M. Verbauwhede. A secure fingerprint matching technique. In *Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications*, volume 1 of *International Multimedia Conference*, pages 108–112. SIGMULTIMEDIA: ACM Special Interest Group on Multimedia, 2003.