

**A comparative study on Match-on-Card
fingerprint matching algorithms using the
ISO/IEC 19794-2:2011 format**

MS Ratsoma



orcid.org 0000-0002-8834-7513

Dissertation submitted in fulfilment of the requirements for the
degree *Master of Science in Computer and Electronic
Engineering* at the North-West University

Supervisor: Prof ASJ Helberg

Co-supervisor: Dr KK Nair

Graduation May 2018

Student number: 26975203

DECLARATION

I, Mahlatse Sophia Ratsoma, do hereby declare that this dissertation, submitted in partial fulfilment of the requirements for the degree Master of Science in Computer and Electronic Engineering at the Potchefstroom Campus of the North-West University, is my own work, except where duly acknowledged, and has not previously been submitted by me for a degree at another university.

Mahlatse Sophia Ratsoma

A handwritten signature in dark ink, appearing to read 'S. Ratsoma', is written over a horizontal line.

September 2017

EDITOR'S CERTIFICATE

Pretoria, 23 August 2017

To whom it may concern

This is to certify that I, Klaus-Peter Klein, Translator and Language Practitioner by profession, have edited a thesis for Mahlatse Ratsoma.

The title of the thesis, containing approximately 25 400 words, was

A comparative study on Match-on-Card fingerprint matching algorithms using the ISO/IEC 19794-2:2011 format.

Klaus P. Klein

805, 26th Ave, Rietfontein

Pretoria, 0084

Tel/Fax +27-72-140-5300

e-mail: klausklein@taalring.com

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisors Dr Kishor Krishnan Nair and Prof Albert Helberg. They have provided me with heartfelt support and guidance at all times, with thought-provoking questions, inspiration and suggestions in my quest for knowledge.

To my mentor Meshack Shabalala who, after changing workplaces, continued to give me invaluable support towards my studies. To Johan Van der Merwe for the challenging brainstorming sessions we had.

To my friends and colleagues at the CSIR who at any time stopped their work just to listen to me and assist with my work, I am grateful.

To my parents Mosima and Chichilala Ratsoma, my siblings Pebetsi Ratsoma and Raesetša Mogale and to my boyfriend Takalani Mbooi; I am grateful for your unconditional love, support, understanding and prayers throughout my studies. I am grateful for motivation and for pushing me daily to do better at my work, for listening to me even though some things you don't understand, and for reassuring me when I was discouraged. To my family and wonderful friends who always made an effort to keep me focused and get back up when the pressure seemed too much, I am forever grateful.

To the CSIR, for granting me the Masters Studentship, I am very appreciative for the opportunity. The CSIR did not only provide financial support but also facilities and human capital that I have found to be really helpful.

I am grateful to everyone not mentioned by name for direct or indirect assistance in this project. This journey would have not been easier without you all.

Last but not least, I thank God for the strength and ability to undertake this research study and to persevere until the end.

ABSTRACT

South Africa introduced smart ID cards in 2013 for identification and authentication of South African citizens. This came as an improvement on the old green bar-coded ID book in which authentication was not secure enough. However, the use of the smart-card for authentication comes with new problems. The minutia points that are stored in the smart-card, due to standardisation, appear to cause accuracy problems when used with on-card minutiae comparison algorithms.

This study set out to investigate how the compact-card and record formats of minutia points affect the comparison accuracy and to also find out an algorithm that performs better with the minutiae format developed for use in smart-cards.

This was followed by identification of a problem which had not been studied in depth before, pertaining to the formats of minutia points used for on-card comparison. The methodology of the study is presented and algorithm designs are presented. An analysis of variance is also shown for the analysis of minutia points, and in doing so it was revealed that the minutia points are not significantly different from one another; although differences exist. This led to a detailed study of the MOC algorithms, where two algorithms were selected due to their basic structure that separated the algorithms.

The two categories of matching algorithms used are the local-features and the global-features based algorithms. The nearest-neighbourhood algorithm and the alignment algorithm fall into these categories respectively. These matching algorithms were implemented and also verified to ensure that they are within their specifications. In addition to implementing matching algorithms, the minutia points were also encoded in the 3-byte and 5-byte formats.

The results showed that the template does affect the accuracy and the effect is not the same on each of the algorithms. However, through manipulation of the resolutions that cause some loss of accuracy, the algorithms' performance with 3-byte can be significantly improved.

Keywords: Match-on-Card, alignment algorithm, nearest-neighbourhood algorithm, accuracy, resolution, ISO/IEC 19794-2:2011, compact card format.

CONTENTS

Chapter 1: Introduction	1
1.1 <i>Introduction.....</i>	1
1.2 <i>Problem statement</i>	2
1.3 <i>Research question.....</i>	3
1.4 <i>Research goal</i>	3
1.5 <i>Research objectives</i>	3
1.6 <i>Delimitations and limitations</i>	4
1.7 <i>Significance and rationale of the study</i>	5
1.8 <i>Assumptions</i>	6
1.9 <i>Dissertation layout</i>	6
1.10 <i>Summary</i>	7
 Chapter 2: Background	 8
2.1 <i>Forms of authentication</i>	8
2.1.1 <i>Authentication based on what you know.....</i>	8
2.1.2 <i>Authentication based on what you have</i>	8
2.1.3 <i>Authentication based on what you are</i>	9
2.2 <i>Biometrics and smart-cards</i>	11
2.3 <i>Fingerprint recognition</i>	12
2.3.1 <i>Minutia points.....</i>	13
2.3.2 <i>Basic enrolment and query process.....</i>	14
2.4 <i>Conformance to standards</i>	15
2.5 <i>ISO/IEC template use and MOC</i>	16
2.6 <i>Different minutiae-based fingerprint matching algorithms.....</i>	18
2.6.1 <i>Global features-based algorithms</i>	19
2.6.2 <i>Local features-based algorithms.....</i>	25

2.7	Summary	29
Chapter 3: Research Methodology		30
3.1	Collecting fingerprint images	30
3.2	Extraction of fingerprint minutiae	31
3.3	Template encoding	33
3.4	Minutiae matching	35
3.4.1	Evaluation of matchers	36
3.4.2	Analysis of match scores	36
3.5	Analysis of minutia points through ANOVA.....	37
3.5.1	ANOVA use cases	38
3.5.2	Assumptions for using ANOVA	39
3.5.3	POWER (errors introduced).....	39
3.5.4	How to use ANOVA	40
3.6	Bill of materials	42
3.6.1	Fingerprint images	42
3.6.2	Minutia point data	42
3.6.3	On-card fingerprint-matching algorithms.....	43
3.7	Summary	43
Chapter 4: Design		44
4.1	Nearest-neighbourhood algorithm.....	44
4.1.1	Complexity analysis of the neighbourhood algorithm:.....	49
4.2	Alignment algorithm.....	49
4.2.1	Complexity analysis of the alignment algorithm	53
4.3	ANOVA.....	54
4.4	Summary	55

Chapter 5: Implementation.....	57
5.1 <i>Fingerprint collection</i>	57
5.2 <i>Minutiae extraction</i>	58
5.3 <i>Format conversion</i>	61
5.4 <i>Matching algorithms with no smart-card limitations.....</i>	64
5.4.1 <i>Nearest-neighbourhood algorithm</i>	64
5.4.2 <i>Alignment algorithm</i>	64
5.5 <i>Matching algorithms with smart-card constraints.....</i>	64
5.5.1 <i>Java Card</i>	65
5.5.2 <i>Python</i>	66
5.5.3 <i>Mapping Python with Java Card</i>	67
5.6 <i>ANOVA.....</i>	70
5.6.1 <i>Y-coordinate</i>	70
5.6.2 <i>X-coordinate</i>	71
5.6.3 <i>Orientation</i>	72
5.7 <i>Validation</i>	74
5.7.1 <i>Format converters.....</i>	74
5.7.2 <i>Matching algorithms.....</i>	77
5.8 <i>Summary</i>	81
 Chapter 6: Results, analysis, and validation	 82
6.1 <i>Algorithms.....</i>	82
6.1.1 <i>Nearest-neighbourhood algorithm</i>	82
6.1.2 <i>Alignment algorithm</i>	84
6.2 <i>Improvement of algorithms.....</i>	87
6.2.1 <i>Nearest-neighbourhood algorithm</i>	87
6.2.2 <i>Alignment algorithm</i>	88
6.3 <i>ANOVA.....</i>	89

6.3.1	<i>Y-coordinate</i>	89
6.3.2	<i>X-coordinate</i>	89
6.3.3	<i>Orientation</i>	89
6.3.4	<i>Summary of ANOVA analysis</i>	90
6.4	<i>Validation of results</i>	90
6.5	<i>Summary</i>	92
Chapter 7: Conclusion		93
7.1	<i>Research synopsis</i>	93
7.2	<i>Significant contributions of the study</i>	94
7.2.1	<i>Analysis of different MOC algorithms</i>	94
7.2.2	<i>Investigation of the accuracy change</i>	95
7.2.3	<i>Recommendation for a suitable MOC algorithm</i>	96
7.3	<i>Future studies</i>	97
REFERENCES		98
APPENDICES		102

LIST OF FIGURES

Figure 2-1: Extracted fingerprint features (minutiae).....	14
Figure 2-2: Basic enrolment and query process	14
Figure 2-3: Flow diagram of an ultra-low memory fingerprint matching algorithm and its implementation on a 32-bit smart-card.	20
Figure 2-4: ISO/IEC standards for on-card biometric comparisons.....	23
Figure 2-5: Flow diagram of the nearest-neighbourhood matching algorithm for a Java Card	26
Figure 2-6: Finger-code matching algorithm flow diagram.	28
Figure 3-1: Research methodology.....	30
Figure 3-2: Functional steps of MINDTCT minutiae extraction algorithm adapted from [46]	32
Figure 4-1: Minutia-neighbour relationship.....	46
Figure 4-2: Neighbourhood features	47
Figure 4-3: Bounding boxes for aligned minutia points	52
Figure 4-4: Flow chart of the alignment algorithm	53
Figure 4-5: Minutia points in different formats.....	54
Figure 4-6: Samples of the dependent variable Minutiae (x, y, and orientation)	55
Figure 5-1: Converting from tiff to jpg using Imagemagick.....	59
Figure 5-2: Extracted minutiae	60
Figure 5-3: Snapshot of minutia neighbourhood features	61
Figure 5-4: Raw minutiae to 3-byte data format conversion.....	62

Figure 5-5: Raw minutiae-to-5-byte data format conversion	63
Figure 5-6: Compact card converter unit tests	75
Figure 5-7: Record format converter unit tests.....	76
Figure 5-8: Execution of compact-card converter unit tests	77
Figure 5-9: Execution of record converter unit tests	77
Figure 5-10: Test cases for alignment algorithm.....	78
Figure 5-11: Test cases for the nearest-neighbourhood algorithm	79
Figure 5-12: Execution of the alignment algorithm test.....	80
Figure 5-13: Execution of the nearest-neighbourhood algorithm test	80
Figure 6-1: PDF for nearest neighbourhood algorithm with 5-byte data	83
Figure 6-2: PDF for nearest neighbourhood algorithm with 3-byte data	84
Figure 6-3: PDF for alignment algorithm with 5-byte data.....	85
Figure 6-4: PDF for alignment algorithm with 3-byte data.....	86
Figure 6-5: Nearest-neighbourhood algorithm adapted for 3-byte data	87
Figure 6-6: Alignment algorithm adapted for 3-byte data.....	88

LIST OF TABLES

Table 1-1: 3-byte minutia format	2
Table 1-2: 6-byte minutia format	2
Table 2-1: Definition of coordinates of minutia location (adapted [9])	16
Table 3-1: Minutiae field sizes	34
Table 3-2: ANOVA errors	40
Table 3-3: ANOVA table	41
Table 3-4: Inference about the null hypothesis	42
Table 4-1: Nearest-neighbourhood matching algorithm – Source [38].....	44
Table 4-2: Steps taken to calculate neighbourhood dissimilarity value	48
Table 4-3: Alignment algorithm	50
Table 5-1: FVC2000 Scanners	57
Table 5-2: FVC2002 Scanners	58
Table 5-3: Format conversion pseudocode.....	61
Table 5-4: Java Card resources.....	66
Table 5-5: Java Card built-in data types	66
Table 5-6: Python data types	67
Table 5-7 : Single-factor ANOVA for Y-coordinates in FVC2000	70
Table 5-8: Single- factor ANOVA for Y-coordinates in FVC2002	71
Table 5-9: Single-factor ANOVA for X-coordinates in FVC2000	71
Table 5-10: Single factor ANOVA for X-coordinates in FVC2002	72
Table 5-11: Single factor ANOVA for orientation on the FVC2000 database.....	72

Table 5-12: Single factor ANOVA for orientation on the FVC2002 database..... 73

Table 6-1: Match scores in original implementation..... 91

Table 6-2: Match scores on the improved algorithms 91

LIST OF ABBREVIATIONS

Abbreviation	Definition
ANOVA	Analysis of Variance
ANSI/NIST	American National Standard for Information/ National Institute of Standards and Technology
CBEFF	Common Biometric Exchange File Format
CPU	Central Processing Unit
EER	Equal Error Rate
FBI	Federal Bureau of Investigation
FMR	False Match Rate
FNMR	False Non-Match Rate
FVC	Fingerprint Verification Competition
INCITS	International Committee for Information Technology Standards
ISO/IEC	International Organization for Standardization/ International Electrotechnical Commission
ISO-CC	International Organization for Standardization – Compact Card
JPEG	Joint Photographic Experts Group
MINDTCT	Minutiae Detection
MINEX	Minutiae Interoperability Exchange
MOC	Match on Card
NBIS	NIST Biometric Image Software
NIST	National Institute of Standards and Technology
OS	Operating System
PDF	Probability Density Function

PIN	Personal Identification Number
POPI	Protection of Personal Information
PVM	Python Virtual Machine
SOC	System on Card
TIFF	Tagged Image File Format
TOC	Template on Card
WSQ	Wavelet Scalar Quantization
ULW	Universal Latent Workstation

Chapter 1: Introduction

1.1 Introduction

The scientific establishment of fingerprint recognition in the 19th century by Galton Francis and Henry Faulds among others led to a breakthrough for law enforcement agencies worldwide. Fingerprint recognition was used to establish the identity of a victim or suspect and for identifying repeat offenders. It was then later extended into other applications outside of law enforcement, such as identity management, access control (physical and logical), and Automated Teller Machine (ATM) transactions [1]. In general, fingerprint recognition is used for identification and verification of a person's identity [2].

Smart-cards are credit card-sized memory and processor-enabled cards [3]. They come with an embedded processor chip of 8- to 32-bit central processing units (CPU) and with a memory capacity of 140 bytes to 8 kilobytes (kB). Smart-cards have allowed for a diverse number of applications, such as fingerprint recognition for access control, identity management, information storage, and processing of money (bank cards). In order to increase the level of security and ensure that smart-card users are indeed who they are, biometric recognition can be used together with smart-cards. Fingerprints are commonly used as a biometric characteristic for securing smart-cards.

Match-on-card (MOC), which will be explained in detail in the next chapter, is one popular way of merging fingerprint recognition with smart-cards. In MOC, the matching process of fingerprint verification is performed inside the card. Since smart-cards have less memory capacity than computers, it is better to use optimised fingerprint minutiae templates than to use the actual fingerprint image. Fingerprint minutiae templates offer a compact representation, cheap computation, and better privacy since they are not easily convertible to full fingerprints [4].

The International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) have developed the ISO/IEC 19794-2:2011 standard for fingerprint minutiae format. The standard allows for a maximum of 60 minutiae, each minutia representation being in 3-bytes. This implies that the maximum minutiae template size accommodates 180 bytes. The 3-byte representation is illustrated as in Table 1-1. In a

normal computer environment, there can be up to 360 bytes, with a single minutia point featuring up to 6 bytes of data, which is shown in Table 1-2. The last byte of the quality field is optional, so some minutia points may only have 5-bytes. In compressing the data to fit into a smart-card, a number of round-offs and/or truncations have to occur on the angle field, which implies a loss of resolution that could potentially lead to loss of accuracy in the matching process. This problem will be explained further in the problem statement.

Table 1-1: 3-byte minutia format

X-coordinate	Y-coordinate	Type	Angle
1 byte	1 byte	2 bits	6 bits

Table 1-2: 6-byte minutia format

Type	X-coordinate	Reserved	Y-coordinate	Angle	Quality
2 bits	14 bits	2 bits	14 bits	1 byte	1 byte

1.2 Problem statement

Before mentioning the problem statement, it is essential that the reader understands alternative terms that are used interchangeably in this thesis. The 3-byte minutia format is also referred to as compact card (CC), ISO-CC format, or the on-card comparison format. The 5-byte format can also be referred to as the record format

In a study conducted by Yoshida and Hangai the compact card format of ISO/IEC 19794-2:2005 was compared with other formats [5]. The compact-card format yielded results with a higher Equal Error Rate (EER) than the other formats. EER is the rate at which errors of falsely matching and falsely not-matching fingerprints are equal, and a higher EER indicates a lower accuracy [4].

Another study was conducted with the aim of assessing the capabilities of the ISO/IEC 7816 card-based on-card matching [6]. This study aimed, among other things, to measure the accuracy of MOC implementations using ISO/IEC 19794-2:2005 compact-card minutiae templates. It revealed that the ISO-CC representation offered performance approaching that of the template defined by American National Standards Institute (ANSI) International Committee for Information Technology Standards (INCITS) 378 for finger

minutiae. The study stated that performance of algorithms can be improved when algorithm providers know that they are specifically targeting the ISO-CC template.

Although it has already been established that the 3-byte representation has an effect on MOC fingerprint-matching accuracy, it is not clear whether different types of algorithms are affected similarly. When fingerprint minutia points are encoded into the ISO-CC template, there is a change in resolution in the x- and y-coordinates, and the angle values. This loss of resolution may have an effect on the matching accuracy of the fingerprint MOC algorithms. Hence, the research question that will be addressed in this study is as follows:

1.3 Research question

When using the ISO/IEC 19794-2:2011 on-card-comparison minutiae format, do different types of MOC algorithms have their accuracy affected with the same amount of Equal Error Rate?

1.4 Research goal

To investigate how the ISO/IEC 19794-2:2011 template for on-card matching affects the accuracy of different MOC algorithm approaches.

1.5 Research objectives

- i. To examine different MOC algorithms and their accuracy when used with the ISO/IEC 19794-2:2011 defined template.
- ii. To investigate the accuracy change when using the ISO/IEC 19794-2:2011 on-card comparison format and the ISO/IEC 19794-2:2005 record format for on-card comparison, in order to determine whether the difference is caused by the inefficiency of the algorithm or exacerbated by the template format.
- iii. To be able to recommend a type of MOC algorithm that can best be used with the ISO template format and which will yield good results with regard to accuracy.

1.6 Delimitations and limitations

Extraction algorithms when used by different vendors generally result in different minutia points, as reported by Tabassi et al [4]. Different extraction algorithms were used for a standard fingerprint template, but varying values for x -coordinate, y -coordinate, and angle θ were reported. When a single feature extractor¹ is used for all test data, the minutia points will be extracted similarly and the error that could be caused by use of different feature extractors will be eliminated. Hence, only one feature extractor was used in the study.

Performance, acceptability and resistance to spoof attacks are the key requirements that a biometric system should fulfil. Performance, in terms of accuracy, is what this study will take into consideration. MOC is already widely accepted and used [7]. Similarly, fingerprint recognition is widely accepted as a biometric feature [8]. Securing the system is a different study on its own and is not assessed in detail in this study.

Two main minutiae-based matching algorithm approaches have been identified by various researchers. They are local and global matching, with the “neighbourhood-based algorithm” and the “alignment-based algorithm” falling into each category respectively [2]. The other algorithmic approaches contain the basic structure of the main algorithms and they will be described in the literature review. Two algorithms from each main category will be used.

Since the study is focused on analysing MOC algorithms, the process of selecting the minutiae extractor may be limited to some extent. Minutiae extraction algorithms yield different results, which could affect the accuracy of matching. However, for this study, in order to deal with this limitation, minutia points used will be from the same feature extractor. This is to avoid variability, which implies that minutiae will remain constant throughout this study.

The smart-card environment has limitations like small memory and number of read/write actions allowed on a card, which make it cumbersome to implement, simulate and test algorithms on-card. As a result, Python will be used for testing the algorithms as the research focus is on algorithm performance. Python is an open-source high-level

¹ A program that extracts minutiae features from a fingerprint image

computing language. This language makes it easy to execute and integrate computations and visualisations.

1.7 Significance and rationale of the study

Yoshida and Hangai conducted a study on the accuracy and the problems that come with using the standard ISO/IEC 19794-2: 2005 for minutiae when automatically verifying fingerprints with smart-cards [9]. The study revealed that the compact card format had a higher EER when compared to the record format, normal card format, and an external format XYT².

If smart-cards are to be used to authenticate card-holders based on their biometrics, then it is required that the authentication module be significantly accurate. This is even more necessary for smart-card applications like a smart identity card or a medical records card, or in hybrid applications (Identity, medical record, and credit card applications all in one card), as room for error in the authentication process should be minimal.

Although it has been shown that the matching accuracy is significantly affected when using the compact-card template defined by ISO/IEC 19794-2:2011 (previously ISO/IEC 19794-2:2005), it has not previously been shown how the various algorithm approaches each differ in accuracy when using the ISO/IEC 19794-2:2011 template. Furthermore, it has not yet been shown whether the template format exacerbates the accuracy difference on algorithms and whether the format affects the different types of algorithms differently.

This study will extend the work of Tabassi et al. for the revised standard ISO/IEC 19794-2:2011 to find out how the on-card comparison format affects the matching accuracy for on-card minutiae-based matching algorithms, depending on the type of algorithm [4].

The aim of MOC is to keep an original template stored in the smart-card and accurately match it in the smart-card with a query template whenever an authentication request has been made from an authenticated terminal. The implementation of this process needs to take into consideration processing power limitations posed by the smart-card environment. Matching accuracy is highly important for any matching system. Therefore, it is important

² XYT is a simple text file format that is produced by the minutiae detector program mindtct.

that the matching accuracy should not be traded against keeping the template secure inside the smart-card environment.

The outputs of this study will help developers in making an informed choice of a smart-card matching algorithm when they want to use the ISO/IEC 19794-2:2011 compliant fingerprint minutiae representation. This will also help in applying the correct mitigation measures on a chosen category of algorithms to ensure a better matching accuracy.

1.8 Assumptions

This study will be based on the following underlying assumptions:

- i. Good quality and true minutia points are extracted;
- ii. Algorithms perform differently because of the way they are tailored;
- iii. Performance of algorithm in terms of speed can only be tested on the smart-card, but accuracy can be tested outside the smart-card – provided that the algorithm is implemented on the card exactly as it was in Python.

1.9 Dissertation layout

This dissertation consists of 7 chapters, and their layout is as follows:

Chapter 1 introduces readers to the whole study, outlining the research question, research goals and research objectives.

Chapter 2 will be a background study on literature that brought about the problem and also what has been done with the problem.

Chapter 3 will present the proposed research methodology.

Chapter 4 will deal with the design of the nearest-neighbourhood and alignment algorithms, together with the analysis of variance experimental design.

Chapter 5 is about implementation of the minutiae format converter, matching algorithms and the analysis of variance on the minutia points. The algorithms are also validated to ensure that they have been implemented according to their specifications.

Chapter 6 will present results, analysis of and recommendations on the algorithm. The matching algorithms' performance is also validated against published literature.

Chapter 7 concludes the study, discusses the present contributions and also suggests future studies.

1.10 Summary

This chapter has laid a foundation for the problem on which this study is based. The problem statement has been outlined in detail, the goal and objectives of the study have also been defined. Aspects that will not be part of the study and those that may hinder the study have been outlined in the scope and delimitations section. The next chapter will take the reader through the background and literature studies relevant to the defined problem of this study.

Chapter 2: Background

This chapter will take the reader through the concept of one-factor and two-factor authentications. The concept of fingerprint matching will be explained in detail together with match-on-card (MOC). The standard for minutiae templates defined by ISO/IEC will be also explained in greater detail.

2.1 Forms of authentication

Yahaya et al. define authentication as a process of verifying the authenticity of the digital identities of people or users [10]. For instance, when a student needs access to a university campus, he or she then produces a student card to security guards at the gate so that they can verify that the student is indeed a student at that university. The act of verifying the student's identity using the student card is a form of authentication. In computer and information security terms, authentication is any process by which a computer system verifies the identity of the user who requests access to it or its resources.

Authentication is based on one of three main forms; namely what you know, what you have and what you are [10,11].

2.1.1 Authentication based on what you know

Authentication based on what you know uses things like passwords, Personal Identification Number (PIN) codes, or phrases that a user knows for authentication. It is also referred to as knowledge-based authentication, which is predominantly used for user authentication in many applications. This authentication method is prone to being forgotten by users or guessed by impostors. Passwords that are simple and meaningful can be easy to remember, but they are vulnerable to attack. Passwords that are complex and have no meaning can be very difficult for the user to remember, although they are more secure [12]. This form of authentication needs a balance between complexity and meaningfulness, which can be hard for users to maintain.

2.1.2 Authentication based on what you have

Authentication based on what you have refers to an identity, security or access token. Examples of tokens are the identity document, a student card, bank card, remote gate

opener or an employee access card at a company. It is a physical device that aids the authentication process. Loss or theft are two main concerns with the use of smart-cards.

A smart-card, as defined by Shelfer and Procaccino, is any credit-card-sized card that contains more memory than the traditional magnetic stripe card and has an on-board embedded processor or smart chip [3]. There are two categories that smart-cards can fall into: processor-enabled card and memory card [13]. A memory card is used offline and data flow is one-directional, i.e. data can only be transferred from card to terminal. The processor-enabled card can be used both online and offline and data flow is bidirectional, as data can be written into the card and read from the card. A processor-enabled card has a chip that can perform cryptographic operations and can also run other applications that have been tailored for smart-cards.

Smart-cards are more limited in processing power, memory, and size than general-purpose computers. Therefore, applications developed for smart-cards are designed with the memory and processor power taken into consideration. Due to the limited processing power on smart-cards, applications that are designed for computers tend to run more slowly on smart-cards (if the memory is at all enough to contain the application). Thus, applications made for smart-cards must be processor and memory efficient.

A token can be lost by its bearer, and if it gets in the hands of a thief, the information could be compromised. Depending on the level of security that has been applied to the token, it may be duplicated and eavesdropped-on as well. Tokens, especially smart-cards, are considered secure because they usually have physical protection from being tampered with.

2.1.3 Authentication based on what you are

Biometrics is the science of recognising the identity of a person based on the person's physical or behavioural attributes [14]. The individual is recognised by attributes such as fingerprints, face, voice, palm prints, gait, and iris. Biometric features are the personal characteristics of a person and a biometric system or technology is used to measure and analyse these biometric features.

Although biometrics has been used for identification by people since their existence, it was officially introduced to law enforcement by Alphonse Bertillon in the mid-19th century, in order to identify criminals. To use biometrics for personal identification, seven

requirements must be satisfied. The requirements can be further divided into two categories [15,16]. The first category deals with human biometric features or characteristics that are candidates for being used for personal identification. The second category deals with the biometric systems that will be used for personal identification of the human biometric characteristics.

Biometric characteristics must satisfy the following requirements [17,18]:

- a) *Universality*, which means every person must possess that characteristic;
- b) *Permanence*, which means that the characteristic cannot be lost over time;
- c) *Uniqueness*, meaning that the same characteristic should differ from person to person; and
- d) *Collectability*, which means that the characteristic can be quantitatively measured.

Biometric systems must be satisfactory with regard to the following requirements:

- a) *Performance*, which means identification must be achievable at a certain level of accuracy;
- b) *Acceptability*, which specifies that people must be willing to accept the biometric system to some extent; and
- c) *Circumvention*, which refers to how easily the system can be fraudulently tricked.

As long as a biometric system or characteristic complies with these requirements, it can be used for identification and verification. However, there is no single biometric that will meet all the requirements of every possible application.

The use of fingerprints as a biometric characteristic is widely accepted because a person's fingerprints are unique, not easy to forge, share or misplace. Jain et al. outlined the main reasons for the popularity of fingerprints as follows [1]:

- a) They have been a success in various forensic, government, and civilian domain applications.

- b) Since hands are used for most activities, criminals usually leave their fingerprints at crime scenes.
- c) Large legacy databases exist.
- d) Compact and relatively cheap fingerprint readers are available.

2.2 Biometrics and smart-cards

Combining smart-cards and biometrics creates a connection between the card and the card-holder. It allows for a strong verification and identification of the card-holder's identity. A smart-card with biometric recognition capabilities is designed in such a way that it would not be used without the card-holder's biometric feature being verified against the stored biometric feature in the smart-card [18].

The process of verifying the user's identity makes use of a template of the user's recorded biometric characteristic at the registration time and matches it with the template that is obtained from the user at the time of the transaction request. There are three forms of authenticating users with biometrics and smart-cards [10]:

- a) Template-on-card (TOC): A person's biometric characteristic is stored in the smart-card and the process of verifying the user's identity takes place on a computer/terminal [20, 21]. The stored template is sent to the terminal from the smart-card during verification.
- b) Match-on-card (MOC): A fingerprint is captured on a terminal, then features are extracted and a query template is sent to the smart-card where the matching process will take place [19, 20]. MOC ensures that the original template remains in the smart-card environment by only sending a yes/no response to a terminal indicating whether there was a match.
- c) System-on-card (SOC): The whole system, comprising fingerprint sensor, feature extractor and matcher, is on the smart-card [20, 21]. This is the most secure, yet the least practical, especially in large-scale applications, because it is expensive.

Storage of biometric data may take place on a central database or on the smart-card. When storing the biometric data on a central database, it means that whenever an

authentication process has to occur, there must be a network connection to the database from the terminal used for the authentication process. Without such a connection (which indeed is not always guaranteed to be available), the central database cannot be accessed.

In September 2015, about 5,6 million United States federal employees' fingerprints were stolen. Information about these incidents can be read from the American monthly magazine *Wired* and the popular daily newspaper *The Washington Post*. This story is evidence of the dangers that come with using a central database. The risk posed by this danger is that the confidentiality of data belonging to millions of people is compromised all at once [21].

To avoid compromising confidential data, the smart-card is used to store biometric data. That way, if the smart-card is lost, there is only one biometric record that could be compromised. Since MOC is cheaper to use than SOC, it is then the optimal choice of smart-card for large-scale applications [7].

Processor-enabled smart-cards are similar to a computer, as they also have an embedded memory and processing power, and also constitute an operating system. The difference is that the smart-card possesses only limited resources in terms of memory, processing power, and the absence of human computer interface peripherals. Although they can operate like a computer, i.e. run applications, their limited memory space and computing power has to be taken into consideration when deploying programs and data into smart-cards.

It is widely acknowledged that smart-cards are one of the most secure and reliable forms of electronic identification [18]. That is, when compared to biometrics which takes second place, and shared secrets, which come last. Shared secrets (like passwords) are widely accepted as a form of authentication which is still used even today, as they are easy to implement. However, using a synergy of the three forms of authentication can be most reliable as now it is a multiple-layered authentication.

2.3 Fingerprint recognition

Fingerprint recognition is the process whereby a biometric system is used for identification and verification of fingerprints. The key difference between verification and identification is

that in the case of identification, there is a pre-enrolled database of fingerprints and during the identification process a query fingerprint will be searched against a fingerprint database (e.g. with 1 million fingerprints), and the fingerprint will be identified as the one that it matches with in the database [22].

In fingerprint verification, it is confirmed or verified that persons are really who they claim to be using their fingerprint characteristics [22]. For example, when opening a bank account, the bank personnel will request a client to insert his or her identity card into a terminal, which after authenticating the terminal via secure messages, will ask for the client's fingerprint to be scanned on the terminal, and then verify if the fingerprint scanned is similar to that which is stored in the card. In verification, the query fingerprint along with the user credentials such as ID number will be provided. The database will be searched against the ID number and the fingerprint will be pulled and compared against the query fingerprint to establish a match.

2.3.1 Minutia points

Every person has a unique fingerprint. It is these prints on our fingers that are used to distinguish humans from each other. The finger patterns on our fingers form into different shapes that are used to classify fingerprints. The basic patterns are whorl, loop, and arch. The lines that form these patterns, called ridges, are the ones used to collect minutia points [23]. The minutia points that are extracted on a fingerprint image are the x - and y -coordinates of the location where the ridges end and where the ridges bifurcate as shown in Figure 2-1 below. Given a fingerprint image with length and height ($l * h$), a minutia point is the geographical location on the image where the ridge bifurcation or ridge ending is found.

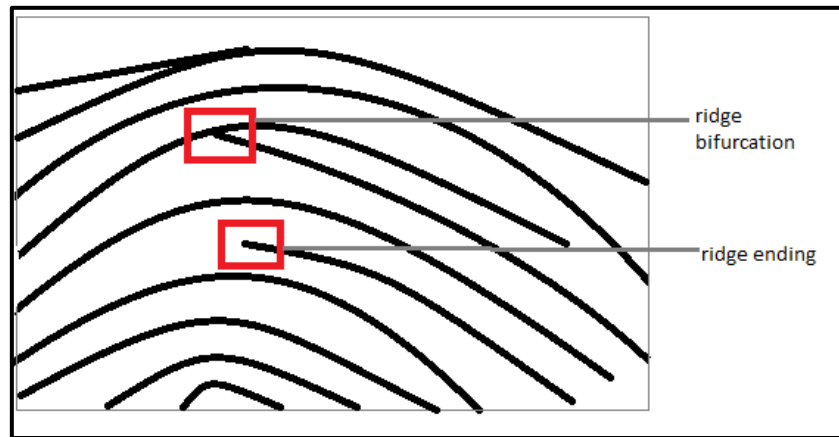


Figure 2-1: Extracted fingerprint features (minutiae)

2.3.2 Basic enrolment and query process

There is a common process flow for a fingerprint recognition system where a user's fingerprint image is obtained through a sensor, then features are extracted from the image and then subsequently stored in a database. The process flow is explained in [1,11,13] and is presented in Figure 2-2.

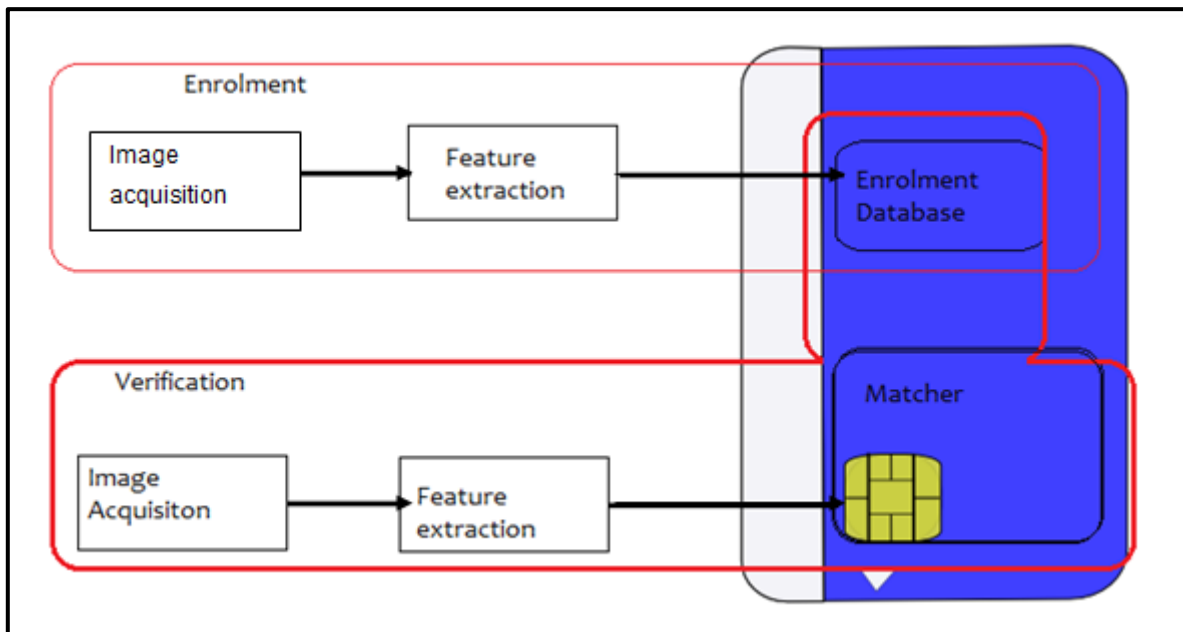


Figure 2-2: Basic enrolment and query process

In the enrolment phase, a fingerprint is enrolled into a database or stored inside a smart-card. A subject's (person's) finger is scanned through a sensor and the finger template is stored typically as a digital greyscale image. A greyscale image is an image with only

shades of grey, from white to black. The greyscale image is then sent to the feature extractor so that minutiae are extracted from the greyscale image. After extraction, minutiae are stored in the enrolment database.

During a verification phase, the above steps as in enrolment are taken, except that the query template is sent to the matcher and the original template is requested from the database where it is stored and then they are used for the purpose of matching. The image acquisition and feature extraction step occur in both the enrolment phase and query phase and should ideally be done using the same tools, to improve performance.

2.4 Conformance to standards

ISO/IEC define a standard as a “document that provides requirements, specifications, guidelines or characteristics that can be used consistently to ensure that materials, products, processes, and services are fit for their purpose” [24]. Standards aid in the development of interoperable systems, products, services, and materials.

When minutiae are extracted from fingerprint images, the minutia points should conform to a certain standard that will enable them to be used in certain applications. ISO has produced a standard (“ISO/IEC 19794-2:2011 - Information technology -- Biometric data interchange formats -- Part 2: Finger minutiae data”) that gives a guideline on the format of the fingerprint template to be used in an MOC environment and the normal computer/terminal environment. The ISO-defined format for minutiae consists of two different formats: record format and on-card comparison format. The record format consists of a general header for each finger record, ridge count data, core data, and delta data. Zonal quality data and defined data are also provided for. Each minutia point can be represented with five or six bytes. The record format is typically used for fingerprint matching on a computer or smart-card reading terminal.

The on-card biometric comparison format does not make use of the representation header and it only has a facility for a single representation record of the finger. This format resembles the compact-size card format that was defined in ISO/IEC 19794-2:2005 as it only needs three bytes for representing each minutia. This size is almost half the size of the record format.

Table 2-1 depicts the minutiae location and angle value varieties and similarities as in the 2005 and 2011 definitions.

Table 2-1: Definition of coordinates of minutia location (adapted [9])

Format	Resolution (Pixels Per Inch (ppi))	Coordinate	Field type	Resolution (degrees)
ISO/IEC 19794-2:2005 record format	Depends on FP image	1 pixel, integer value	8 bit	1,40625
ISO/IEC 19794-2:2005 card format (normal size)	Fixed 2540 ppi	0,01 mm, 14 bit binary	8 bit	1,40625
ISO/IEC 19794-2:2005 card format (compact card)	Fixed 254 ppi	0,1 mm, 8 bit binary	6 bit	5,624
ISO/IEC 19794-2:2011 record format	Fixed 250 ppi	0,1 mm	8 bit	1,40625
ISO/IEC 19794-2:2011 on-card format	Fixed 254 ppi	0,1 mm	6 bit	5,625

2.5 ISO/IEC template use and MOC

The use of fingerprint minutiae templates addresses the size and privacy concerns associated with the storage of raw fingerprint images. The templates of minutia points are more space effective, less resource intensive, and more cost effective [4]. This is because only minutia points' coordinates are stored as numbers and they require less space whereas a normal fingerprint image is stored as an array of pixels and requires more space (average of 200 kB), depending on the size of the image. A data interchange format is considered to be sufficient when the information that is encoded in a standard template, successfully facilitates a fingerprint recognition [4].

A study on interoperability of minutiae templates was conducted by Tabassi et al. [4]. The aim of that study was to assess the adequacy and performance of standards when using several evaluations. The evaluations used were organised to quantify interoperability and performance degradations of fingerprint matching systems when using standard templates compared to proprietary templates. The standard template that was used in this study was the International Committee for Information Technology Standards (INCITS) 378-2004 standard defined template, and it was compared with proprietary templates. The study revealed that error rates are lower when proprietary templates are used and are higher when standard templates are used. This study also revealed that, when the template and the matcher came from different suppliers, the error rate was higher.

These results show that according to the minutiae extraction algorithms, extracted standard minutiae tend to differ from one another. Furthermore, due to the differences in how the matching algorithms function (in terms of parameters taken and how the parameters are used); proprietary templates offer a better matching accuracy.

An evaluation plan on fingerprint match-on-card algorithms was made by Grother and Salamon [26] with the primary objective of ascertaining the capability of MOC by measuring fingerprint algorithm accuracy and speed in ISO/IEC 7816 smart-cards. This study proposed that a mechanism for MOC testing be instantiated and the accuracy of MOC be measured using the ISO/IEC 19794-2:2005 compact-card minutiae templates, and then formulate comments on possible revisions of the relevant standard. The templates to be used in the evaluation were the INCITS 378, the ISO/IEC 19794-2:2005 compact card, and record formats. Record format is similar to the INCITS 378. Hence, the evaluation plan was to compare the INCITS 378 format against the ISO/IEC 19794-2:2005 compact card format.

The Minutiae Interoperability Exchange Test (MINEX) II is an assessment of MOC technology aimed at evaluation and development of the capabilities of fingerprint minutia matchers running on ISO/IEC 7816 smart-cards.

MINEX II trial phase I, phase II and phase III were conducted, following the evaluation plan, and results were published in a report [28, 29]. In the conclusions presented, it was

observed that for all matching algorithms the False Non-Match-Rate³ (FNMR) increases. For cases where less than 20 minutiae are used, FNMR increases by an order of magnitude from approximately 2 % to 20 %. Some systems significantly improve and some degrade as the number of minutiae decrease [27]. An increase in False Match Rate⁴ (FMR) due to fewer minutiae is alarming because security could be weakened.

Some factors that are suspected of affecting the accuracy of matching are the following:

- the number of minutiae that are accepted by the card;
- the number of verification attempts;
- the number of finger presentations allowed;
- the decision-making policy;
- the number of fingers used;
- the sensor and enrolment policy;
- habitation and demographics.

The performance of the ISO-CC template approaches that of the INCITS 378 template, although performance is degraded for some implementations. The degradation, which is only evident in some implementations, could be a result of the implemented algorithms' sensitivity to the way that the templates are structured, as it was noted that the sensitivity of the matching algorithm could affect performance [26]. Yet, "MINEX II did not evaluate the interface standards, secure transmission protocols, nor card or algorithm vulnerabilities" [27].

2.6 Different minutiae-based fingerprint matching algorithms

Minutiae-based fingerprint matching is an approach wherein minutiae patterns that were extracted during the minutiae extraction phase are matched [2]. There are different minutiae-based fingerprint approaches, namely local features-based and global features-based. The global features-based algorithm looks at the overall fingerprint to make

³ FNMR (false non-match rate) is the rate at which a system does not match fingerprints from the same finger.

⁴ FMR (false match rate) is the rate at which a system falsely matches fingerprints that do not belong to the same finger.

decisions, whereas the local features-based algorithm looks at the local relationship between minutia points [28].

2.6.1 Global features-based algorithms

The global features-based algorithms can be classified into alignment methods and work-sharing methods and are explained in what follows:

2.6.1.1 Alignment methods

The alignment methods use the global features of a fingerprint to complete the matching process. Generally, the minutiae features are rotated and translated until a match is found, i.e. aligning the features in order to find the best match score between the features.

In 1997, Jain et al proposed a minutiae-based fingerprint matching system with four general steps [29]. Other papers were also written, explaining the same algorithm, by Jain et al. and Sanchez-Reillo and Sanchez-Avilla [32, 33]. In the algorithm, a pairwise similarity between the minutiae of two fingerprints is computed. The two fingerprints are then aligned according to the most similar minutiae pair. Minutiae correspondence is established and the similarity score of the minutiae is computed to determine the degree of match between two fingerprints. Factors that affect the match score are the number of matching minutiae, the consistency of the ridge count between matching minutiae, and the percentage of matching minutiae in the overlapping area of two fingerprints.

In 2003, Pan et al. proposed a fingerprint matching algorithm with ultra-low memory consumption [34, 35]. In this algorithm, a multi-resolution accumulator array⁵ and the loop blocking technique are used. The multi-resolution accumulator array helps in having an equal amount of memory space required at each resolution notwithstanding the increasing time. The loop blocking technique saves more memory by improving cache performance.

In the algorithm as shown in the flow diagram of Figure 2-3, the input is a query template minutia and a reference template minutia. The algorithm starts by considering the target system and setting initial parameters of the accumulator array together with the size of the accumulator array. Minutiae pairs for all the input minutiae, i.e. reference and query, are generated and their differences are filled into the accumulator array. Positions of the

⁵In a procedure called Hough transform, an accumulator array is used to store votes of imperfect instances of objects within a certain class of shapes.

maxima of the array are found and then the matching score is computed after the alignment process is done according to the estimated parameters.

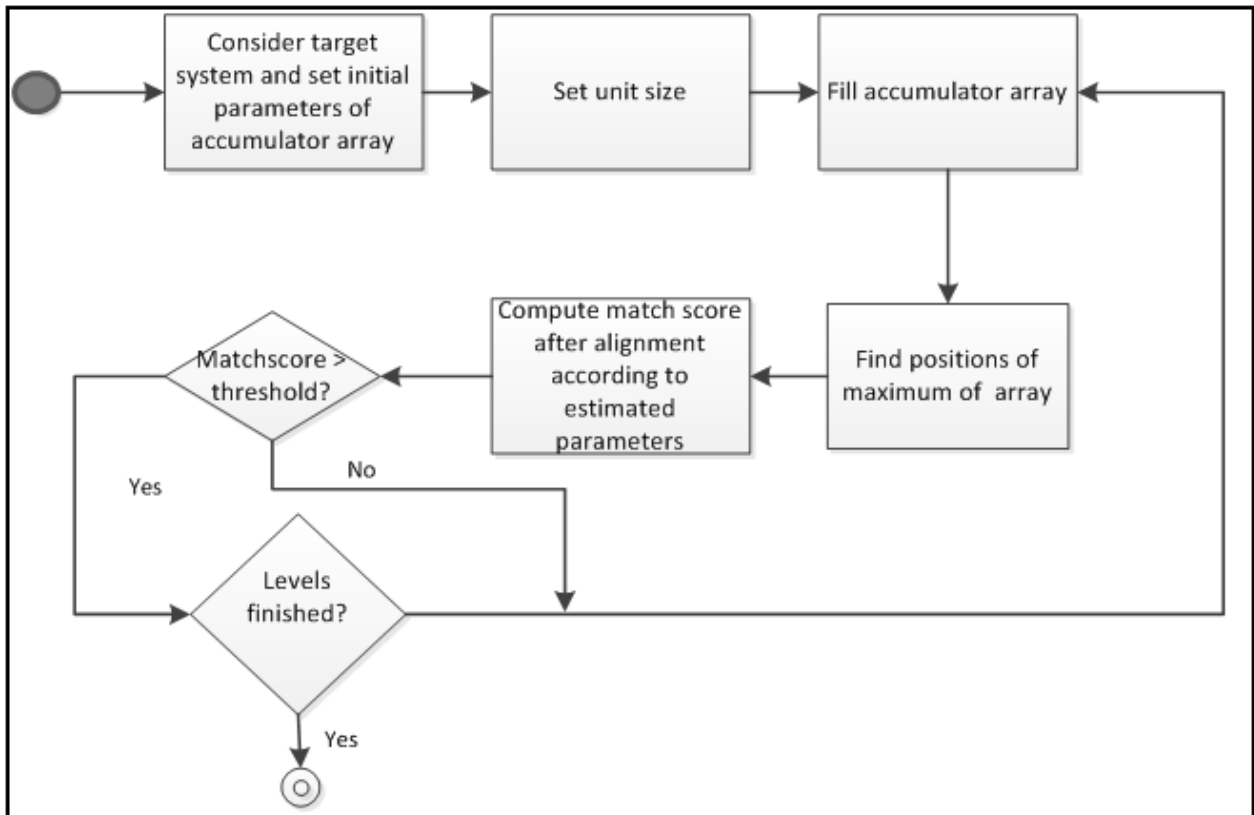


Figure 2-3: Flow diagram of an ultra-low memory fingerprint matching algorithm and its implementation on a 32-bit smart-card.

The alignment parameters are estimated after several transformations and matchings have been performed. Given two minutiae templates, it is assumed that the second fingerprint image can be obtained by applying transformations to the first image. The transformations assumed are similarity transformations such as rotation, translation, resizing, and reflection. The set of all possible transformations is discretised and then for each transformation the matching score is computed. The correct transformation is then assumed to be that one which produces the maximal matching score.

The authors used fingerprint images with a resolution of 500 dpi (dots per inch) and a size of 248*292 pixels. Four fingerprint images per one finger were used from 100 participants. Experimental results of the algorithm confirmed a memory requirement of 4,8 kB Random

Access Memory (RAM) and the Equal Error Rate⁶ (EER) is comparable to that of a typical algorithm [32]. The exact accuracy of the matching algorithm was not reported.

A “secure and fast fingerprint authentication” mechanism for smart-cards using reference point location was proposed by Moon et al. [34]. The proposed algorithm takes as input ridge endings and ridge bifurcations. Minutiae extraction, template selection, and template alignment all occur first in a computer, and then the aligned template is sent to the smart-card for matching. The reference point used in this algorithm is determined with a *sine* component filter. Each feature vector contains the distance between minutia and the reference point, minutiae orientation and rotated orientation. Minutiae are regarded as matching if their distance and threshold⁷ lie within some threshold values.

Chouta et al. proposed an MOC algorithm with an improved accuracy, regardless of fingerprint image distortion [35]. The presented algorithm is transformation/alignment based and has three main phases, namely, smooth registration, Gaussian minutiae pairing and duplication cleaning. These main phases will be explained in detail in the following paragraphs.

Smooth registration is the portion of the algorithm that focuses on aligning the fingerprint images together so as to match them. In the smooth registration phase, an assumption is made that two minutia pairs are not always identical. Therefore, a tolerance value is set for the transformations. For minutiae represented as a triplet (θ, x, y) ⁸, registration is administered by iterating through all minutia points in the input and reference minutiae sets and obtaining an affine transformation⁹ that maps the actual processed pair. Transformations at all tolerance/threshold values are collected and a histogram is created where a peak can be detected for genuine comparison attempts.

In the second phase of Gaussian minutiae pairing, a number of overlapping minutiae are computed between each set. This is where the actual matching process is done. A bounding box is used with a threshold to find out the difference between two minutiae,

⁶ EER (Equal error rate) is the value where False Match Rate (FMR) and False Non-Match Rate (FNMR) are equal.

⁷ A threshold is the point at which the certainty of a biometric sample matching a particular reference template becomes reasonable.

⁸ (x, y) are the 2- dimensional minutia coordinates on the fingerprint image and θ is the direction in which the minutiae coordinate is pointing.

⁹ An affine transformation is any transformation that does not cause loss of collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distance.

helping in dealing with distortion of fingers. Smooth Gaussian scalar thresholding is used to give the algorithm a higher flexibility with regard to malformed fingerprints.

In duplication cleaning, the algorithm ensures that there are no duplicates in the matching pairs. For example, minutiae pairs (1, 3) and (3, 1) signify the same matching pair. This pair, if counted twice, would give a higher and uncertain match score than it would have been when only one pair was counted.

2.6.1.2 Work-sharing methods

Alignment methods come with computing power demands as the minutiae records have to be rotated and translated numerous times while looking for a match. Work-sharing methods have been created with the aim of decreasing the computational power requirements that come with alignment algorithms. They decrease the computational power by performing the alignment process outside the smart-card using information about the original template that cannot be used to reconstruct it. The alignment parameters are then sent to the card to align the query template and compare it to the reference template.

Lam et al. proposed a pre-alignment algorithm for hybrid MOC systems in 2007 [36]. There are two steps that are presented in the algorithm, namely the registration step and the pre-alignment step. In the registration step, a captured fingerprint image is enhanced¹⁰, binarised¹¹, and segmented¹². Then minutiae and a reference point which is the core are detected on the image. Five different 35*35 pixel regions are extracted from the fingerprint binary image, with the first region centred at the reference point and the other four centred at minutiae around the reference point. The five regions, represented as a triplet according to the ISO/IEC SC37 standard, are stored in the smart-card in conjunction with their normalised x-y coordinates and the rest of the standard minutiae information.

In the pre-alignment step, a query is made to the smart-card to match minutiae. The stored triplet of five regions is presented to the host PC together with the corresponding normalised x-y coordinates. This step finds similar data to those stored in the smart-card from the query fingerprint image. The five regions are then rotated and correlated in order

¹⁰ Enhancement – a process of applying techniques to emphasise fingerprint images in order to facilitate the identification of ridge-valley structures and hence their features

¹¹ Binarisation – a process whereby a greyscale image is converted so that only black and white colours are visible and grey is removed to either the black or white, using a certain threshold

¹² Segmentation – the task consists in separating the fingerprint area from the background

to find the best correlation score that will be used as the alignment point. The best correlation score is then used to transform the query minutiae before they are sent to the smart-card for comparison.

Using FVC 2000 and FVC 2004 – fingerprint databases provided by Fingerprint Verification Competitions from the years 2000 and 2004, the algorithm produced the best average True Accept Rate¹³ (TAR) of 93 % and lowest average TAR of 75 % respectively. FVC 2004 has a large number of distorted images and that could be the most likely reason for the poor performance of the algorithm when using that database. This is an indication that distortion of fingerprint images can affect the fingerprint matching quality. The feature extractor and fingerprint matcher have to be implemented effectively to handle distortion on fingerprint images, in order to improve the fingerprint matching process.

The authors Chen et al. present a novel work-sharing approach to on-card fingerprint matching [37]. The algorithm performs off-card reference minutiae computation and off-card query template alignment. The system flow diagram is presented in Figure 2-4 below:

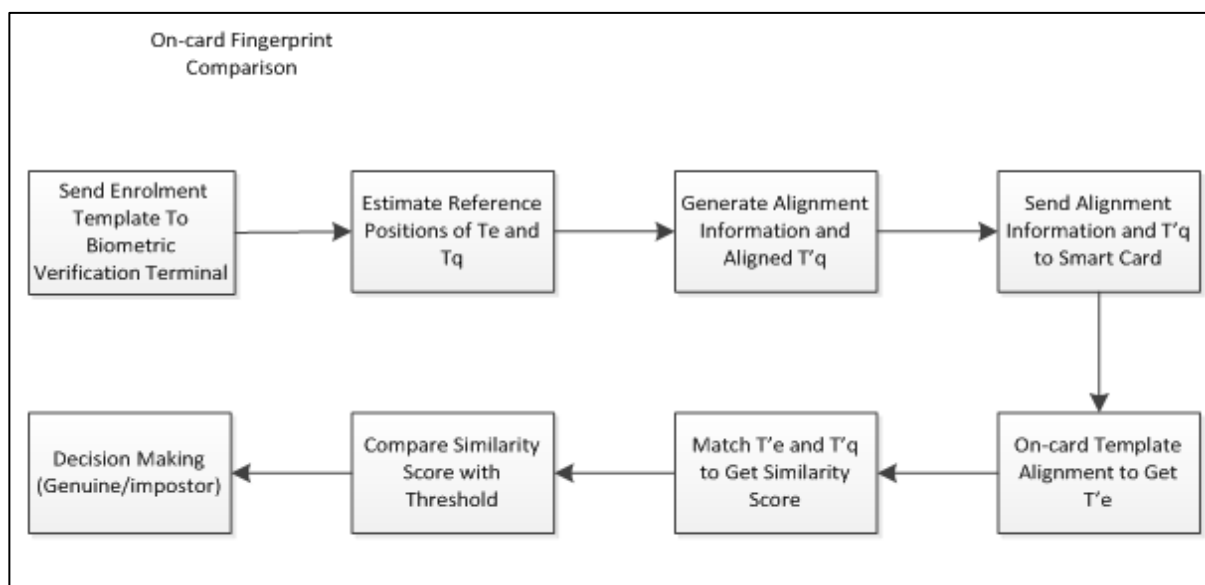


Figure 2-4: ISO/IEC standards for on-card biometric comparisons

During enrolment, a fingerprint template is created and it is divided into two portions: a mandatory and an open template (Te and Te_open). Both templates are stored in the smart-card, but Te is never exposed to the public. The open template contains relative information of a minutia's three neighbours.

¹³ True accept rate is the percentage of times a biometric system correctly verifies a true claim of identity.

During verification, a query fingerprint template is created using an open and mandatory template of the query finger (T_q and T_{q_open}). The biometric verification terminal sends a request to the smart-card, which then returns T_{e_open} to the BVT. The BVT performs reference minutiae search and alignment of the T_q . The aligned T'_q with T_{q_open} and alignment information are further sent to the smart-card so that T_e can also be aligned in the smart-card.

Once the templates are aligned, T'_q and T'_e are then ready for matching. In the matching process, all minutiae in the transformed templates are scanned for the matching pairs whose matching error is lower than a predefined tolerance. A match score is subsequently computed by dividing the average number of minutiae between the enrolment template and the query template.

The algorithm is performed with an average verification time of 2,5 seconds. An 8-bit Java Card¹⁴ was used for the experimentation and it produced an average EER of 4,3 % using the FVC2000 and FVC2002 databases. The authors claim that the performance is comparable to algorithms running on a PC [37].

Pre-alignment of minutiae outside the smart-card could be beneficial to the matching process because there is better memory and processing power on a terminal than on-card. It would make the matching process faster, and that is an advantage for real-time applications such as access control and retail.

Sending of information about minutiae to a terminal could make the original template vulnerable, as now it is just a matter of time before that information can be used to reconstruct the original template. Unlike passwords, fingerprints cannot be changed, so a person's identity as a whole is being endangered if the original template is reconstructed.

This work-sharing method presented by [37] is also an amalgamation of MOC and TOC. This is because it inherits some of the advantages and disadvantages of using either one of MOC and TOC. In the case of TOC, the template has to be sent out from the card to the terminal. But in the case of work-sharing, only certain information about the template is sent into the terminal for computation of the alignment parameters. That saves processing

¹⁴ Java Card is a software technology that allows Java based applications to be run securely of smart cards.

costs that would be incurred by the card if the alignment procedure was to be done on the card. In the case of MOC, the final matching procedure is performed in the card.

2.6.2 Local features-based algorithms

The nearest-neighbourhood and finger-code algorithms use the local structure of minutiae to match fingerprints. The local relationship between minutia points is used to infer whether minutiae templates come from the same finger or not.

2.6.2.1 *Nearest-neighbourhood algorithms*

An asymmetric fingerprint-matching algorithm for Java Card was proposed by Bistarelli et al [38]. This algorithm is based on minutiae characteristics and their local structure information. The local structure information removes the need to pre-align the fingerprint templates. For every minutia point in the template, its 8 nearest neighbours with respect to Euclidean distance¹⁵ and good reliability estimation are selected.

The neighbours in this algorithm are described using the Euclidean distance between the central minutia and its neighbour minutia, the angle between the central minutia and the central-minutia ridge direction, the difference angle between central minutia and neighbour-ridge orientation angle, and the ridge count between central and neighbour minutiae.

The matching procedure receives the input of the entire reference template and neighbour features information for matching the candidate minutiae one-by-one. The list of minutiae in the template is stored according to increasing minutia reliability values. The algorithm computes how closely the neighbourhood of a minutia in the query template is similar to the neighbourhood of each minutia in the reference template.

The algorithm takes as input the minutia M2 in reference template and minutia M1 from the candidate template. Then neighbour N2 of M2 and neighbour N1 of M1 are compared to see if they are a match. If the neighbours are not matched before, the neighbourhood dissimilarity between the two neighbours is calculated and also taken through a bounding box to find out if they are within the box. If they are not within the neighbourhood

¹⁵ Euclidean distance is the straight-line distance between any two points.

dissimilarity score is discarded. However, if they are within the bounding box, then $N2 - N1$ are matched with the best neighbourhood dissimilarity value.

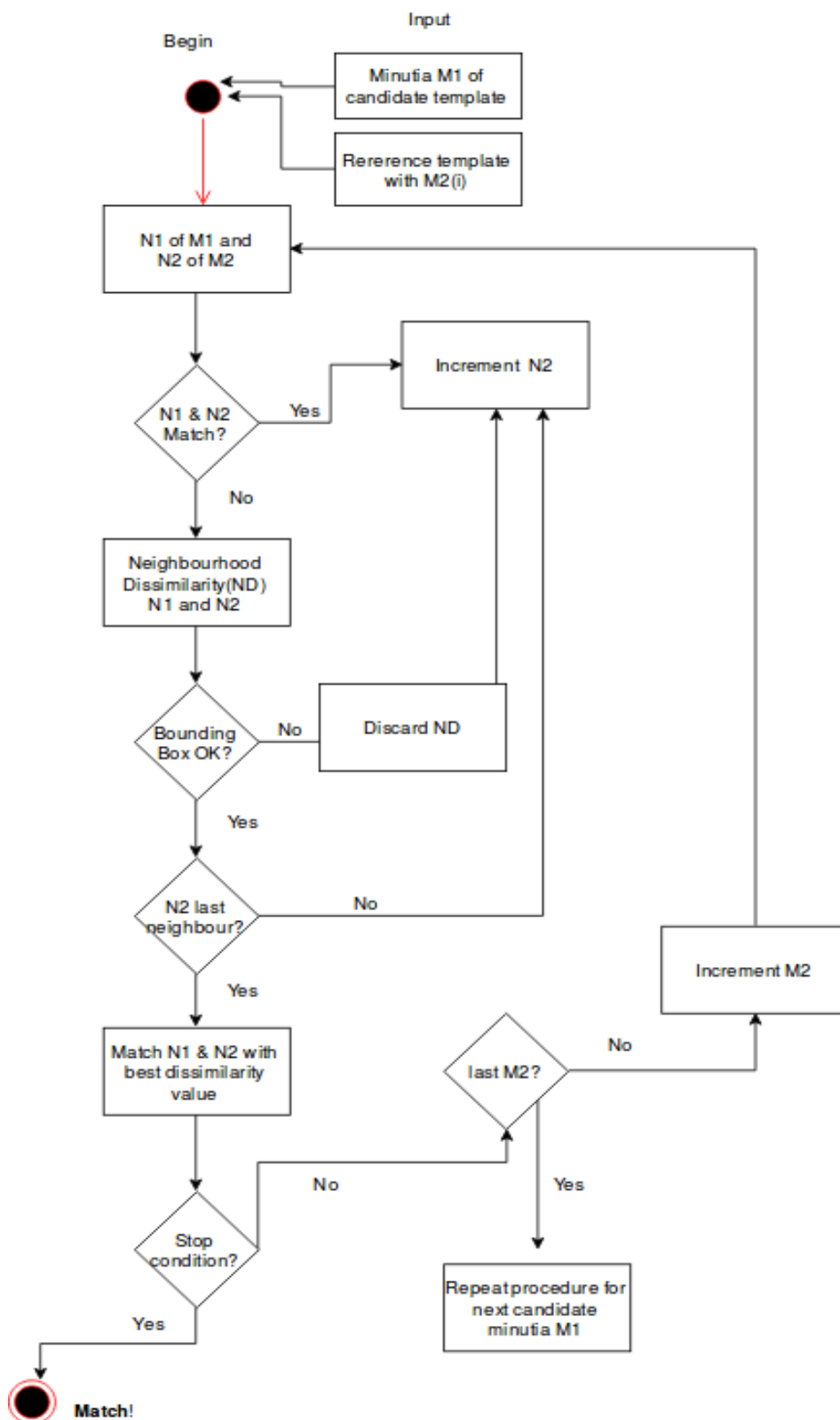


Figure 2-5: Flow diagram of the nearest-neighbourhood matching algorithm for a Java Card

There are four steps taken to compute the dissimilarity between two minutiae in different algorithms using neighbourhood features as information. First, the difference in absolute value between corresponding features is computed. Then, every feature difference is checked to see if it is below the corresponding acceptance threshold. Furthermore, each feature difference is multiplied by a weight factor to get a relative weighted value, which is necessary to show up more features that match better. Finally, the global dissimilarity is computed by summing up all the four weighted differences.

The algorithm was developed in an 8-bit Java Card. The FVC2002 database of fingerprints was used and the algorithm achieved a lowest EER of 0,48 % and a highest EER of 8,5 % [38]. It was observed that a good quality enrolment image improves both reliability and speed. Figure 2-5 depicts the flow of this algorithm.

2.6.2.2 Finger-code algorithms

An algorithm for fingerprint matching in smart-cards using finger code and minutia points was developed by Cucinotta et al. [39]. The proposed algorithm makes use of finger code and minutia points to verify an identity. A circular 120-pixel region centred on the fingerprint core is used to extract the finger code. The finger code is then normalised and Gabor filters¹⁶ are applied, then it is tessellated¹⁷ into rings and sectors.

The matching score of two images is calculated as the sum of absolute differences between the values associated with corresponding tessellations. The point-pattern matching algorithm inspired the minutiae-matching procedure used. The algorithm counts the number of neighbourhood pairs that match under each possible rotation, using a common reference point from which the matching process extends towards the periphery. This algorithm performs with an EER approximated to 1,4 %, which is relatively lower than the EER values obtained by the minutiae and finger code matchers which are 3 % and 4 % respectively.

An embedded fingerprint-matching algorithm on smart-card was proposed by Benhammadi and Bey, and the flow diagram is illustrated in Figure 2-6 [5]. This algorithm makes use of minutiae neighbourhood information and finger code to match fingerprints.

¹⁶ A Gabor filter is a linear filter used for edge detection.

¹⁷ Tessellation is the tiling of a plane using one or more geometric shapes, called tiles, with no overlaps and no gaps.

This approach can be divided into two main steps, viz. a finger code generation process and a minutiae-matching process.

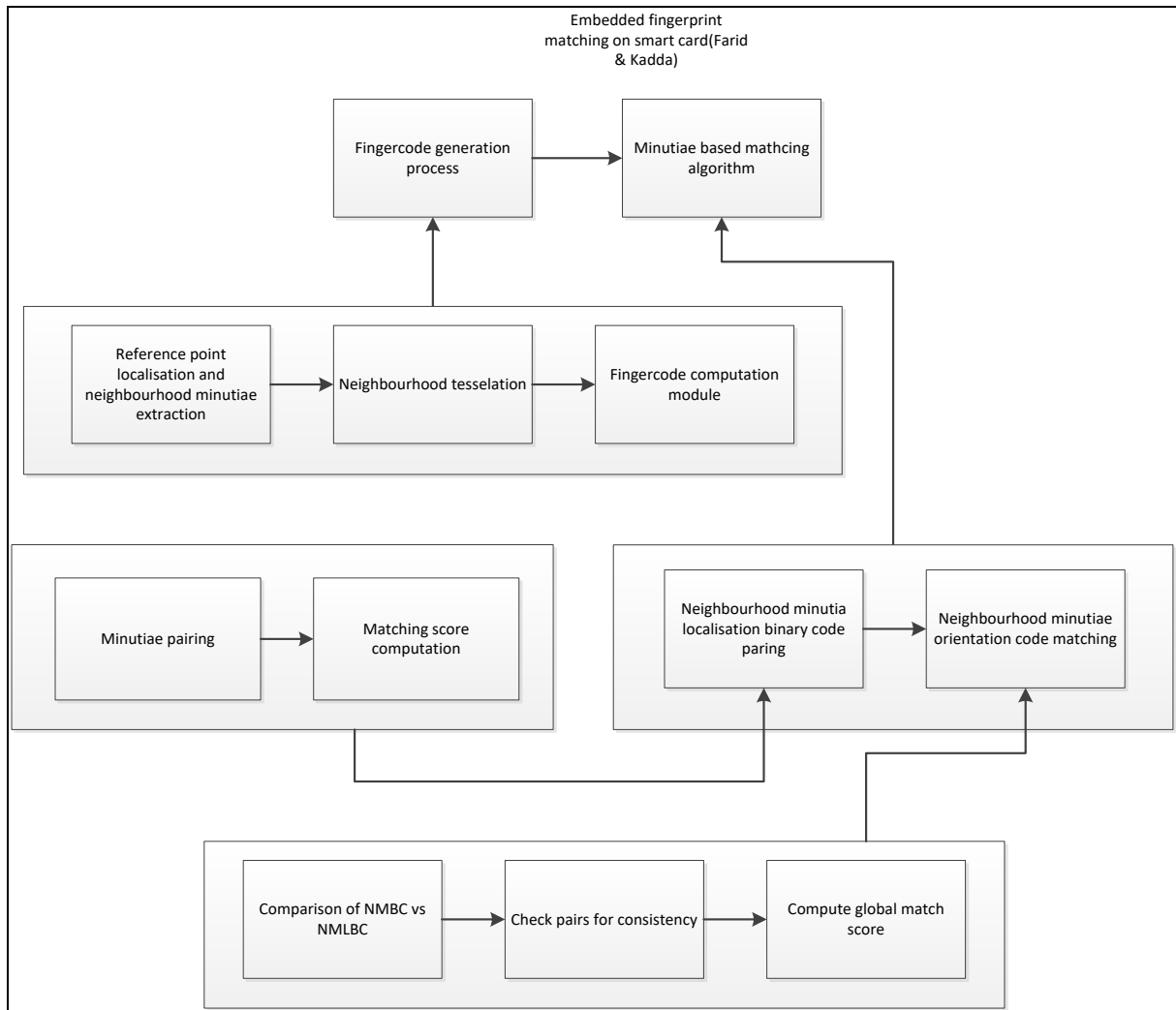


Figure 2-6: Finger-code matching algorithm flow diagram.

In the finger code generation process, a reference point of the fingerprint is extracted together with neighbourhood minutiae near the point. The extraction process is done on thinned¹⁸ or greyscale¹⁹ images. The thinned images are then tessellated to compute the binary component of the finger code feature vector. The finger code is then generated by computing neighbourhood minutiae localisation binary codes associated with their orientation features.

¹⁸ Thinning is a stage which causes the ridge line thickness to be reduced to one pixel

¹⁹ Greyscale image has a range of shades of grey without apparent colour. The darkest possible shade is black and the brightest possible shade is white, and other light intensities are in between.

The finger code features are then used in two steps for the minutiae-based matching algorithm. First, the neighbourhood minutia localisation binary code (NMLBC) in the input is paired with the NMLBC in the template fingerprint to produce a list of the best-matching NMLBC pairs. The NMLBC are then checked for consistency; they are considered to be consistent if the corresponding relative minutiae orientations are below a given orientation threshold.

The algorithm was tested on a Java Card using the FVC2002 DB1a and DB2a database. The error rates obtained were EER values of 6,28 % and 4,49 % using DB1a and DB2a respectively. In the results, ridge counts above 8 seemed to return a higher EER due to elastic distortion of fingerprint images.

2.7 Summary

The first ISO/IEC 197942-2 standard for minutiae template had two representations for on-card matching in the first edition of the standard in 2005, but only one remained in the second edition of the standard released in 2011. When tested for usability, the ISO/IEC template for on-card comparison proved to be usable, although it performs with accuracy less than the record or the normal sized template. This raises questions of whether the loss of resolution affects the accuracy for each type of matching algorithm or not. The suspicion that the template was causing a reduced accuracy is strengthened by the loss of resolution that occurs when the minutia points are encoded into the on-card comparison template. Although studies have shown the effect of the minutiae on match-on-card algorithms, it has not been clearly studied and established whether MOC algorithms are affected differently by the on-card comparison template and what the depth of the effect was on the algorithms. This study sets itself the aim to clear up these uncertainties. The next chapter will present the methods that are used to establish the missing information.

Chapter 3: Research Methodology

This chapter presents the methodology to carry out the research study in order to find out which minutiae-matching algorithm approach between the global and local-feature based algorithms is affected most by the ISO/IEC 19794-2 template for on-card matching. An overview of the research methodology is presented in the following block diagram:

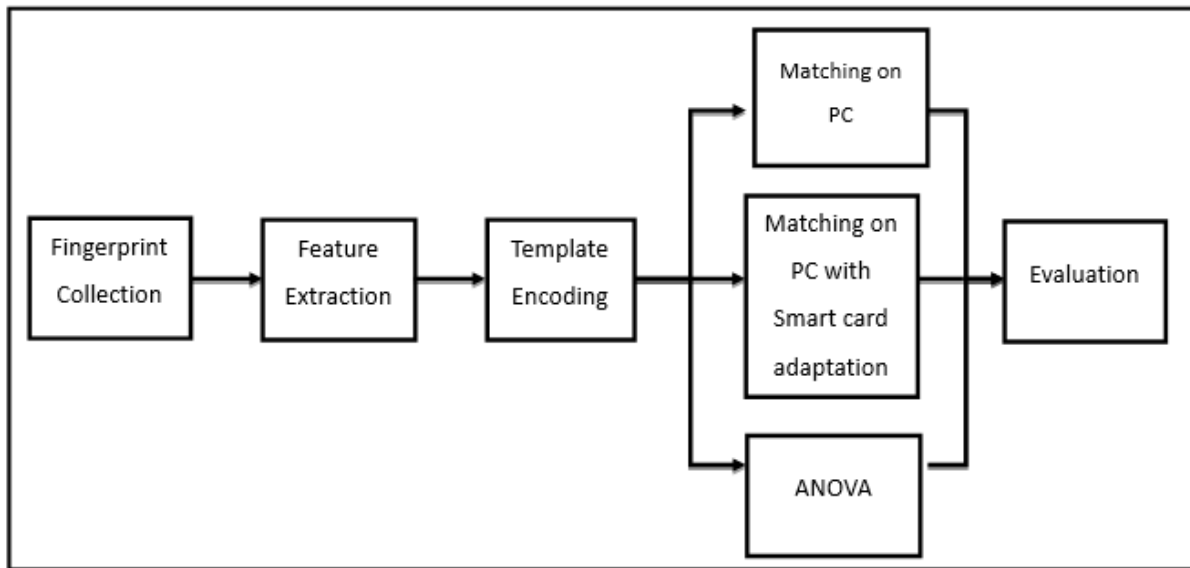


Figure 3-1: Research methodology

3.1 Collecting fingerprint images

Collecting fingerprints comes with high costs and significant time requirements, depending on the size of fingerprint images to be collected. In South Africa, the collection and processing of demographic information must also comply with the Protection of Personal Information (POPI) Act [40]. The POPI Act offers severe penalties if one does not comply with its conditions while processing personal information. As a result, when one takes the route of collecting fingerprints, even just for research, there will be legal regulations to be complied with.

In addition, the collection of fingerprints may be a difficult and time-consuming task. This can be influenced by the fact that the researcher has to find people who will volunteer to offer their time. People are generally happier doing things if there is a reward. The point was already made by Dale Carnegie [41] who showed that people will continue doing something if they feel recognised and appreciated. This now adds the factor of cost to a

researcher, regardless of the reward system chosen. If there is no incentive for volunteers to provide fingerprints, there is a chance that not many people will participate in the collection of fingerprints.

A number of databases have been created by different research groups which are available for public use. These databases allow for a common and repeatable benchmarking of algorithms. Making use of the public databases saves a lot of time and effort and will allow a researcher to focus on what needs to be done with the fingerprints instead of trying to find ways of collecting them.

Fingerprint verification competitions have pre-generated fingerprint databases such as FVC2000, FVC2002, FVC2004 and FVC2006 [44, 45, 46, 47]. These fingerprints have been acquired under different conditions. Heterogeneous populations have been used and, on some databases, displacement, rotation, and distortion have been exaggerated in order to make the databases cover a wide range of the population. These databases are freely available for use and they will be used for this study. Each database from 2000 to 2004 consists of image groups captured with optical and capacitive sensors, and one group of synthetic fingerprints. In 2006, two more sensor types were used. They are the electric field sensor and the thermal sweeping sensor. Due to the variety of sensors and capturing conditions, the FVC fingerprints will model the real-world fingerprint capturing conditions.

3.2 Extraction of fingerprint minutiae

Minutiae extraction is the process of extracting ridge endings and ridge bifurcation points from a fingerprint image. This process relies heavily on the image quality. Hence, before extraction commences, image enhancement steps are taken to ensure that the fingerprint images are of a good enough quality and minutiae are easily detectable.

A number of open algorithms for minutiae extraction have been proposed, and they use different approaches to extract fingerprint minutiae. One openly available extraction tool has been created by the National Institute of Standards and Technology (NIST); it takes

RAW²⁰ format images and outputs the minutia points [46]. Major functional steps of the algorithm are presented in Figure 3-2. An *input raw fingerprint RAW image* takes a RAW format image and creates *image maps* from the image. The image is then binarised to have only two colours (black and white). When the image is binarised, minutiae are easily detectable from the image. Not all minutiae detected in the detection step are true minutiae; consequently, false minutiae are removed. Quality values are then assigned to the minutiae, using the image map and other pixel intensity statistics. The quality ranking is from 0 to 5, where 0 is poor and 5 is excellent quality.

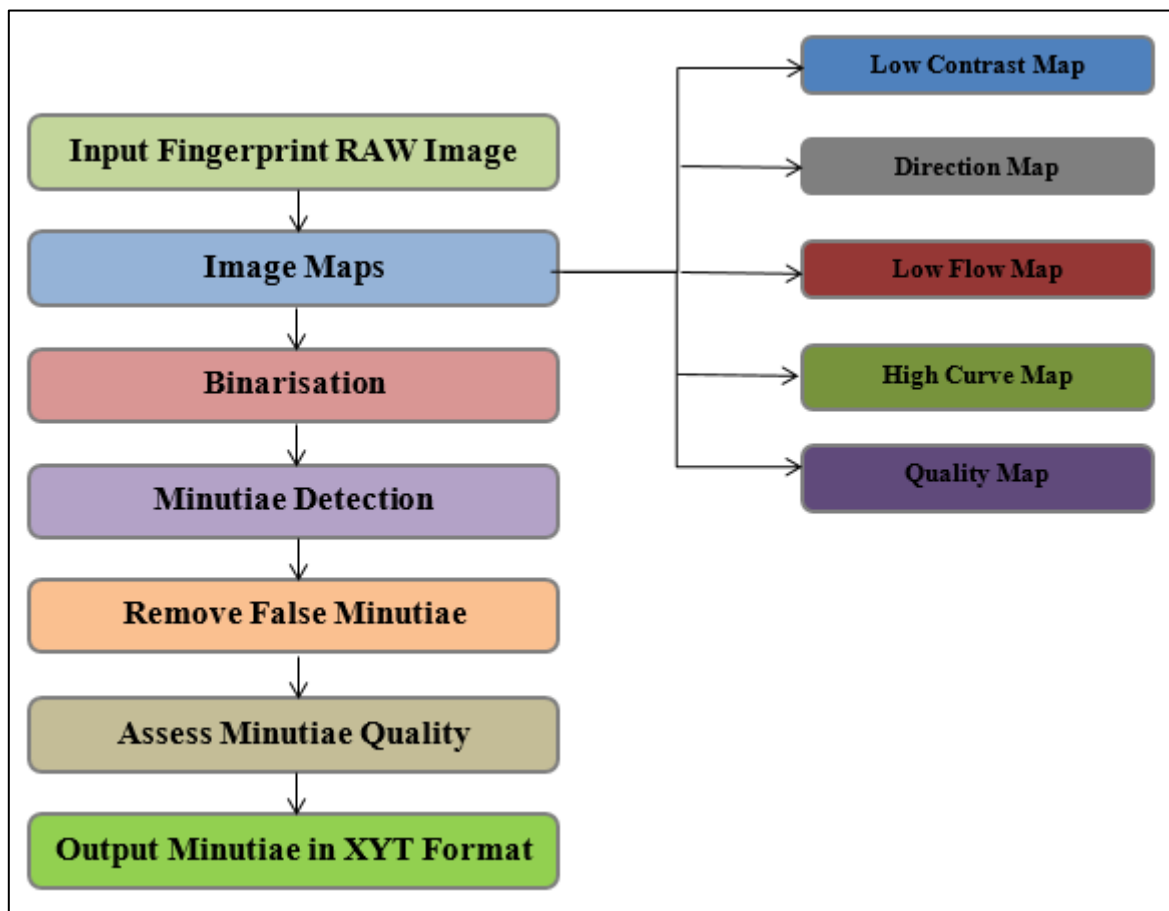


Figure 3-2: Functional steps of MINDTCT minutiae extraction algorithm adapted from [46]

The feature extractor illustrated in Figure 3-2 comes with the advantage of ridge count data for the number of ridges between two neighbouring minutiae. This information will be useful when implementing the local features-based algorithm, as it requires neighbourhood data. In addition, five nearest neighbours of a minutia are also extracted by the algorithm.

²⁰ A RAW image file contains minimally processed data from the image sensor of a digital camera, an image scanner or a motion picture film scanner.

Making use of different feature extractors for a single project may bring feature extraction errors into the study. Even if features are extracted following a single feature extraction method, there are variabilities introduced with the results, caused by coding dissimilarities. Hence, for the whole study only the Minutiae Detector (MINDTCT) feature extractor will be used.

MINDTCT was created to be able to better handle latent fingerprints. It is the result of a collaboration between the FBI and NIST. This feature extractor is customisable to fit one's requirements, and it is flexible enough to have more modules added to it. MINDTCT is also interoperable with other systems and therefore not conformant to only one system.

The minutia points are extracted and the y-coordinate is measured from the bottom of the image going upward. The x-coordinate is measured from left to right, and the orientation value is in degrees with the origin (zero degrees) pointing horizontally to the right, and it increases counter-clockwise. The feature extractor is optimized for images scanned at 19,69 pixels per millimetre (ppmm) which is equivalent to 500 pixels per inch (ppi).

3.3 Template encoding

Encoding minutiae into a template is essential for the application that will use the minutia points. Minutiae templates can be proprietary or standardised. The main aim of standardising the template encoding format is to address the minutiae interoperability issue. In order for the minutiae to be usable in multiple applications, they are standardised to a common format. There are a number of minutia representations available. The most relevant of these are CBEFF, ANSI-INCITS 378-2004 and ISO/IEC 19794-2 [26,49–51].

Common Biometric Exchange Formats Framework (CBEFF) allows for different biometric applications and devices to exchange biometric information efficiently by describing a set of necessary data elements such as a single standard biometric header, biometric specific memory block, and an optional signature block [48].

ANSI-INCITS 378-2004 describes where minutiae shall be defined, a data format for containing the data, and conformance information. This standard has no application-specific information. Moreover, this standard also specifies fingerprint minutiae format for on-card matching.

ISO/IEC 19794-2:2005 specifies the fundamental data elements that are used for the minutiae-based representation of a fingerprint. Furthermore, it defines three data formats for interchange and storage of these data. They are the record format, normal format, and compact-card format. For purposes of on-card comparison, the compact card format and record format are used. This standard was superseded by ISO/IEC 19794-2:2011, which was revised and does not have a record format for on-card comparison.

The MINDTCT feature extractor from the NIST Biometric Image Software (NBIS) version 4.2.0 extracts fingerprint minutiae that are compliant to ANSI INCITS 378-2004. This feature extractor can be found on the NIST website under service resources. This minutiae extractor is the only known no-cost system of its kind and is also used by the FBI's Universal Latent Workstation (ULW). The ULW is a tool developed by the FBI for rendering and enhancing fingerprint images, assisting operators in labelling minutiae and other fingerprint images, and formatting the information into a standard transaction file for searching federal, state and local law-enforcement fingerprint repositories.

The minutia points obtained from this extractor will be encoded to the correct ISO/IEC 19794:2-2011 formats for normal and on-card comparison. In encoding the templates to the correct ISO format, the sizes of the data fields will be used to ensure correct length. These length specifications are obtained from the ISO/IEC 19794-2:2005 and ISO/IEC 19794-2:2011 standards. The minimum value for these fields is zero and it goes up to the specified maximum as shown below in Table 3-1. The 5-byte format has an extra 2 bits that are reserved for future use and are always 00 (in binary).

Table 3-1: Minutiae field sizes

	Record format (5-byte)	Compact card format (3-byte)
Minutiae record	<= 1099511627775	<= 16777215
X- coordinate	<= 16383	<= 255
Y-coordinate	<= 16383	<= 255
Theta	<= 255	<= 63
Type	<= 3	<= 3
Reserved	= 0	Not available

3.4 Minutiae matching

The minutiae-matching process uses the extracted and formatted minutiae templates to compare the fingerprints. In order to perform minutiae matching, the matching algorithms should be implemented first. The implementation plan for the algorithms together with selected algorithms is explained in this section.

There are two main classes of fingerprint matching algorithms, with distinct fundamental characteristics. These fundamental differences (local-features matching and global-features matching) have been explored in Chapter 2. Both classes are affected by the fingerprint template that the algorithms use for the matching process. Therefore, in the comparison of algorithms, the first thing to do is to compare the algorithms' performance with the basic algorithms. The local- and global-features based algorithms are alignment and nearest-neighbourhood algorithms respectively. The matching algorithms will be implemented and then compared to each other.

The alignment algorithm implemented for this study is the one that was developed by Pan et al. [32]. This particular algorithm was chosen because it offers low memory consumption and is therefore ideal for use in smart-cards. The algorithm is based on global features. It also has the basic structure that does the rotation and translations in order to find the best transformation parameters, to give the best match between two minutiae templates. The algorithm is detailed in section 4.2 of Chapter 4.

The local features-based algorithm that has been selected for this study is the nearest-neighbourhood algorithm; it will be described in detail in section 4.1 of Chapter 4. This algorithm uses the neighbourhood structures of minutia points, to find out if there is a possible match between two minutia points [38].

Unit testing²¹ will be used to test the correctness of each function. Given a set of input, the output of a function can be worked out manually when it is known what the function does. Integration testing²² will be also performed at the end of algorithm implementation. The algorithm will also be tested with different sets of databases, to compare performances with other researchers' work.

²¹ Unit testing is testing of individual software units or groups of related units.

²² Integration testing is a phase in software testing in which individual software modules are combined and tested as a group.

The algorithms are implemented with and without the smart-card restrictions. The implementation represents the total functionality of the algorithm. It shows how good an algorithm is, without being constrained by the smart-card environment. That means the performance obtained from this implementation indicates the best representation of the algorithm.

With the smart-card constraints, the functionality of the algorithms can still be tested. Even though the algorithm functionality can still be tested, the algorithms with smart-card constraints do not perform similarly to the original implementations. In the implementation chapter 5, these selected algorithms will be implemented both with and without smart-card constraints, and the results will be compared. After the algorithms are implemented, the next section on the evaluation plan shows how the algorithms and the minutia points will be evaluated.

3.4.1 Evaluation of matchers

In the algorithm evaluation plan, two MOC algorithms will be implemented. Both are written in the Python programming language²³; one from the local features-based algorithm and one from the global features-based algorithm. Match scores between fingerprints will be collected from each of the algorithms. The match score is a value between 0 and 1, where 0 indicates a poor match and 1 indicates a perfect match.

The algorithms will go through a sensitivity test to determine which parts of the algorithm are most affected by the change of format. This is a test that will go through each function with the 5-byte and the 3-byte inputs. Through each step, the points of the algorithm where changes in results begin to occur will be identified. These are the points where improvements can be recommended, depending on the algorithms' behaviours.

3.4.2 Analysis of match scores

The following terms need to be understood before an explanation of the performance indicators that will be used:

- *Matching score* is a value between 0 and 1, where 0 is a perfect non-match and 1 is a perfect match between two fingerprints.

²³ Python is an object-oriented, high level, open-source interpreted programming language.

- *Threshold* is a value between 0 and 1, which will be used to determine whether or not fingerprint minutiae match.

Using the match score and threshold, the following error types will be used to indicate the performance of an algorithm:

- False Match Rate (FMR), which is the proportion of impostor attempts that return a match score above the allotted matching threshold. Impostor attempts are non-matching fingerprints.
- False Non-Match Rate (FNMR) is the proportion of genuine attempts that obtain a match score that is below the allotted matching threshold. Genuine attempts are attempts which are expected to match, i.e. from the same person.

The metric for measuring the accuracy of the fingerprint-matching algorithm is the EER, viz. the point where the False Match Rate and False Non-Match Rate are equal (i.e. $FMR = FNMR$ equivalent to EER). The EER value should be significantly low for an algorithm to be considered as accurate. EERs for the two different algorithms will be used as an indicator of algorithm accuracy.

3.5 Analysis of minutia points through ANOVA

ANOVA is an acronym for Analysis of Variance. It is a statistical method used to test how the average value of a numerical variable varies across a set of conditions that have all been tested within the same experiment. The numerical value being tested is called a dependent variable and the various conditions being compared in the experiment are called independent variables. The value of the dependent variable is influenced by the condition of the independent variables.

ANOVA can be used in both simple and complicated experimental designs. When only two groups of independent variables are tested, a *t*-test can be used to analyse if there is a statistically significant difference between the two group means. The *t*-test is similar to ANOVA, the only difference being that the *t*-test can only work for two groups of independent variables whereas ANOVA is like an extension of the *t*-test because it works for two or more groups. When only two groups are being tested, both ANOVA and *t*-test produce the same results [53, 54].

A statistical design that involves different levels of one factor is referred to as one-way ANOVA. When two factors are studied, the design is called 2-way ANOVA. K_i -by- K_j ANOVA is a j -way ANOVA with K_i levels of i factor and j levels of the other [52]. An example that explains the above terms is as follows: if a study is conducted to test how study time affects pass rate in a class of 20 students, the study time will be a factor; which is the independent variable, then the test score will be the dependent variable. Various levels within the study time will then be 10 hours per week, 20 hours per week, and 40 hours per week. This example is of a one-way ANOVA. In a two-way ANOVA, the example given above can be extended with another factor, e.g. class attendance which is an independent variable. So, the question would be: does class attendance have an effect on pass rate in the class of 20 students? The different levels would then be 100 % attendance, 50 % attendance and 20 % attendance. This example involving two factors is a two-way ANOVA, with two levels being attendance and study time.

Other designs are called repeated measures ANOVA; which involves levels of one or more factors from the same subject. These are also referred to as within-subject ANOVA. When each level is measured from a different group of subjects the design is called between-subjects ANOVA [52].

3.5.1 ANOVA use cases

ANOVA is used in hypothesis testing. Hypothesis testing is a statistical test which is used to determine whether there is enough evidence in a sample of data to make an inference about the truth value of a certain condition. Two opposite hypotheses are made about the population; the null (H_0) and alternate (H_a) hypotheses. H_0 states that the population means between groups are equal. The alternate hypothesis (H_a) states that at least one of the group population means is different from the other groups. ANOVA tests whether to accept or reject the null hypothesis. When the null hypothesis is accepted, the alternate hypothesis is rejected and vice versa. H_0 and H_a cannot be both accepted/rejected simultaneously. If the null hypothesis is accepted, the mean estimates are expected to be approximately equal to each other and the ratio between them approximately equal to 1. In the case that the null hypothesis is rejected, this implies acceptance of the alternate hypothesis, the ratio between the mean estimates is expected to be very different from 1.

3.5.2 Assumptions for using ANOVA

In order to use ANOVA, a number of assumptions have to be fulfilled. The distribution of samples in a level of a factor is assumed to i) follow a normal distribution, ii) have equal variances and iii) be independent from one another [53]. In chapter 13 of the Introductory Statistics book [51], additional assumptions are stated; these are obvious and may be overlooked but they make it easy to complete the ANOVA. They are: the factor is a categorical²⁴ variable and the response (sample) is in numerical values.

Normal distribution means that a sample x is normally distributed about a mean μ with variance σ^2 , i.e. $x \sim N(\mu, \sigma^2)$.

The model assumes that subjects are randomly selected and also have to be randomly assigned to exactly one treatment (level of factor) each. This means a subject cannot be assigned to the same level of a factor more than once for one study. The errors should also be independent from one another. This means that true deviations of one level would not help predict any other true deviation. If the data are not independent and samples not random, this raises possibilities of the F -ratio²⁵ (F in Table 3-3) to be strongly affected. This may lead to an incorrect decision being made about the hypotheses.

Equality of variances assumes that all populations have the same variance values. Since we are dealing with samples, they will not be exactly equal but significantly close. When variances are unequal, bias is introduced and the F -ratio will usually not be as significant as it appears to be [53]. As a rule of thumb, when testing for equal variances, no single standard deviation should be larger than twice any of the other standard deviations [50].

With regard to normal distribution and equality of variances, when the sample sizes are equal, the statistical test is relatively insensitive to both non-normality and inequality of variances [53].

3.5.3 POWER (errors introduced)

As in any test, there are errors that are introduced with the use of Analysis of Variance; they are called type I and type II errors. “Type I error” denotes the error of rejecting H_0 and

²⁴ Categorical variables are values or observations that can be sorted into groups or categories.

²⁵ F -ratio is the statistic used to test the hypothesis that the means are significantly different from one another.

accepting H_a when H_0 is actually true and should be accepted [51]. The probability of a type I error is represented by α . A “Type II error” is the error of failing to reject H_0 when H_0 is false, i.e. when H_0 should be rejected [51]. Failing to reject H_0 means that H_a will be rejected instead. The probability of a type II error is represented by β .

An experiment is said to be well designed if it has a relatively high power with respect to all alternatives which are, in a practical sense, different from H_0 [54]. Power is the probability that the decision rule rejects H_0 when a specified H_a is true, that is, the probability that the null hypothesis will be rejected when it is indeed false. The power is represented by: $Power = 1 - \beta$ [51]. The table below shows the terminology introduced by errors.

Table 3-2: ANOVA errors

Decision	State of affairs	
	H_0 true	H_0 false H_a true
Reject H_0 Accept H_a	Type I error (α)	No error ($1 - \beta$)
Fail to reject H_0 Fail to accept H_a	No error	Type II error (β)

3.5.4 How to use ANOVA

The various sources used for these steps on using ANOVA are [53]–[55,57,58]. This is not an extensive list as more sources were studied, but they all use similar steps.

When performing an ANOVA test, the aim is to get an F -ratio which can be compared to its corresponding probability in order to reach a conclusion.

Given populations with the following features:

k – Number of different groups

n_i – Total number of observations in the i^{th} group

N – Total number of observations from all groups combined

x_{ij} – j^{th} Observation in the i^{th} group where $i = 1, 2, \dots, k$ and $j = 1, 2, \dots, n_i$

$\bar{x}_i = \sum_j \frac{x_{ij}}{n_i}$ - mean of observations in i^{th} group

$\bar{x} = \sum_i \sum_j \frac{x_{ij}}{N}$ - mean of all observations

Then this information can be used to populate Table 3-3 with all necessary values.

Table 3-3: ANOVA table

	Sum of squares (SS)	Degrees of freedom (DF)	Mean square (MS)	F
Between groups	$SS_B = \sum_i n_i (\bar{x}_i - \bar{x})^2$	$n_i - 1$	$MS_B = \frac{SS_B}{n_i - 1}$	$F = \frac{MS_B}{MS_W}$
Within groups	$SS_W = \sum_i \sum_j (x_{ij} - \bar{x}_i)^2$	$N - n_i$	$MS_W = \frac{SS_W}{N - n_i}$	
Total	$SS_T = SS_B + SS_W$	$N - 1$		

The fields used in Table 3-3 are as follows:

- SS_B is the sum of squares between the groups
- SS_W is the sum of squares within groups
- MS_B is the mean square between groups
- MS_W is the mean square within groups. It is a pure estimate of variance that is unaffected by the truth value of the null or alternative hypothesis.
- F (or F -ratio) is the statistic that will be compared with the F -critical value which is obtained from the F -distribution table (attached in Appendix C)
- F -critical value depends on DF between groups and DF within groups and is represented as

$$F (DF_w, DF_b) \quad (3.1)$$

To make an inference about the null hypothesis, the steps in the table below are followed:

Table 3-4: Inference about the null hypothesis

If the F-statistic is less than $F(DF_w, DF_b)$ critical value, Accept H_0 Else Reject H_0

3.6 Bill of materials

In order to complete the study, a few materials will be required to be available. These will help in conducting experiments that yield results that can be analysed to draw up conclusions.

The specific requirements are elaborated in the following subsections.

3.6.1 Fingerprint images

The Fingerprint Verification Competition (FVC) has a number of databases that can be used for fingerprint verification. These databases have been widely used by researchers. This allows an easy repetition or verification of the study method.

MINDTCT (Minutiae Detection) is a NIST-developed minutiae detector software. It takes in a fingerprint image and produces a minutiae file. From the fingerprint image, the algorithm generates image maps, and it then binarises the image before detecting the minutiae, removing false minutiae and counting neighbour ridges. The minutiae quality is then assessed before outputting a minutiae file.

3.6.2 Minutia point data

The data are represented in the form of minutiae location (x, y), angle (θ), and *type*. *Type* can be a ridge ending or a ridge bifurcation, represented by a 1 or 2 respectively.

3.6.3 On-card fingerprint-matching algorithms

The algorithms must be minutiae-based since the study is going to analyse the difference made by minutia formats to the algorithms.

3.7 Summary

This chapter presented the research methodology that will be applied in conducting this study. Following these steps from fingerprint databases through minutiae extraction, template encoding, implementation of algorithms and evaluation will provide the information that helps answer the research questions set in Chapter 1. The next chapter will be on design of the algorithms to be implemented and design of ANOVA.

Chapter 4: Design

The previous chapter covered the research methodology, indicating the way in which a conclusion will be reached. This chapter describes the design of the algorithms in detail, together with an ANOVA design. The algorithm designs covered are the neighbourhood algorithm in section 4.1 and the alignment algorithm in section 4.2. The detailed analyses include the algorithm, the algorithm's data-flow diagram, the formulae used by the algorithm and the complexity analysis of the algorithms.

4.1 Nearest-neighbourhood algorithm

The nearest-neighbourhood makes use of local minutia structures. If two minutiae from different fingerprints have the same neighbourhood structure, it is likely that they are from the same person. The nearest-neighbourhood algorithm used in this study is presented below, as in the original paper of [38]. The minutiae matching algorithm is illustrated as in Table 4-1:

Table 4-1: Nearest-neighbourhood matching algorithm – Source [38]

```
{MINUTIAE MATCHING PROCEDURE}

- Input: * one candidate template minutia m1;
        * minutiae list of the reference template;

1 For each minutia m2 in reference template not yet matched {
2   For each neighbour n2 of minutia m2 {

    - MinDiff = upperLimit;
    - ChosenNbr = null;
3   For each not-already-matched neighbour n1 of m1 {
4     - Executes the four steps between the n1-n2
      corresponding features (directly processes next n1 if
      the bounding box rejects the controls);

      If (NeighDissimilarity < MinDiff) {
        - MinDiff = NeighDissimilarity;
        - ChosenNbr = n1;}
    }
  If (ChosenNbr != null) {
    - ChosenNbr is marked as "matched";
    - MinDissimilarity += MinDiff;
```

```

- number of matched neighbours NM= NM + 1;}
5   If (NM > N)
    - m1 and m2 are "matched": break from this For;
  }
6   If (NM < N)
    - Continue with the next minutia m2 else {
7   - m1 and m2 are "matched": +=
    (MinDissimilarity \ NM);
    - break from this For;
  }
}
- m1_m2_MatchCost = MinDissimilarity \ NM;
If (m1 and m2 are "matched") {
  - MinutiaeNMatched++;
  - Mark reference minutia m2 as "matched";
8  If (m1_m2_MatchCost < VeryOptValue)
    - STOP: the match is accepted;
  If (m1_m2_MatchCost < OptValue)
    - OptMinNumber++;
9  If (OptMinNumber == OptNumberThreshold)
    - STOP: the match is accepted;
}
- Process another minutia m1 if no stopping condition has
  occurred or if m1 and m2 are not "matched";

```

The algorithm takes as input one minutia from the candidate/input/query template and a minutia list of the reference template which is stored in the smart-card. For each neighbour of each minutia in the reference template that is not yet matched, parameters are initialised. If the candidate minutia that is being currently referenced is not matched, the algorithm goes through steps 1 to 4, as described below with the help of Figure 4-1 and Figure 4-2 which outline the relationship between two minutia points that are neighbours.

After computation of the *NeighDissimilarity*, the algorithm compares it to the *MinDiff* and if the *NeighDissimilarity* is less than the *MinDiff*, that dissimilarity value is assigned to be the smallest minimum difference. The number of matched minutiae *NM* gets updated each time a match is found between two minutiae points. The algorithm continues comparing minutiae points as long as the *NM* value is still less than *N* the total number of minutiae. A *STOP* condition occurs when the *m1_m2_MatchCost* is below an optimised threshold value, or when the optimum minimum number is the same as the threshold value.

The algorithm iterates through the candidate minugia points, comparing them with every minugia point in the reference template. The minugia-neighbour relationship required by the algorithm is presented in Figure 4-1. For each minugia point, there is at least one neighbour – this is because a valid dataset of minugia points has at least 16 minugia points. Regardless of how far apart the points are, a neighbourhood relationship exists between the minugia being referenced and the 8 minugia points closest to it. Each neighbourhood relationship involves two minugia points, where one is the minugia point under consideration and the other is one of its neighbours.

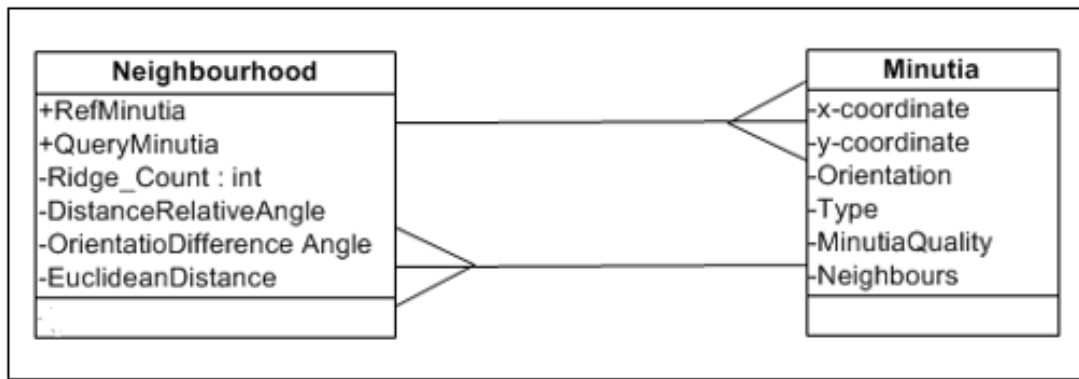


Figure 4-1: Minugia-neighbour relationship

Given two neighbours A and B, where $A = (x_1, y_1, \theta_1)$ is the reference minugia and $B = (x_2, y_2, \theta_2)$ is the neighbouring minugia to A. In order to obtain the neighbourhood features that are graphically represented in Figure 4-2, the following equations are used.

1. Euclidean distance between a reference minugia and its neighbour (Ed)

$$Ed = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (4.1)$$

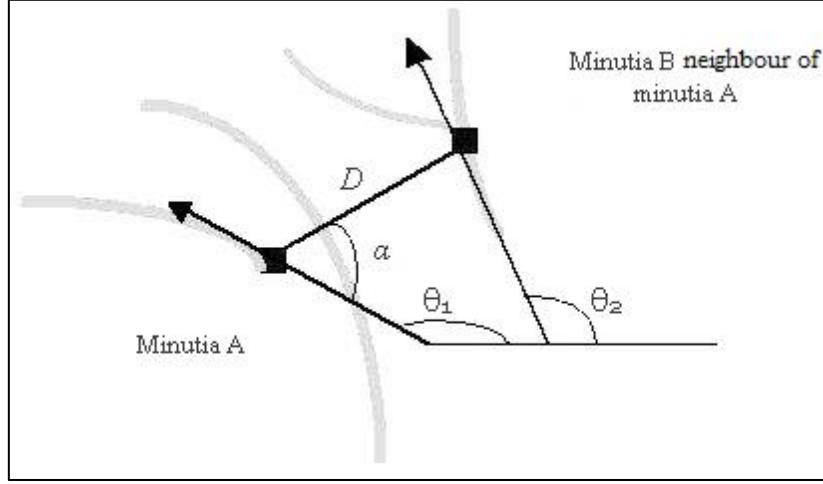


Figure 4-2: Neighbourhood features

2. Orientation difference angle (*ODA*), which is the difference angle between the reference minutia and the orientation of the neighbour ridge considered. *ODA* is calculated as in equation 4.2:

$$ODA = \theta_1 - \theta_2 \quad (4.2)$$

3. Distance relative angle (*DRA*), which is the angle between segment *D* and the reference minutia ridge direction α . To obtain α , the following steps were taken.

Creating a unit vector from the reference point: as shown in equation 4.3

$$A' = (\cos(\theta_1), \sin(\theta_1)) \quad (4.3)$$

And deriving y (represented by B') in terms of the new unit vector:

$$B' = (\cos(\theta_1), \sin(\theta_1)) \quad (4.4)$$

And a new vector

$$VN = (x_2 - x_1, y_2 - y_1) \quad (4.5)$$

Therefore

$$\alpha = \tan^{-1}\left(\frac{y' \cdot VN}{x' \cdot VN}\right) \quad (4.6)$$

4. Ridge count (RC) between the reference minutia A and the neighbouring minutia B.
The MINDTCT minutia extraction algorithm from NIST has ridge count information for the neighbouring minutia.

Using the above neighbourhood features, the dissimilarity between two minutiae from different templates is computed. The dissimilarity value will be used to infer whether or not there is a likelihood of the minutia points being from the same finger.

Let ed , rc , dra and oda be feature vectors FV_1 , FV_2 , FV_3 , and FV_4 respectively. Then at any time FV_i (where $i = 1, \dots, 4$) can be used to represent the feature vectors separately and a and b represent the query or reference minutiae respectively. Then to calculate dissimilarity values, each of the feature vectors go through the steps as in Table 4-2:

Table 4-2: Steps taken to calculate neighbourhood dissimilarity value

1. Absolute value difference between corresponding features:

$$FV_iDiff = |FVa - FVb| \quad (4.7)$$

2. Check that the two neighbours correspond in the neighbourhoods by checking if every FV_iDiff is below corresponding acceptance threshold:

$$if FV_iDiff \leq FV_iDiffThr \quad (4.8)$$

3. Multiply each FV_i diff for relative weight value:

$$FV_iWghtDiff = FV_iDiff * FV_iWght \quad (4.9)$$

$$\text{where } FV_iWght = \frac{FV_iDiff}{FV_iDiffThr} \quad (4.10)$$

4. Sum together all the four weighted differences so that a global dissimilarity value can be obtained:

$$NeighDiss = \sum_{i=1}^4 FV_iWghtDiff \quad (4.11)$$

NeighDiss is a value that represents the amount of dissimilarity between the two minutia points; it is the value that is used to decide if two minutiae are from the same finger or not.

4.1.1 Complexity analysis of the neighbourhood algorithm:

The number of instructions that the algorithm has is $f(n) = 14 + 4n + 8n^2 + 16n^3$. These instructions are counted from the algorithm's assignments, for loops and condition statements. Since the algorithm has IF conditions, there are some parts of the algorithm that will not be executed if the conditions are not satisfied. So, it is not everything in the algorithm that will execute at any given time. Given the number of instructions, the part of the algorithm that will be used most often is the 3 loops. This implies that the algorithm solves to a complexity of (n^3) . Because of the exponential nature of the algorithm it will run more slowly as the value of n increases.

4.2 Alignment algorithm

The alignment algorithm uses a multi-resolution accumulator array, thus saving memory and being suitable for smart-card use. Instead of aligning the normal way, the algorithm computes alignment parameters each time the rotation angle and direction difference are above a certain threshold. The algorithm is taken from [32] and [56] and is shown in Table 4-3.

Table 4-3: Alignment algorithm

```
Input: Minutia P
      Minutia Q
Set initial parameters of accumulator array
Set unit size
Estimate parameters
Find positions of maximum of array
Compute match score
Compare match score with threshold:
If match score > threshold
    Minutia P and Q match
Else P does not match Q
```

The algorithm in Table 4-3 can be explained in the following text and the flow diagram in Figure 4-4 further supports the algorithm and the text description:

The algorithm takes as input the minutia P template and the minutia Q template, where P is the input minutia that requires to be verified. Minutia Q is the reference template that is stored in the smart-card memory and it is used to verify if the user represented by minutia P is really the one the user claims to be.

An accumulator array is an array that is used to store votes. Similar to a statistical frequency distribution, an accumulator array records the frequency or occurrences of certain outcomes in a particular group or interval. The accumulator array is initially set to zero, and it is discretized to a finite set of values that will be counted. Not one array is created but a number of arrays are created, each for the variable that it will count. In a normal computing environment, a multidimensional array would be suitable, but in a smart-card environment, accessing a multidimensional array is computationally intensive. The arrays are initially set to zero, and are ready to collect counts/votes.

The next step estimates alignment parameters. This step is completed by recording only parameter estimates that are above a given threshold and pushing them into their respective buckets (bins) – which are called accumulator arrays – depending on the range of numbers into which the value falls.

The sets of transformation parameters are as follows:

$$\Delta x^+ \in \{\Delta x_1^+, \Delta x_2^+, \dots, \Delta x_a^+\} \quad (4.12)$$

$$\Delta y^+ \in \{\Delta y_1^+, \Delta y_2^+, \dots, \Delta y_b^+\} \quad (4.13)$$

$$\theta^+ \in \{\theta_1^+, \theta_2^+, \dots, \theta_c^+\} \quad (4.14)$$

$$s^+ \in \{s_1^+, s_2^+, \dots, s_d^+\} \quad (4.15)$$

where Δx^+ is a list of all x-coordinate transformation parameters, Δy^+ is a list of all y-coordinate transformation parameters, and θ^+ and s^+ are the associated transformation angles and scales. Now to get the best transformation parameters, the accumulator arrays are checked for the maximum number of occurrences that a certain value has, and that will be used to transform the minutia points of the input minutiae template.

To translate the minutia points from the two templates in order to estimate parameters, the following formula is used:

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - s * \begin{bmatrix} \cos\theta^+ & -\sin\theta^+ \\ \sin\theta^+ & \cos\theta^+ \end{bmatrix} \begin{bmatrix} x'_j \\ y'_j \end{bmatrix} \quad (4.16)$$

The minutia points of the query minutiae are then mapped into the reference minutiae, in order to get the best representation of the reference minutiae using the query minutiae. The mapping function is as follows:

$$\text{map}_{\Delta x, \Delta y, \theta} (m'_j = \{x'_j, y'_j, \theta'_j\}) = m''_j = \{x''_j, y''_j, \theta'_j + \theta\}, \quad (4.17)$$

where

$$\begin{bmatrix} x''_j \\ y''_j \end{bmatrix} = \begin{bmatrix} \cos\theta^+ & -\sin\theta^+ \\ \sin\theta^+ & \cos\theta^+ \end{bmatrix} \begin{bmatrix} x'_j \\ y'_j \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \quad (4.18)$$

Then after mapping the query template onto the reference template, the mapped template is compared with the reference template and the comparison result is a match score that lies between 0 and 1, where 0 means totally unmatched, and 1 means a perfect match between two minutiae has been obtained. The comparison is carried out through a process called minutiae pairing, which finds pairs between minutia points in the reference template and the newly created minutiae set from the transformation parameters. A

minutia point cannot be paired twice. The minutia pairs will be like the pairs illustrated in Figure 4-3.

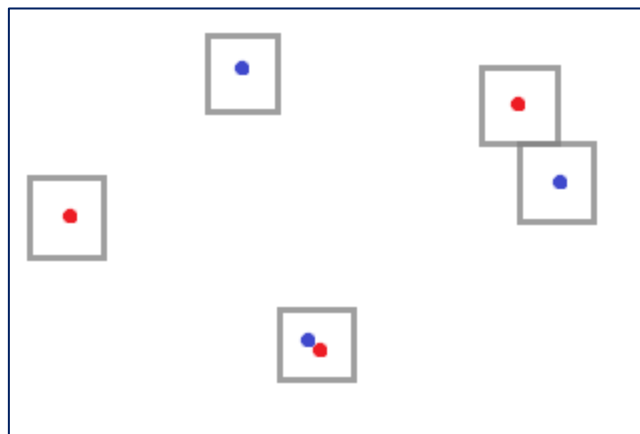


Figure 4-3: Bounding boxes for aligned minutia points

In order to obtain the match score between two minutiae, the mapped minutiae template is compared with the template that is stored in the smart-card. This is done by creating bounding boxes which will ensure that minutiae that correspond are within the bounding box and are marked to be corresponding. This does not necessarily mean that the two minutiae match but it only says that they are likely to be a match, considering their closeness which is belonging to one bounding box. The size of the bounding box can be changed, depending on the amount of elasticity²⁶ the fingerprints are allowed to have. The smaller the bounding box, the less distortion is it able to handle, and the bigger the bounding box, the better the algorithm handles distortion. However, when the box is bigger the algorithm is at risk of having false matches, as more than two minutia points may be found in a single bounding box. The above scenario is depicted in Figure 4-3. The red dot represents the input minutia, the blue dot represents the reference minutia, and the bounding box is the grey square box. When two minutiae from each separate template are found to be in the same box, they are likely to be a match. Figure 4-4 illustrates a flowchart diagram that shows all the steps of the alignment algorithm in Table 4-3 as described in this text passage:

²⁶ Elasticity of fingerprint occurs when the ridge shape is distorted due to the finger being pressed against a fingerprint scanner.

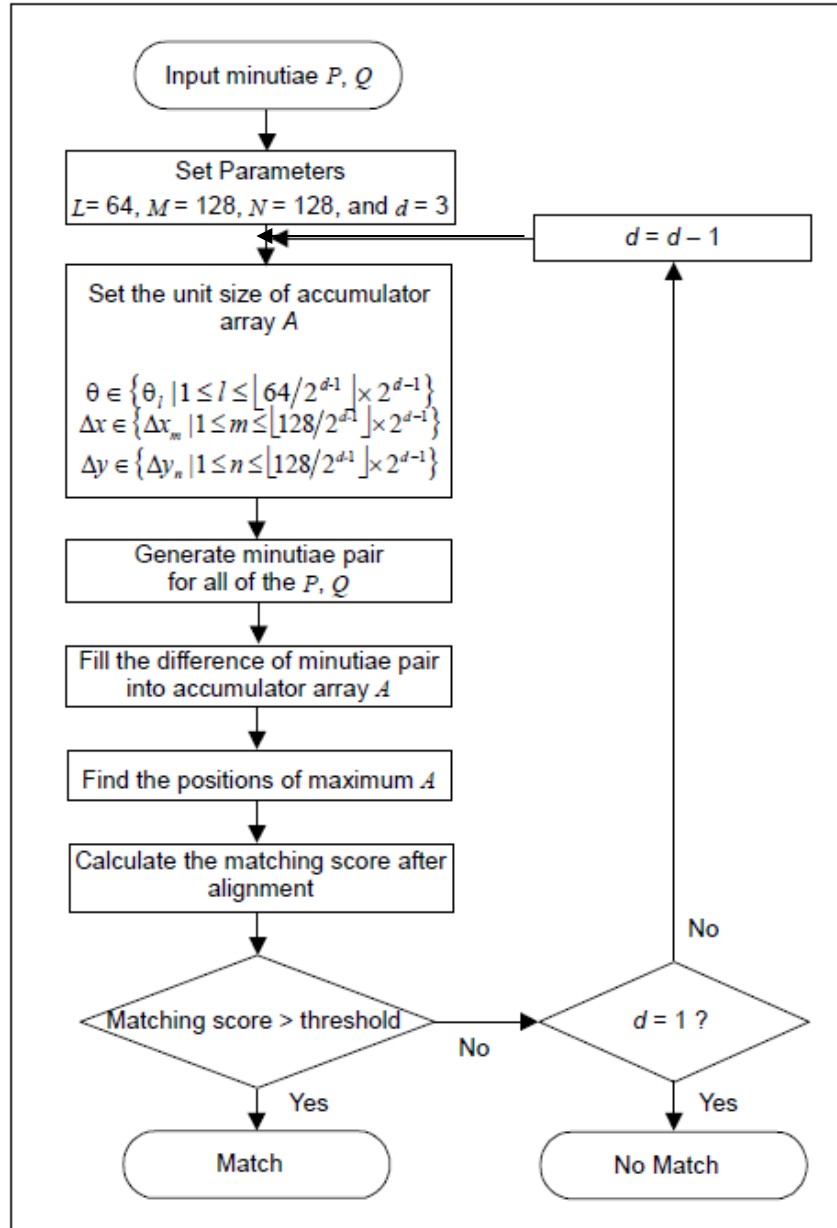


Figure 4-4: Flow chart of the alignment algorithm

4.2.1 Complexity analysis of the alignment algorithm

The alignment algorithm uses arrays to store variables. In accessing arrays, a FOR loop is efficient in doing that. So, for each time an operation is performed in the accumulator array, the complexity is $\theta(n)$. To generate minutiae pairs for all the P, Q with sizes M and N, the algorithm requires $\theta(MN)$, thus the overall complexity of the algorithm is $\theta(N) + \theta(MN) = \theta(MN)$.

4.3 ANOVA

Given a minutia file with n minutia points, where each point is represented by its (x, y) coordinates and orientation value, it is expected that when it is represented using 3-byte or 5-byte data format the minutia points still resolve to the same values when reverted back. Figure 4-5 shows minutia points in their raw form, in 3-byte representation and in 5-byte data format representation.

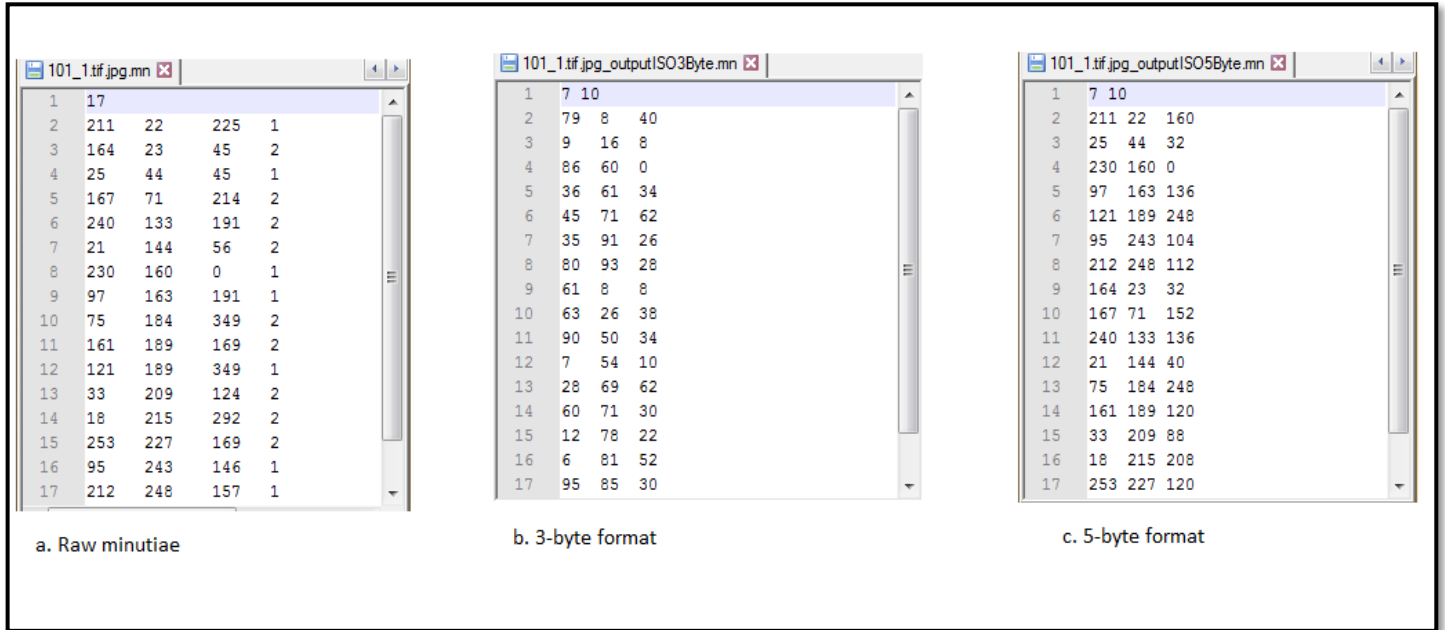


Figure 4-5: Minutia points in different formats

It should be noted that since these points should be as the standard recommends, they cannot be sent to the card in floating-point form, as floating point is not supported in smart-cards [2]. Minutia points are first rounded off to integer values before being sent to the card. The rounding-off of values is because of the limited space in the memory of the smart-card, and these values should not exceed their recommended data restrictions.

The dependent variable describes minutia points. It has three sub-categories, viz. the x and y coordinates and the orientation value. The independent variable is given in minutiae format. The minutiae format has three levels, viz. the raw/normal minutia, the minutia in 3-byte, and the minutia in 5-byte data formats.

For sampling purposes, each sub-category of the minutia points will be taken as a sample. So, for example, a minutiae file *1_3_manual.mn* has three samples which are the x and y coordinates, and the orientation value. It is these features of the minutia points that will

reveal the similarity between the minutiae along the different formats, or the lack of similarity. For each of the features in a single minutiae file, a sum will be obtained which will be used for the main samples as illustrated in Figure 4-6.

For each minutiae file, e.g. *1_3_manual.mn* with n minutia points, there will be one value which goes into the different samples obtained by: $\sum x_n$, $\sum y_n$ and $\sum \theta_n$. Figure 4-6 shows the samples obtained from different minutiae files.

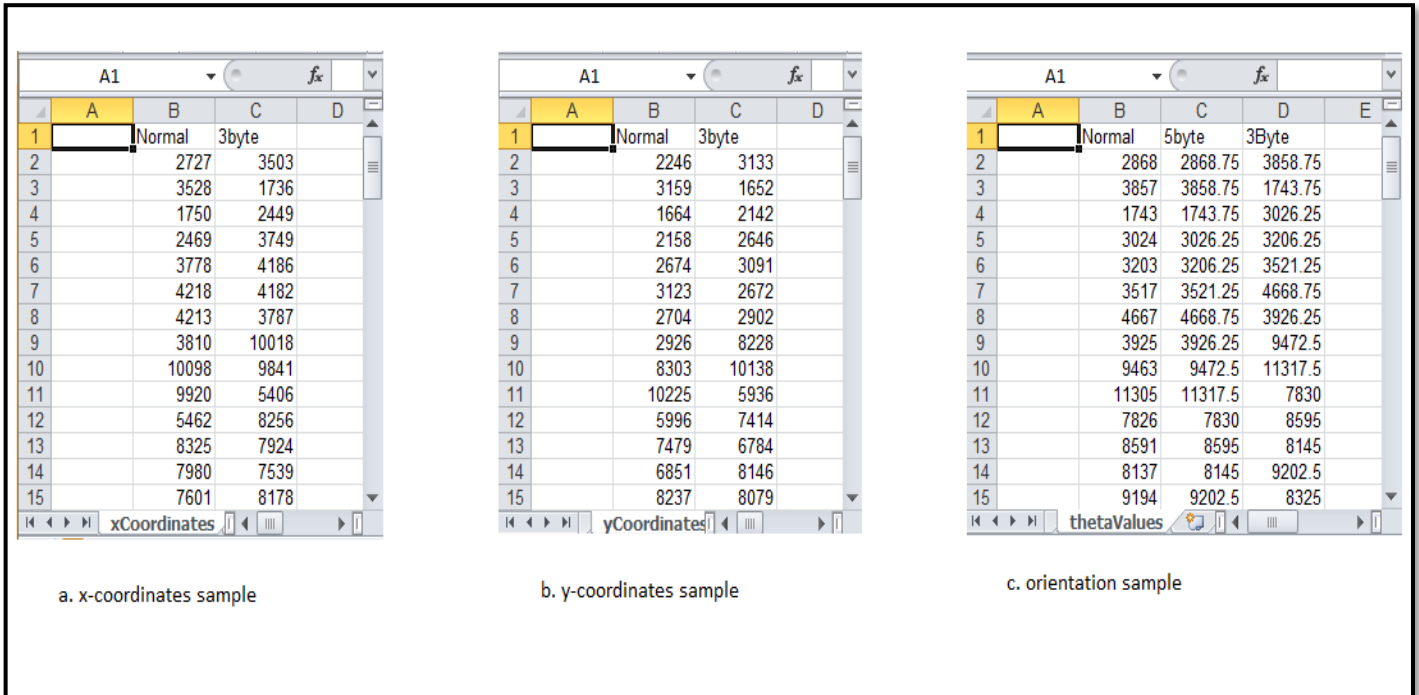


Figure 4-6: Samples of the dependent variable Minutiae (x, y, and orientation)

The ANOVA design presented above will be used in finding out whether the difference in formats causes a statistically significant difference between minutia points.

4.4 Summary

The algorithms that have been presented in this chapter are the ones that will be used to analyse and reach a conclusion on the effect of the minutia template on matching algorithm types. The alignment algorithm relies a lot on the accuracy of the resolution value as it is used to make an estimate of the transformation parameters whereas the nearest-neighbourhood relies on the existence of all minutia points, so that the local minutia structures may be similarly represented and matched. The ANOVA experiment

design is also presented. The next chapter will go through the implementation of the algorithms together with the ANOVA experimentation.

Chapter 5: Implementation

This chapter covers the implementation phase of the project. It begins with fingerprint collection and minutiae extraction from the fingerprints. Then the minutia points are converted into different minutiae formats. After this, the matching algorithms before and after adaption to a Java Card are executed with the different minutiae template formats. The ANOVA implementation is also included for the different databases, showing how the different fingerprint scanners are affected by the change in format of minutia points.

5.1 Fingerprint collection

Fingerprints from FVC2000, FVC2002 were used. In each database, four different scanners are used. Table 5-1 shows the differences between scanners for all the fingerprints. The resolution is at 500 dpi, except for the minutiae obtained from the synthetic generator, although these are also at ± 500 dpi. Each sub-database (e.g. DB1) consists of 10 subjects and 8 impressions of each subject. For each scanner there are 80 fingerprints, therefore, 320 fingerprints are available from each year's database. Image sizes vary from 240*340 pixels and 448*478 pixels, with other sizes in-between. Indicate units for image size.

Table 5-1: FVC2000 Scanners

Database	Sensor type	Image size (pixels <i>l*h</i>)	Resolution	Subject * Impressions
DB1_B	Low cost optical sensor	300*300	500 dpi	10x8
DB2_B	Low cost capacitive optical sensor	256*364	500 dpi	10x8
DB3_B	Optical sensor	448*478	500 dpi	10x8
DB4_B	Synthetic generator	240*320	About 500 dpi	10x8

In FVC2002, the image sizes vary from a width and height (W*H) of 288*384 pixels to 388*374 pixels, with other sizes in between, and are shown in Table 5-2. The resolution is 500 dpi, except for the minutiae obtained from the synthetic generator; although they are at approx. 500 dpi too. Each sub-database (e.g. DB1) consists of 10 subjects and 8 impressions of each subject. For each scanner there are 80 fingerprints, therefore FVC2002 has a total of 320 fingerprints.

Table 5-2: FVC2002 Scanners

Database	Sensor type	Image size (pixels <i>l*h</i>)	Resolution	Subject Impressions *
DB1_B	Optical sensor	388*374	500 dpi	10x8
DB2_B	Optical sensor	296*560	569 dpi	10x8
DB3_B	optical sensor	300*300	500 dpi	10x8
DB4_B	Synthetic generator	288*384	About 500 dpi	10x8

5.2 Minutiae extraction

The images obtained from FVC are in .tiff²⁷ extension. The MINDTCT tool used for extraction of fingerprint minutia points only takes ANSI/NIST²⁸, Wavelet Scalar Quantization (WSQ)²⁹, JPEG³⁰ and IHEAD fingerprint image types. IHEAD³¹ and TIFF images are lossless³², meaning they are in raster graphic format³³ and JPEG, WSQ and ANSI/NIST images are lossy compressed images.

²⁷ A TIFF- tagged image file format is an image format for exchanging raster graphics.

²⁸ ANSI/NIST - American National Standard for Information/ National Institute of Standards and Technology defined image format.

²⁹ WSQ formatted images are greyscale compressed images, converted using WSQ algorithm based on wavelet theory.

³⁰ JPEG - Joint Photographic Experts Group image format with lossy compression.

³¹ IHEAD is a raster graphics format associated with some image databases published by NIST.

³² Compression can be lossy or lossless. Lossless compression means that as the file size is compressed, the picture quality remains the same - it does not get worse. Also, the file can be decompressed to its original quality. Lossy compression permanently removes data.

³³ Raster graphics, also called bitmap graphics, a type of digital image that uses tiny rectangular pixels, or picture elements, arranged in a grid formation to represent an image.

In order to convert the TIFF images to any of the supported formats, Imagemagick was used. Imagemagick is a free software found at imagemagick.org that is distributed under the Apache 2.0 licence and it can be used for editing, composing or converting bitmap images [57]. It supports a variety of image formats and hence was helpful in converting the fingerprints images to JPEG. The following is a code snippet that was used to do the conversion.



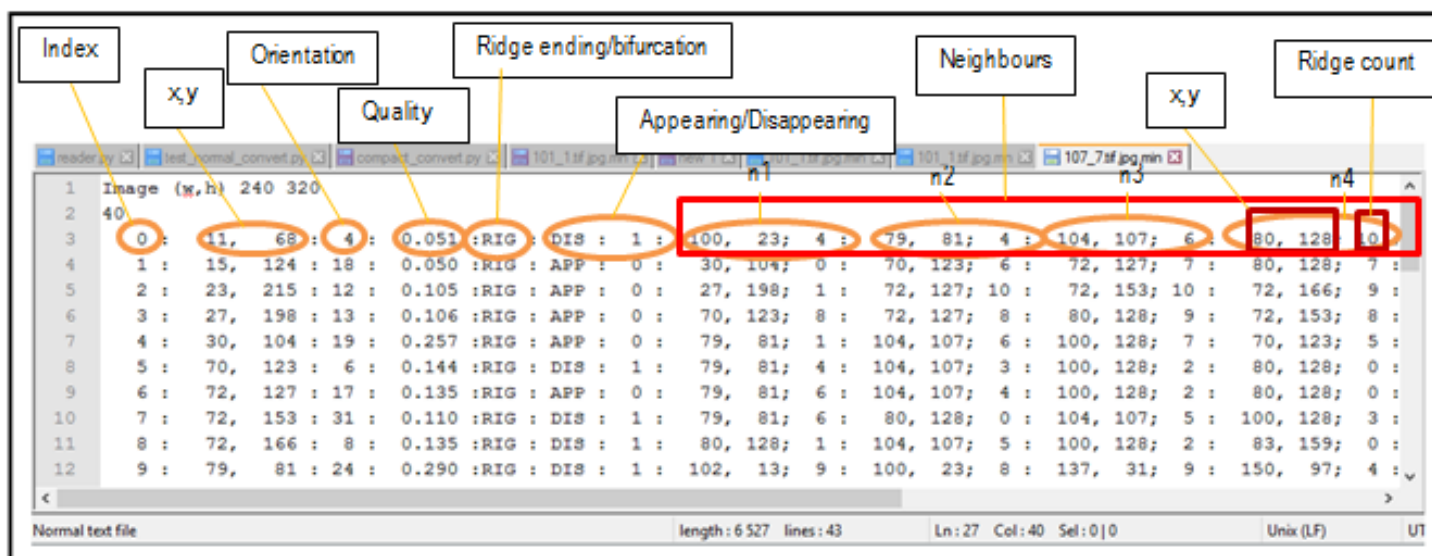
```
for f in *.tif;
> do
>   echo "converting $f";
>   convert "$f" "${basename "$f"}.jpg";
> done
```

Figure 5-1: Converting from tiff to jpg using Imagemagick

MINDTCT was edited so that it extracts fingerprints as specified by ISO/IEC 19794-2 to extract fingerprints, and also provides sufficient data to be used by the matching algorithms. The edits are as follows:

- Resolution of the orientation value in MINDTCT was incremented by 11,25°, so it was edited to make the resolution 1°. This was done to ensure that the orientation values were as close to their true values as possible.
- The neighbourhood algorithm implemented in this study makes use of the 8 nearest neighbours. The MINDTCT was edited so that it extracts the 8 nearest neighbours. It should be noted that in order to avoid redundancy, some minutia points have less than 8 neighbours. That is, towards the end of the list of minutia points, the number of neighbours begins to decrease because the resulting neighbourhood relationship might end up like one of the relationships that have been mentioned above.

The extracted minutia points appear as in Figure 5-2.



the extracted minutia points with other neighbourhood features. This snapshot involves only one minutia point which is represented by the first 3 lines and followed by the neighbourhood features for 5 nearest neighbours on the remaining lines.

```

C:\Windows\System32\cmd.exe

The minutia point
X   Y   Theta Type
170 184 349 1

The neighbours of the point above
-----
X   Y   RC   DD           ODA           DRA
175 248 7   ;64.1950153828   ;12.0;   254.974355957
195 196 2   ;27.7308492477   ;203.0;   211.667473611
216 206 4   ;50.9901951359   ;34.0;   211.609313741
227 155 1   ;63.9531078213   ;34.0;   520.766337343
184 126 4   ;59.6657355607   ;225.0;   452.823124064
entering the neighbourhood

```

Figure 5-3: Snapshot of minutia neighbourhood features

5.3 Format conversion

The pseudocode for converting the formats of the minutia points is as follows in Table 5-3. The code was developed using Python 2.7. The format conversion algorithm takes as input the minutiae points of the fingerprint that has to be converted and the format that the minutiae points have to be converted to. For every minutia point, the input is verified to check if it is within the normal values of minutiae points, i.e. below image size. Using the input format, x- and y-values together with the theta value are calculated. The calculation formula is the same for both the 3- and 5-byte formats, however, the constants (image resolution and theta resolution) are different.

Table 5-3: Format conversion pseudocode

input: list of minutia points (x_i, y_i, θ_i), format for each point x_i, y_i, θ_i verify input if format = 3_byte imageresolution = 0.1/0.26458333 thetaresolution = 5.625 xvalue = $x_i * \text{imageresolution}$ yvalue = $y_i * \text{imageresolution}$ theta = $\theta_i / \text{thetaresolution}$ if format = 5_byte thetaresolution = 1.40625 theta = $\theta_i / \text{thetaresolution}$ xvalue = x_i

```

yvalue = yi
return modified list of minutia points

```

The output of the implementation for the 3-byte and for the 5-byte data format conversion is as illustrated in Figure 5-4.

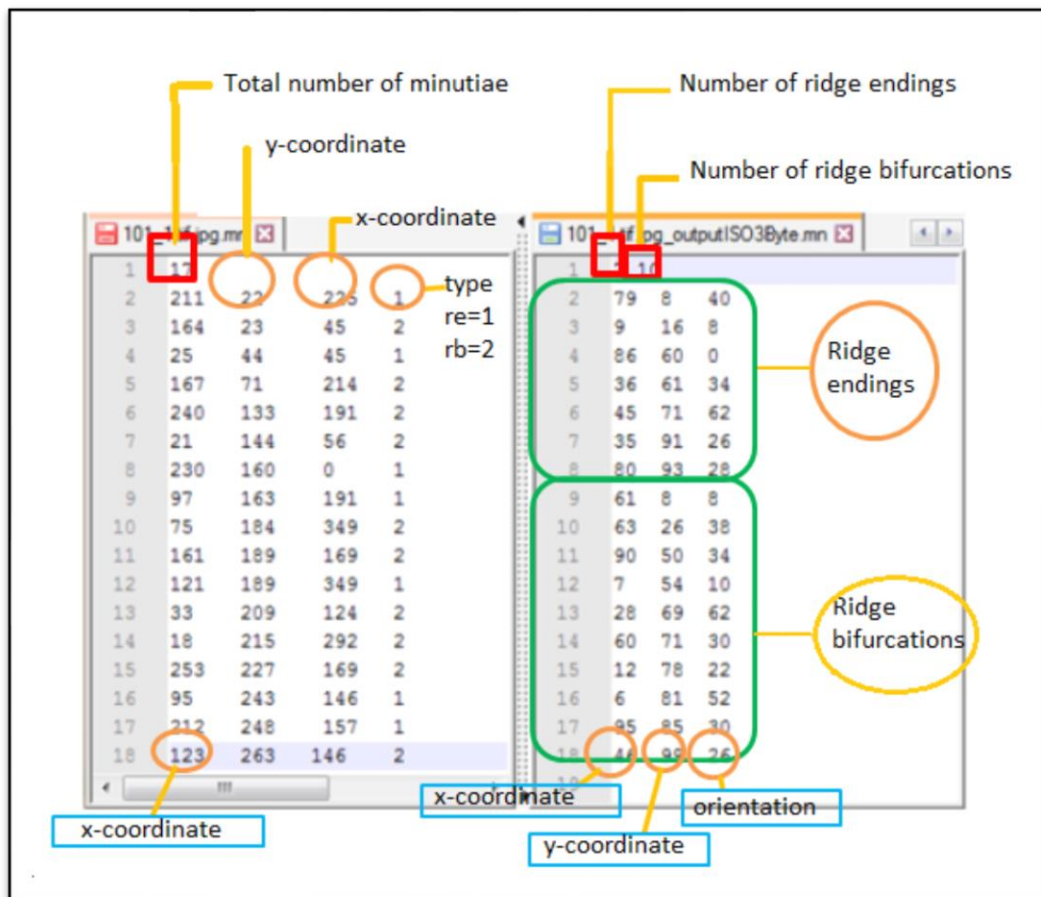


Figure 5-4: Raw minutiae to 3-byte data format conversion

The first column of output is y-coordinates, then x-coordinates, theta value and the minutiae type. The first line of output is the number of minutia points in the minutiae file. The first row on the raw minutiae has the total number of minutiae points in the file. The subsequent rows are the raw minutiae points. on the right, the first row is the number of ridge endings and the number of ridge bifurcations, of which totals to the number of minutiae points. then the subsequent rows begin with the minutiae that are part of the ridge endings followed by the ridge bifurcations. The converted minutiae as can be seen on row 3 are not the same for raw and 3-byte because of the conversion.

Figure 5-5 shows converting to 5-byte data format representation. These points are also ordered similarly to the 3-byte minutiae points format with x- and y-coordinates, the orientation value and the minutiae type. The first row on the raw minutiae has the total number of minutiae points in the file. The subsequent rows are the raw minutiae points. on the right, the first row is the number of ridge endings and the number of ridge bifurcations, of which totals to the number of minutiae points. then the subsequent rows begin with the minutiae that are part of the ridge endings followed by the ridge bifurcations. The converted minutiae as can be seen on row 3 are not the same for raw and 5-byte because of the conversion. With conversion to 5-byte, it is evident that the affected value is the theta as x- and y-coordinates still remain the same.

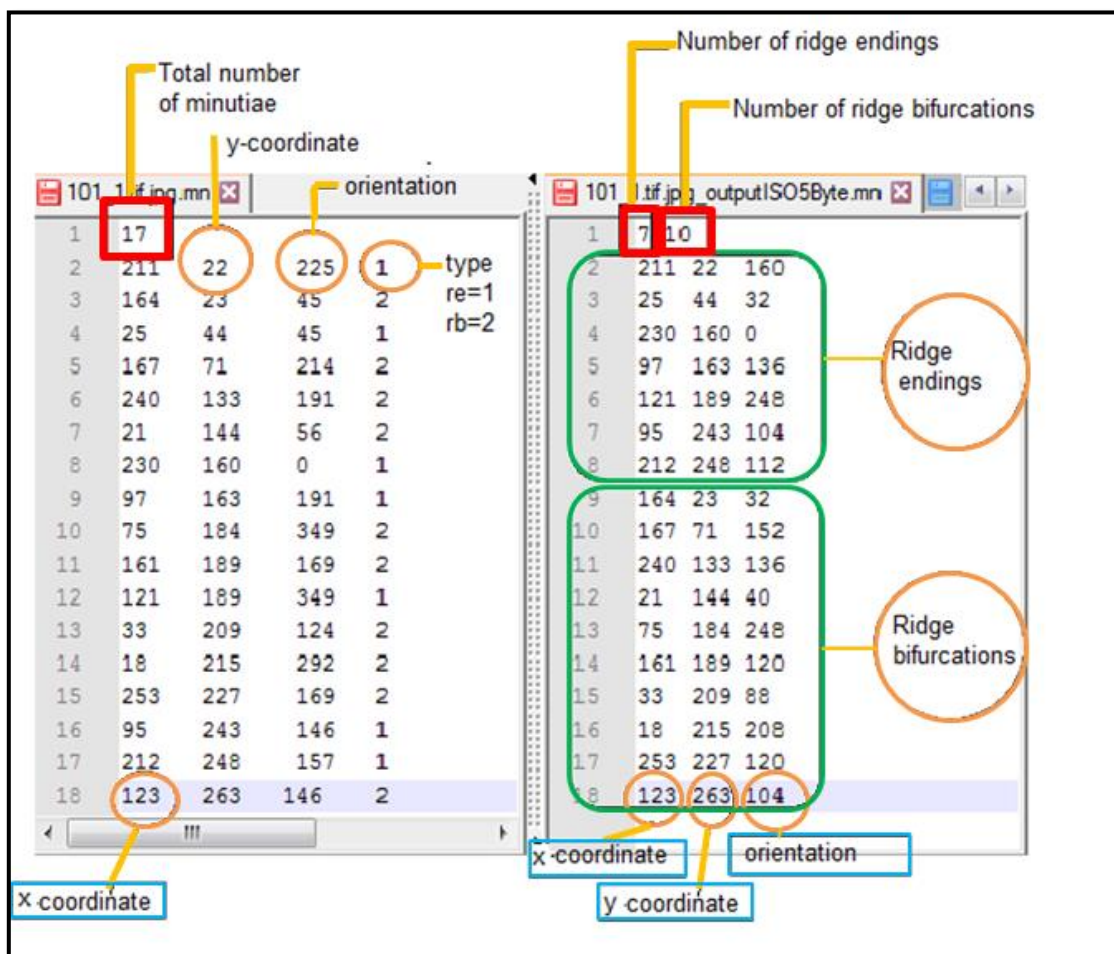


Figure 5-5: Raw minutiae-to-5-byte data format conversion

5.4 Matching algorithms with no smart-card limitations

This section is about the implementation of the matching algorithms without taking into consideration the imposed smart-card environmental limitations.

5.4.1 Nearest-neighbourhood algorithm

The design of the nearest-neighbourhood algorithm was implemented using Python 2.7 and unit tests were also implemented in Python. The code snippet for the main function that finds the dissimilarity between two points in the nearest-neighbourhood algorithm is captured in Appendix A towards the end of the thesis.

5.4.2 Alignment algorithm

The alignment algorithm was implemented in Python 2.7, following the design of the algorithm as outlined in 4.2. To ensure that the algorithm does the alignment calculations correctly (as expected), unit tests were performed on each of the main functions of the algorithm as shown. The code snippet for the alignment algorithm in Python language is captured in Appendix B towards the end of the thesis.

5.5 Matching algorithms with smart-card constraints

This section focuses on the implementation of the algorithms with the aim of bringing in the smart-card constraints. It starts with the introduction to the Python platform that is used for implementation of the code and the Java Card platform that is used for reference. Java Card is cumbersome and time-consuming to implement and test due to time limitations. Therefore, Python has been used for the sole purpose of evaluating the algorithm performance accuracy.

The Python environment that is used for implementation and testing of the algorithms does not model the real world of smart-cards. The South African smart ID card uses a Java Card operating system for the smart ID. This motivates the selection of a Java Card environment instead of the (Multi-application operating system) MULTOS or .NET Windows smart-card³⁴.

³⁴ Microsoft Windows operating system for smart card

This section maps some of the special library functionality that comes with Python 2.7 to Java Card 2.2. This is done to ensure that the algorithm can still function as it is on the card. The Java Card does not have all the functions and libraries that can be used in Python, but there are still other methods that can attempt the functionality of the Python libraries.

5.5.1 Java Card

The Java Card comes in two basic types; the memory card and the intelligent card commonly known as the smart-card. The Java smart-card has a microprocessor embedded in it, and the microprocessor enables the card to execute instructions.

The Java Card specifically has several benefits - for example, like Java, the Java Card applets³⁵ are platform independent. They are multi-application capable, flexible and compliant with the ISO/IEC 7816 smart-card standards.

The card CPU is 8-bit single tasking with 1 kB RAM. Smart-cards have three memory types, namely:

- a) Random Access Memory (RAM), which is a fast and volatile memory; all the data are wiped out when the card is disconnected from power.
- b) Read Only Memory (ROM), which is a memory whose data and memory cannot be modified. It does not lose memory when disconnected from power.
- c) Electrically Erasable Programmable Read Only Memory (EEPROM), similar to the ROM, but the data are read- and write-friendly. The Java Card provides a subset of the Java programming language and runtime environment for smart-card programming.

The Java Card resources and built-in data types are illustrated respectively in Table 5-4 and Table 5-5.

³⁵An applet is a very small application, especially a utility program, designed to perform one or a few simple functions.

Table 5-4: Java Card resources

Resource	Size
ROM	8 kB to -24 kB
EEPROM	1 kB to 24 kB
RAM	512 B to 1 kB

Table 5-5: Java Card built-in data types

Data type	Size	Range
Short	16 bits	-32 767; 32 767
Byte	8 bits	-128; 127
Bool	8 bits	True or false
Int (Only available on 32-bit smart-card)	32 bits	- 2 147 483 647; 2 147 483 647

The smart-card comes with limitations. The smart-card limitations are like those of computers in the 1980s when the CPUs were limited to 8-bit single-tasking devices with a RAM size of 1 kB or less. In the Java Card applet developer's guide these limitations have been listed [58]. Some of the limitations with regard to data type are that *long*, *char*, *float* and *double* are types that are not supported by the Java Card.

5.5.2 Python

Python is an interpreted language, which is interactive and implemented using the object-oriented method of code implementation. Code written in Python is automatically compiled by the interpreter into bytecode that is platform independent and therefore can be executed in almost any platform that has the Python Virtual Machine (PVM).

The Python software distribution used for this study is Python 2.7.0. The built-in numerical data types and sequences that are compatible with Python 2.7.0 are given in Table 5-6:

Table 5-6: Python data types

Numerical data types	Sequences
Int	String (8-bit characters)
Long	Bytes (0-255)
Float	Byte Array (mutable bytes)
Complex	List
	Tuple

5.5.3 Mapping Python with Java Card

This section maps the Python implementation of the algorithm's functionality to a Java Card.

The Python libraries that have been used in the alignment algorithm are listed as follows:

a) Fnmatch

This is a UNIX- filename pattern-matching library. For the implementation and testing of the algorithm, this library is essential in file naming of the minutiae-file templates. This is not an essential part of the matching process as the matching environment will be one-on-one, where the smart-card only receives the file contents.

b) Math

The math library provides access to mathematical functions defined by the C standard. In the math library, the modules used are as follows:

- `Math.sqrt ()`: square root

The `math.sqrt(x)` module returns the square root of a value `x`. The Java Card Application programming interface (API) has no such library; therefore, a lookup table has to be used to implement such a function. Another way this can be done is

through the crypto API by performing an RSA³⁶ modulo exponentiation using 2 as public key and a large modulus to perform squares. There is the Newton-Raphson method, which is a technique that uses a linear approximation which can also be used, as it solves for equations numerically [62,63].

- `Math.cos(x)` and `Math.sin(x)`: cosine and sine

These two modules perform trigonometric functions and they return the sine and cosine of x . Since Python `math.sin(x)` and `math.cos(x)` modules take x as a radian value in Python, x first gets transformed to radians. Java Card does not have a module for these operations. To do the `sin(x)` operation, a polynomial of degree 13 can be used to approximate the value of this operation. Also for the `cos(x)` operation a polynomial can be used. This polynomial algorithm can be a Taylor series implemented using the COordinate Rotation Digital Computer (CORDIC) algorithm [61]. This is a simple and efficient algorithm that calculates hyperbolic and trigonometric functions by rotating the phase of a complex number by means of multiplying it by a succession of constant values.

- `Math.radians(x)`: radians

This module returns the radian of a value x . In order to do this on the Java Card, a constant can be used. Radians of an angle = angle in degrees $\times \frac{\pi}{180}$. This value, if calculated with the Java Card short-number data type, yields a non-floating-point result. This can cause loss of precision. In order to increase some precision a byte array can be used to contain the result of this operation.

- `Math.float(x)`

This module returns the floating-point representation of a value. If provided with a quotient of two numbers in any of the accepted number types, it will return the result of the quotient in floating point. The Java Card does not support this type, therefore to represent floating point numbers, a byte array takes over.

³⁶ RSA is an algorithm that is used to encrypt and decrypt messages by modern computers. It is an asymmetric cryptographic algorithm with two different keys.

c) Numpy library

This library adds support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays. The module used from this library is:

- `Numpy.dot()`

This module returns the dot product of two arrays. This dot-product module takes an input of two arrays (vectors). It can also perform the dot product of scalar values, which is similar to multiplication.

d) Python Image Library (PIL) library

This library adds image-processing capabilities to the Python interpreter. The `Imagedraw()` module was used for visualisation of the minutia points on the fingerprint image. This module will not be used in the smart-card as it is non-essential for the matching procedure.

e) Sys library

This library provides system-specific parameters and functions. The module used from this library is:

- `Sys.argv()`

The module holds the list of command-line arguments passed to a Python script. This was used for the purpose of testing for multiple minutiae files. The Java Card Application Protocol Data Unit (APDU) corresponds to this module as it is the communication unit between a smart-card reader and a smart-card. In Java Card, it is used to pass arguments, minutiae file, and commands between the smart-card and smart-card reader.

f) Multidimensional arrays

The Java Card does not accept multidimensional arrays. Only one-dimensional arrays are accepted by a Java Card. Therefore, all arrays will be implemented in the form of one dimension of $(1*n)$ objects.

With the Java Card restrictions as stated and the ways in which they will be mitigated, the Java Card will have a functionality that can approach that of the Python implementation.

The Python libraries used in the nearest-neighbourhood algorithm are as follows:

The nearest-neighbourhood algorithm uses the same libraries as the alignment algorithm. In addition to modules in the Math library, the inverse tangent is also used.

a) Math.lib

- Math.atan2(x): This module performs a trigonometric function and it returns the arctan of x . Since Python math.atan(x) modules take x as a radian value in Python, x first gets transformed to radians. The Java Card does not have a module for these operations. Similarly to the sine and cosine modules, a polynomial of degree 13 can be used to approximate the value of this operation.

5.6 ANOVA

An ANOVA implementation was done for each of the main databases, then each of the sub-databases, then for all the databases combined. The ANOVA design that was described in section 4.3 was used for all the implementations in the sub-sections below. The procedure is the same for the implementations, however the input variables change and the input variable will be shown in each implementation in sub-sections below.

5.6.1 Y-coordinate

Below are the ANOVA implementations for the y-coordinates in FVC2000 and FVC2002.

Table 5-7 : Single-factor ANOVA for Y-coordinates in FVC2000

SUMMARY				
Groups	Count	Sum	Average	Variance
Normal	320	4187391	13085.6	1.53E+08
3byte	320	4159409	12998.15	1.52E+08

ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	1223426	1	1223426	0.008029	0.92863	3.856075
Within Groups	9.72E+10	638	1.52E+08			
Total	9.72E+10	639				

Table 5-8: Single- factor ANOVA for Y-coordinates in FVC2002

SUMMARY				
Groups	Count	Sum	Average	Variance
Normal	320	2704127	8450.397	16482516
3byte	320	2685532	8392.288	16380385

ANOVA						
Source of Variation	SS	df	MS	F	P-value	Fcrit
Between Groups	540271.9	1	540271.9	0.03288	0.856167	3.856075
Within Groups	1.05E+10	638	16431451			
Total	1.05E+10	639				

5.6.2 X-coordinate

Below are the ANOVA implementations for the x-coordinates in FVC2000 and FVC2002.

Table 5-9: Single-factor ANOVA for X-coordinates in FVC2000

SUMMARY				
Groups	Count	Sum	Average	Variance
Normal	320	3630101	11344.07	1.18E+08
3byte	320	3602268	11257.09	1.16E+08

ANOVA						
Source of Variation	SS	df	MS	F	P-value	Fcrit
Between Groups	1210431	1	1210431	0.010356	0.918976	3.856075
Within Groups	7.46E+10	638	1.17E+08			
Total	7.46E+10	639				

Table 5-10: Single factor ANOVA for X-coordinates in FVC2002

SUMMARY						
Groups	Count	Sum	Average	Variance		
Normal	320	2281491	7129.659	5967947		
3byte	320	2263004	7071.888	5904465		

ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	534014.3	1	534014.3	0.089959	0.764327	3.856075
Within Groups	3.79E+09	638	5936206			
Total	3.79E+09	639				

5.6.3 Orientation

Below are the ANOVA implementations for the orientations in FVC2000 and FVC2002.

Table 5-11: Single factor ANOVA for orientation on the FVC2000 database

SUMMARY				
Groups	Count	Sum	Average	Variance
Normal	320	3850707	12033.46	76533542
5byte	320	3853283	12041.51	76627379
3Byte	320	3853283	12041.51	76627379

ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	13819.17	2	6909.584	9.02E-05	0.99991	3.00513
Within Groups	7.33E+10	957	76596100			
Total	7.33E+10	959				

Table 5-12: Single factor ANOVA for orientation on the FVC2002 database

SUMMARY				
Groups	Count	Sum	Average	Variance
Normal	320	2581805	8068.141	6499675
5byte	320	2583596	8073.738	6506995
3Byte	320	2583596	8073.738	6506995

ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	6684.534	2	3342.267	0.000514	0.999486	3.00513
Within Groups	6.22E+09	957	6504555			
Total	6.22E+09	959				

The ANOVA results presented here will be analysed in the next chapter.

5.7 Validation

The algorithms that have been implemented were validated to ascertain if they are functioning as they should. Unit tests were used to test the functions to ensure that algorithms correctly perform computations. This section presents the validation process with test cases and screen shots of the execution of the tests.

5.7.1 Format converters

The format converters were validated in order to ensure that the templates were correctly encoded as defined by the ISO/IEC 19794:2 Standard. The test cases implemented for validating 3-byte and for 5-byte are captured in Figure 5-6 and Figure 5-7 respectively. The html files were created using a Python library called *pydoc* that helps with producing documentation for code written in Python. The tests were written with the Python library called *unittests*.

[index](#)

[test_compact_](#)

[d:\onedrive\codingprojects\test_compact_.py](#)

Modules

[record_convert](#)
[unittest](#)

Classes

[unittest.case.TestCase](#)([_builtin_.object](#))
 [CompactTest](#)

class **CompactTest**([unittest.case.TestCase](#))
 Method resolution order:
 [CompactTest](#)
[unittest.case.TestCase](#)
[_builtin_.object](#)

 Methods defined here:
 toCompactTemplate(self)
 Tests the fuction that converts input into compact card values
 :param: x = 102
 y = 13
 theta = 270
 type = 1
 :output: x = 38
 y = 4
 theta = 48
 type = 1

Figure 5-6: Compact card converter unit tests

[index](#)
[d:\onedrive\codingprojects\test_record.py](#)

test_record

Modules

[record_convert](#)
[sys](#)
[unittest](#)

Classes

[unittest.case.TestCase\(\[_builtin_.object\]\(#\)\)](#)
[RecordTest](#)

```

class RecordTest(unittest.case.TestCase)
    Method resolution order:
        RecordTest
        unittest.case.TestCase
        \_builtin\_.object

    Methods defined here:

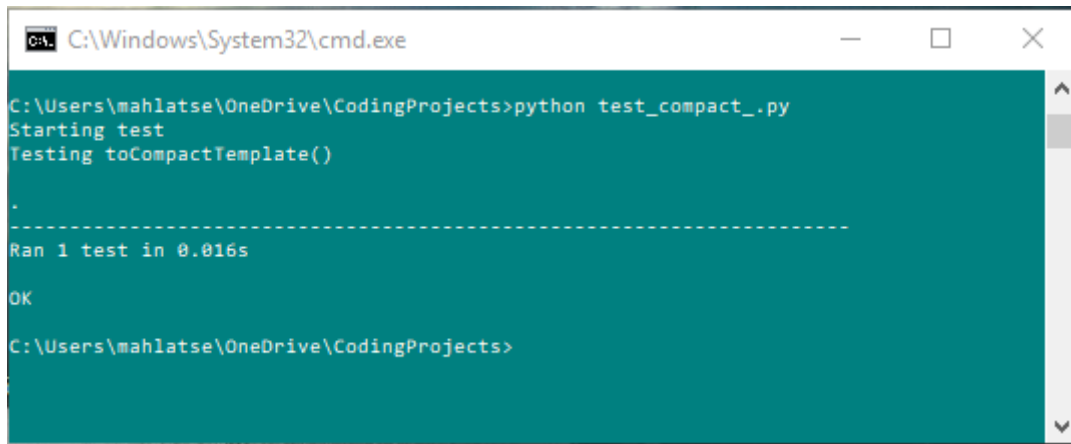
    test_toRecord(self)
        Tests the fucntion that converts input into record format values
        :param: x      = 102
                   y      = 13
                   theta = 270
                   type  = 1

        :output: x      = 102
                   y      = 13
                   theta = 192
                   type  = 1

```

Figure 5-7: Record format converter unit tests

Upon completion of the test cases, the unit tests were executed and the results are as follows in Figure 5-8 and Figure 5-9 for the compact card and record format respectively.



```
C:\Windows\System32\cmd.exe

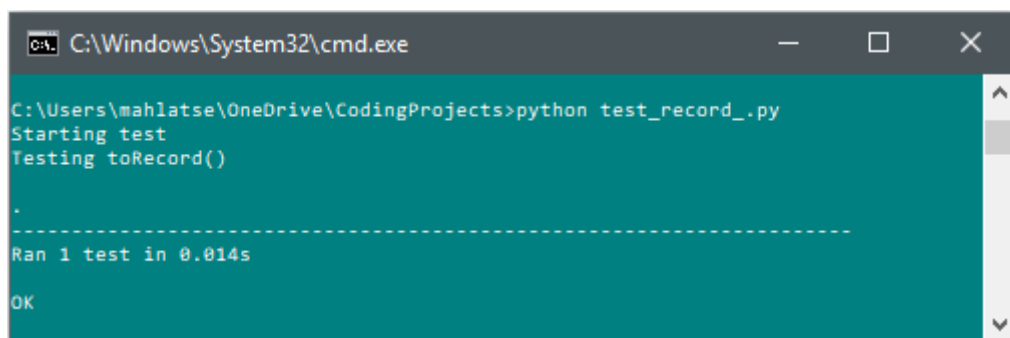
C:\Users\mahlatse\OneDrive\CodingProjects>python test_compact_.py
Starting test
Testing toCompactTemplate()

-----
Ran 1 test in 0.016s

OK

C:\Users\mahlatse\OneDrive\CodingProjects>
```

Figure 5-8: Execution of compact-card converter unit tests



```
C:\Windows\System32\cmd.exe

C:\Users\mahlatse\OneDrive\CodingProjects>python test_record_.py
Starting test
Testing toRecord()

-----
Ran 1 test in 0.014s

OK

C:\Users\mahlatse\OneDrive\CodingProjects>
```

Figure 5-9: Execution of record converter unit tests

Successful completion of execution of the code implies that the test cases were fulfilled and therefore ascertains that the core functions of the algorithm are performing correctly.

5.7.2 Matching algorithms

The test cases for matching algorithms are presented below, along with the execution screenshots. The Python library *pydoc* was used to create the html files and *unittests* library was used to create the unit tests.

The test cases for the alignment algorithm and nearest-neighbourhood algorithm are as shown in Figure 5-10 and Figure 5-11 respectively.

Modules

[align_algorithm](#)[sys](#)[unittest](#)

Classes

[unittest.case.TestCase](#)([builtin .object](#))[AlignmentTest](#)class **AlignmentTest**([unittest.case.TestCase](#))

Method resolution order:

[AlignmentTest](#)
[unittest.case.TestCase](#)
[builtin .object](#)

Methods defined here:

test(self)**test_DirectionDiff**(self)Tests function for finding direction difference between two minutiae points
Input: theta1 = 0
 theta2 = 0

Output: chTheta = 0

test_bucketing(self)Tests the placing of minutia value in its rightful bin/bucket, used by findingBuckets()
:input: value = 34
 bucketLength 120

:return: 40**test_map**(self)**test_mm**(self)Tests finding of minutiae pairs between reference template and mapped minutiae points
: input: query template and reference template

: return: Average number of paired minutiae points from (mi and mj)

: this function has been tested manually and it passed**test_spatial_diff**(self)Tests the spatialDiff function for finding the spatial difference between two minutiae points
Input: x1 = 100 x2 = 100
 y1 = 100 y2 = 100

Output: sd = 0

test_transParam(self)Tests function for calculating the transformation parameters
input: x1 = 100 Expected output: chX = 0 y1 = 100
 x2 = 100
 y2 = 100
 chTheta = 0

chY = 0

Figure 5-10: Test cases for alignment algorithm

test_nearest_neigh

[index](#)[c:\users\mahlatse\onedrive\codingprojects\test_nearest_neigh.py](#)

```
# import nearestneighbourhood
```

Modules

[unittest](#)

Classes

[unittest.case.TestCase](#)([builtin .object](#))[NearestNeighbourhoodTest](#)**class NearestNeighbourhoodTest**([unittest.case.TestCase](#))

Method resolution order:

[NearestNeighbourhoodTest](#)[unittest.case.TestCase](#)[builtin .object](#)

Methods defined here:

test(self)**testDistanceRelativeAngle(self)**

Test the function that calculates the distance relative angle

: input: (x1,y1,t1) = (100,100,0)
(x2,y2,t2) = (200,200,0)

: output: 0

testEuclideanDistance(self)

Test the function that calculates the euclidean distance between two spatial points

: input: (x1,y1)=(5,6)
(x2,y2)=(-7,11)

: output:13

testFeatureVectorWeight(self)

Test the function that

: input: (featureVector, featureVectorThreshold) = (15,5)

: output: 3

testNeighbourhoodDissimilarity(self)

Test the function that

: input: (edDiff, rcDiff, draDiff, odDiff) = (10,5,10,5)

: output: 27.083333333333336

testOrientationDistanceAngle(self)

Test the function that calculates the orientation distance angle between two spatial points

: input: (x1,y1,t1) = (100,100,0)
(x2,y2,t2) = (100,100,0)

: output:0

testRelativeWeightValue(self)

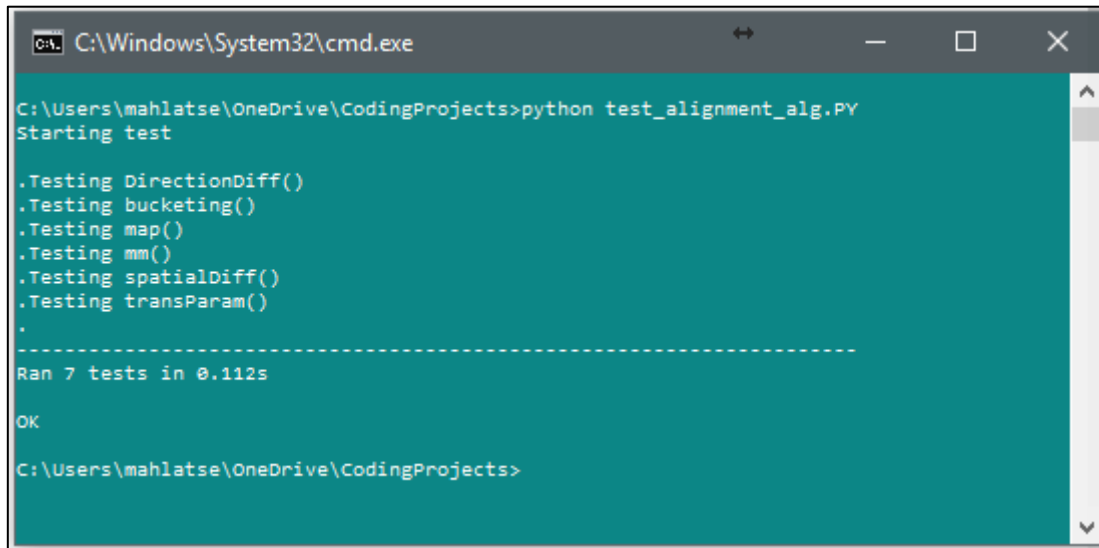
Test the function that calculates the relative weight value

: input: (edDiff, rcDiff, draDiff, odDiff) = (10,5,10,5)

: output: 27.083333333333336

Figure 5-11: Test cases for the nearest-neighbourhood algorithm

Successful execution of the Unit Test classes for the alignment and nearest-neighbourhood algorithms ascertains that the core functions of the algorithm are performing correctly. Execution of the test cases can be seen in Figure 5-12 and Figure 5-13 respectively.



```
C:\Windows\System32\cmd.exe

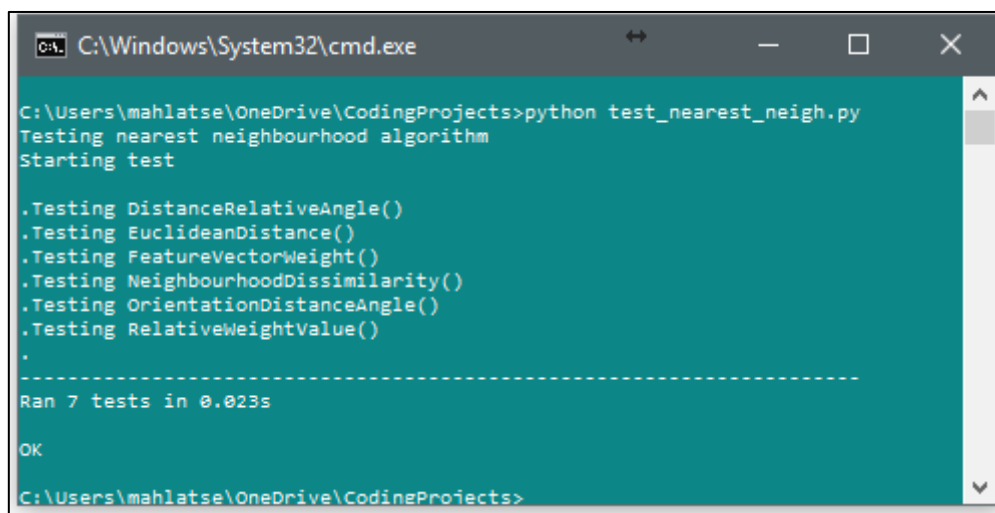
C:\Users\mahlatse\OneDrive\CodingProjects>python test_alignment_alg.PY
Starting test

.Testing DirectionDiff()
.Testing bucketing()
.Testing map()
.Testing mm()
.Testing spatialDiff()
.Testing transParam()
.
-----
Ran 7 tests in 0.112s

OK

C:\Users\mahlatse\OneDrive\CodingProjects>
```

Figure 5-12: Execution of the alignment algorithm test



```
C:\Windows\System32\cmd.exe

C:\Users\mahlatse\OneDrive\CodingProjects>python test_nearest_neigh.py
Testing nearest neighbourhood algorithm
Starting test

.Testing DistanceRelativeAngle()
.Testing EuclideanDistance()
.Testing FeatureVectorWeight()
.Testing NeighbourhoodDissimilarity()
.Testing OrientationDistanceAngle()
.Testing RelativeWeightValue()
.
-----
Ran 7 tests in 0.023s

OK

C:\Users\mahlatse\OneDrive\CodingProjects>
```

Figure 5-13: Execution of the nearest-neighbourhood algorithm test

5.8 Summary

This chapter presented the matching algorithms and also the ANOVA implementations. The algorithms were first implemented “as is” and then the smart-card environmental restrictions were taken into consideration and also implemented for analysis.

Chapter 6: Results, analysis, and validation

This chapter shows the results obtained from the implementations done in the previous chapter, analyses those results and validates them to come to a conclusion. The first section focuses on the matching algorithms and then the second section focuses on the minutia points.

6.1 Algorithms

The results for the nearest-neighbourhood algorithm are presented with the use of probability density function (PDF) graphs, and the same is done for the alignment algorithm. In each result section, there is a subsection with the 5-byte format first and then the 3-byte format.

6.1.1 Nearest-neighbourhood algorithm

The nearest-neighbourhood algorithm implementation results are presented in this section.

6.1.1.1 *Record format*

The nearest-neighbourhood algorithm results using the 5-byte data produced the results in Figure 6-1. The execution of the algorithm produced an EER of 0,18 %. The genuine and impostor graphs show the differences between the two different attempts which are clearly separated with very little overlap. The FMR100, FMR1000 and FMR10000 specify the values of FNMR for FMR=1/100, FMR=1/1000 and FMR=1/10000. All of them are 0 % at the specified EER. The lowest FNMR for FMR=0 % is zeroFNMR=2,22 % and the lowest FMR for FNMR=0 % is zeroFMR=0,36 %. The number of false positives at the EER is 0 and the number of false negatives is 1 at the specified EER. This means that 1 impostor may be accepted but no genuine attempts stand a chance of being rejected.

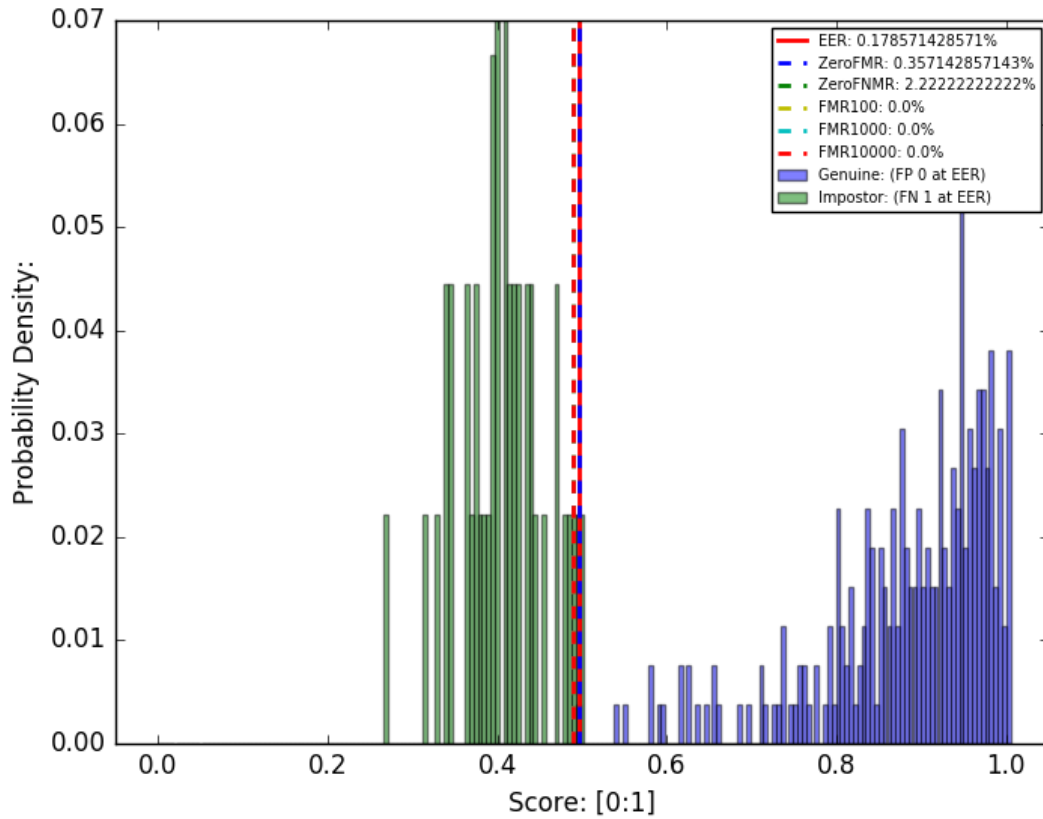


Figure 6-1: PDF for nearest neighbourhood algorithm with 5-byte data

6.1.1.2 Compact-card format

The results from the nearest-neighbourhood algorithm with the 3-byte data format show an EER of 0,54 %. This is the case when using the 3-byte data. There is a clear distinction between the genuine and the impostor attempts and this can be seen in Figure 6-2. However, with the increased EER, the algorithm has more false negatives than it had when the 5-byte data were used. This means that two attempts will be regarded as false negatives when they were supposed to be positive at the specified EER. The values of FNMR for FMR=1/100, FMR=1/1000 and FMR=1/10000. All of them are 0,72 % at the specified EER. The lowest FNMR for FMR=0 % is zeroFNMR=8,89 % and the lowest FMR for FNMR=0 % is zeroFMR=1,07 %.

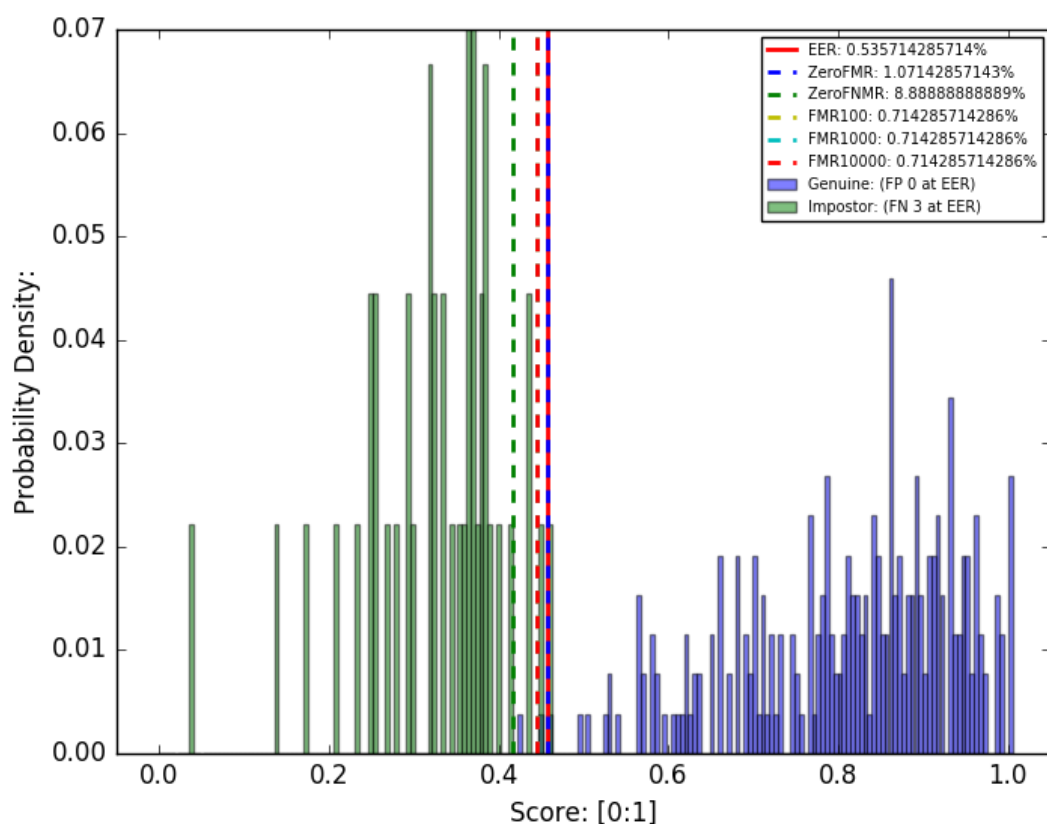


Figure 6-2: PDF for nearest neighbourhood algorithm with 3-byte data

6.1.2 Alignment algorithm

The alignment algorithm implementation results are presented in this section.

6.1.2.1 Record format

The alignment algorithm with the record format data produced the following results. The EER produced by this algorithm is 4,90 % with two genuine attempts being dubbed as impostors and 15 false negatives at the specified EER value. The FMR100, FMR1000 and FMR10000 specify the values of FNMR for FMR=1/100, FMR=1/1000 and FMR=1/10000. All of them are 5,36 % at the specified EER. The lowest FNMR for FMR = 0 % is zeroFNMR = 100 % and the lowest FMR for FNMR = 0 % is zeroFMR = 7,5 %. The impostor and genuine attempts are properly separated, although there is an overlap between the two attempts. There are still more genuine attempts in the lower part of the EER, which means that there will still be falsely rejected genuine attempts. The impostor

attempts that overlap into the genuine side are more than the genuine attempts that overlap into the impostor side of the EER, which is a red flag; there are more impostors that will be falsely accepted than the falsely rejected genuine attempts. Figure 6-3 presents these results.

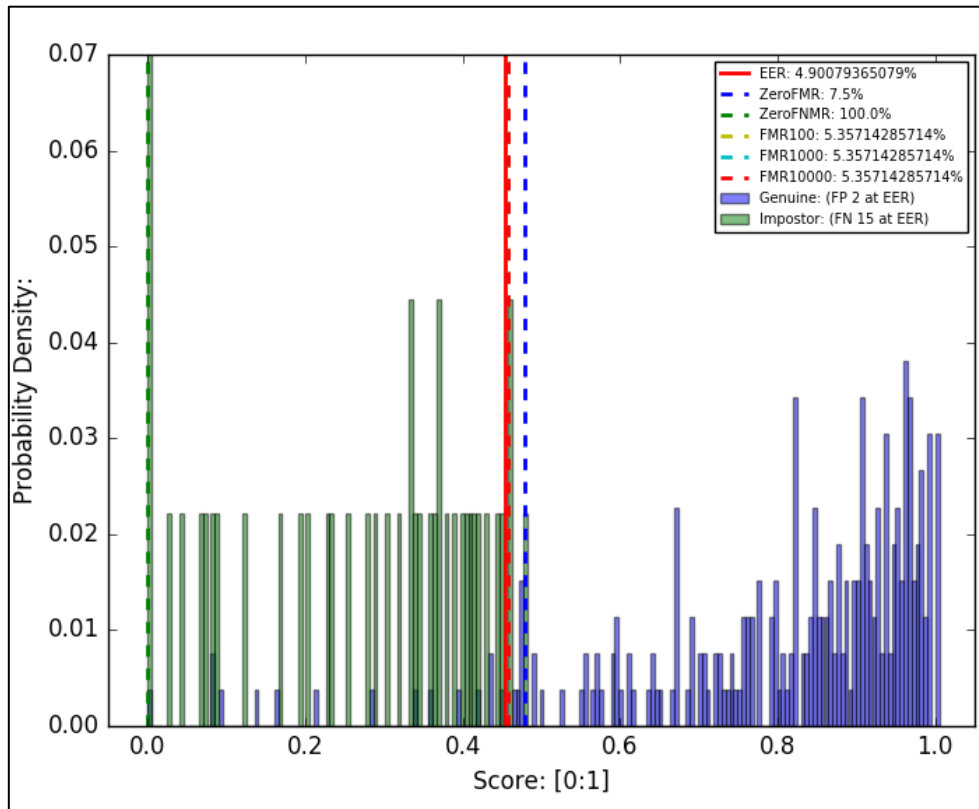


Figure 6-3: PDF for alignment algorithm with 5-byte data

6.1.2.2 Compact card format

The algorithm performance with 3-byte data is as follows. The algorithm produced an EER of 2,00 %, which at first glance may seem as if the algorithm performs better with the 3-byte data than it does with the 5-byte data. But, taking a closer look at the probability density graph in Figure 6-4, it is clear that there is no proper distinction between genuine and impostor attempts. The values of FNMR for FMR = 1/100, FMR = 1/1000 and FMR = 1/10000 are all 1,78 % at the specified EER. The lowest FNMR for FMR = 0 % is zeroFNMR = 82,22 % and the lowest FMR for FNMR = 0 % is zeroFMR = 1,78 %. The performance of this algorithm with 3-byte data is bad, and this cannot be used for a matching system as it will be difficult to even select a matching threshold that would aid in the acceptance and rejection of false and genuine attempts.

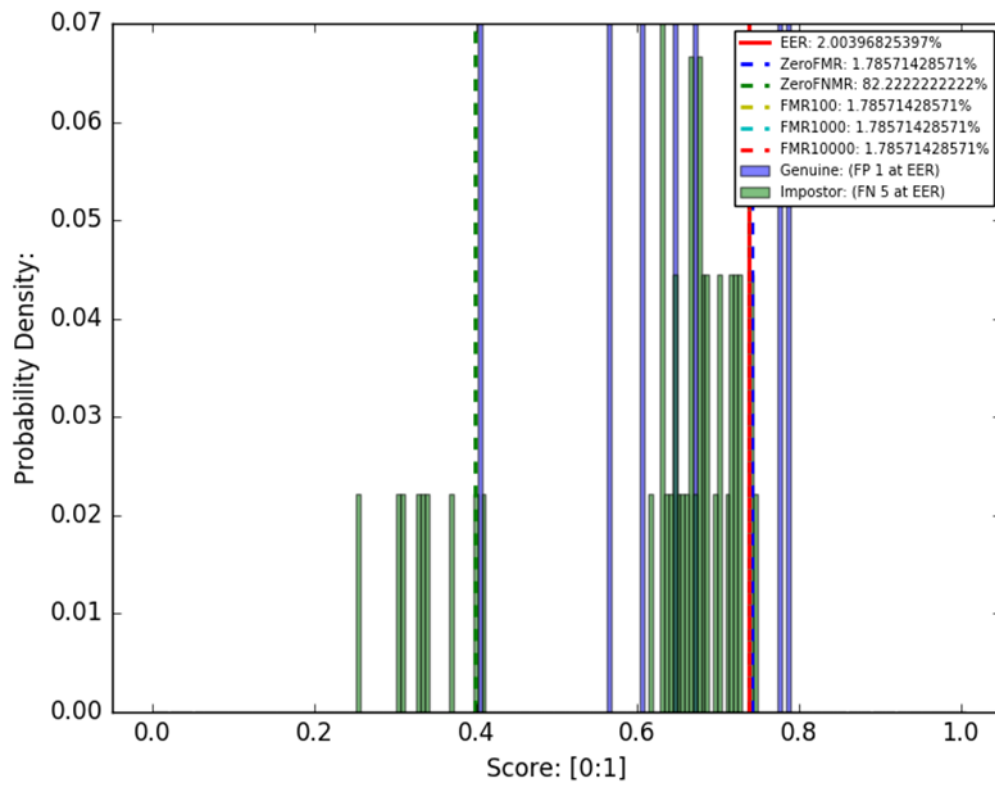


Figure 6-4: PDF for alignment algorithm with 3-byte data

6.2 Improvement of algorithms

6.2.1 Nearest-neighbourhood algorithm

Improvement of the nearest-neighbourhood algorithm yielded the results in Figure 6-5. The EER of 0,36 % is an improvement from 0,54 %. The values of FNMR for FMR = 1/100, FMR = 1/1000 and FMR = 1/10000 are all 0,36 % at the specified EER, which are the same as the EER value. The lowest FNMR for FMR = 0 % is zeroFNMR = 53,33 % and the lowest FMR for FNMR = 0 % is zeroFMR = 0,71 %. This execution shows improvement on the number of false negatives which has been reduced to 2. Most of the impostor attempts match score ranges from about 0,36 to 0,53, which makes it easy to select a threshold when using algorithm in an application. There is very little overlap between the genuine and impostor attempts.

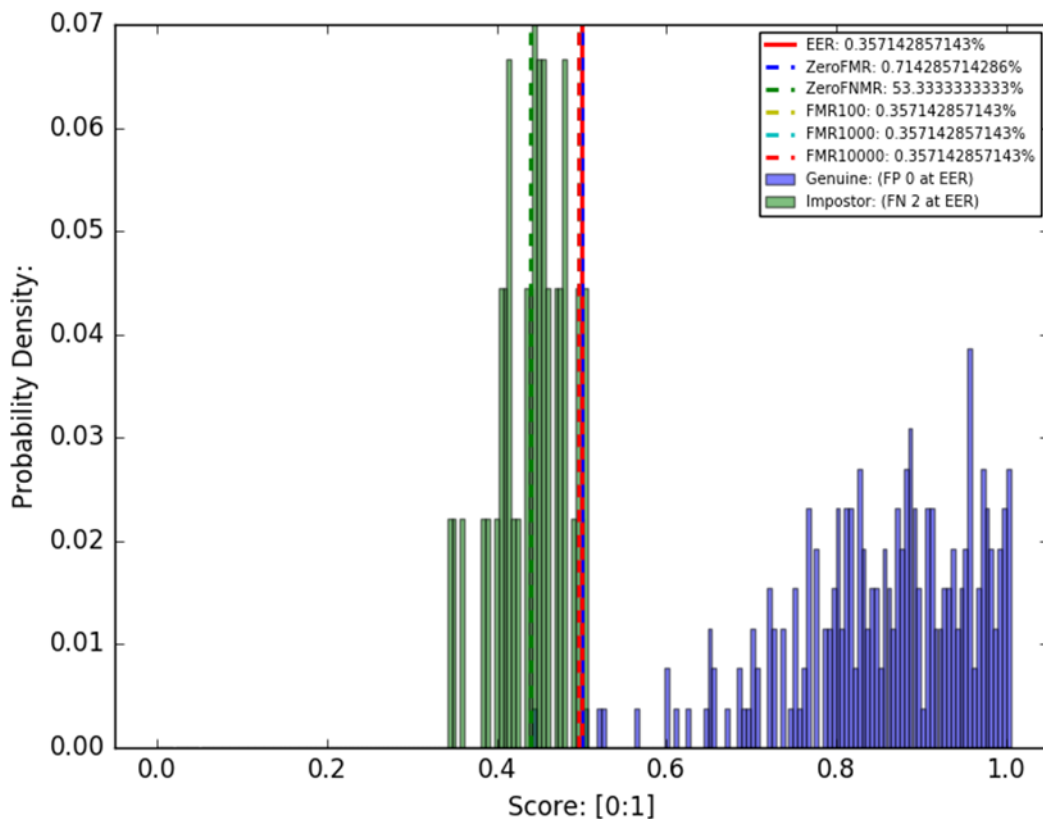


Figure 6-5: Nearest-neighbourhood algorithm adapted for 3-byte data

6.2.2 Alignment algorithm

The alignment algorithm improved for use with 3-byte data yields the results below in Figure 6-6. The alignment algorithm optimised for use with compact-card format shows a significant improvement from using the algorithm for 3-byte in the same way that it is used for the 5-byte data. The EER produced by this algorithm is 4,90 % with two genuine attempts being dubbed as impostors and 15 false negatives at the specified EER value.

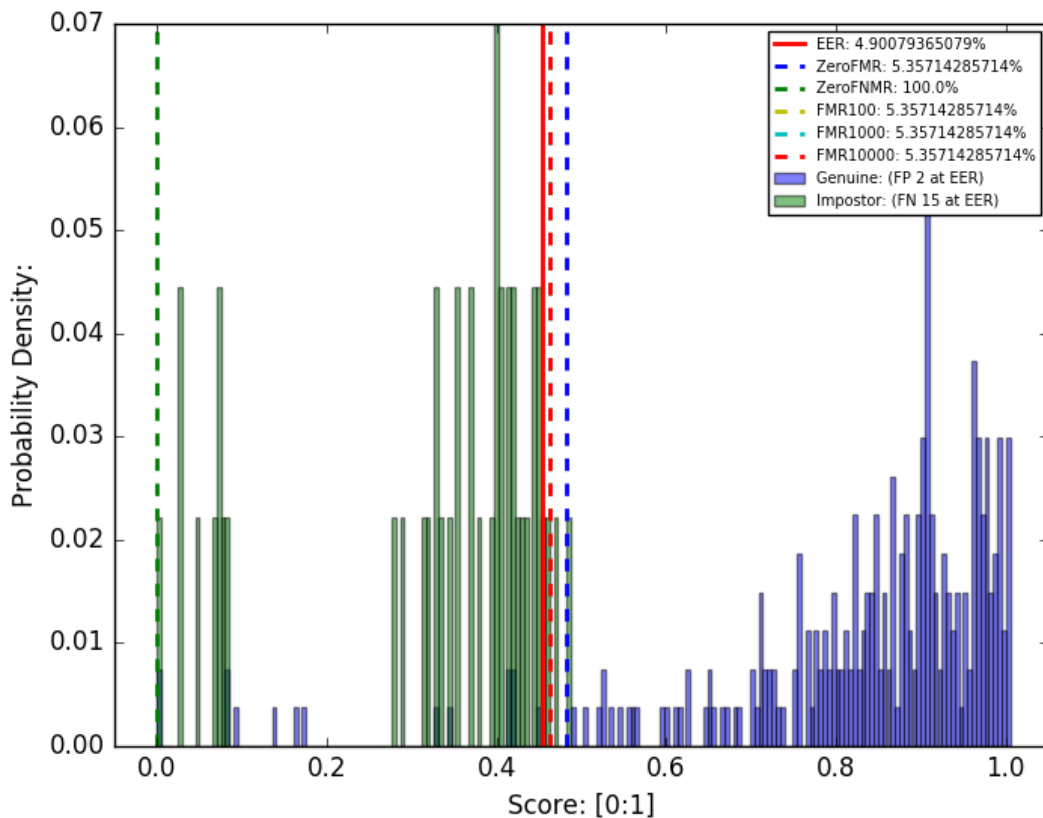


Figure 6-6: Alignment algorithm adapted for 3-byte data

The FMR100, FMR1000 and FMR10000 specify the values of FNMR for FMR = 1/100, FMR = 1/1000 and FMR = 1/10000. All of them are 5,36 % at the specified EER. The lowest FNMR for FMR = 0 % is zeroFNMR = 100 % and the lowest FMR for FNMR = 0 % is zeroFMR = 7,5 %. The probability density function seems almost similar to the one that was produced from executing the algorithm with the 5-byte data, except that the impostor attempts are more closely packed in this case than in the latter. This shows a significant improvement from the algorithm execution with the 3-byte data that was not optimized for compact-card format use.

6.3 ANOVA

6.3.1 Y-coordinate

These results, with an F -value of 0,008029 being less than $F_{critical} = 3,856075$, lead to acceptance of the null hypothesis for the y -coordinate. Accepting the null hypothesis means that the sample shows evidence of the population means between the y -coordinates in FVC2000 being equal with 95 % confidence.

$F = 0,03288$ is less than $F_{critical} = 3,856075$, therefore, one can accept the null hypothesis for the y -coordinates and conclude that the population means between the y -coordinates are equal with 95 % confidence in the FVC2002 database.

6.3.2 X-coordinate

The F -value of 0,010356 being less than $F_{critical} = 3,856075$, leads to acceptance of the null hypothesis for the x -coordinate. Accepting the null hypothesis means that the sample shows evidence of the population means between the x -coordinates being equal with 95 % confidence in the FVC2000 database.

In FVC2002, the single-factor ANOVA for x -coordinates produced an F -value of 0,089959 which is lower than $F_{critical} = 3,856075$. This leads to the null hypothesis being accepted and to the conclusion that there is no evidence of the population means between the x -coordinates being different from one another with 95 % confidence.

6.3.3 Orientation

These results, with the F -value = $9,02E-05$ being less than $F_{critical} = 3,00513$, lead to acceptance of the null hypothesis for the orientation in FVC 2000. Accepting the null hypothesis means that the sample shows evidence of population means between the orientations being equal with 95 % confidence.

In FVC2002, the single-factor ANOVA for orientation value produced an F -value of 0,999486 which is lower than $F_{critical} = 3,856075$. This leads to the null hypothesis being accepted and to the conclusion that there is no evidence of the population means between the orientations being different from one another with 95 % confidence.

6.3.4 Summary of ANOVA analysis

For all the values of x -, y - and θ , the null hypothesis has been accepted. This means that there is 99 % confidence that the population means between formats (3-byte, 5-byte and normal) are equal. This dismisses the formats from being the cause of accuracy loss when using the on-card representation for matching according to ANOVA results, but points to algorithms as the main culprit in the loss of accuracy.

6.4 Validation of results

The results obtained from the matchers are comparable to those in the literature. Proper care was taken to ensure that the algorithms were implemented as similarly to the respective literature as possible. It should be noted however, that some details were withheld in the literature. Another thing to be noted about the nearest-neighbourhood algorithm is that the FVC databases used are not the publicly available databases that were used in this study.

In comparison to literature, the nearest-neighbourhood algorithm performs with an EER of between 0,54 % and 0,17 %. The alignment algorithm used from literature [56] is reported to perform with an EER of 6,0 %. In [9], the EER for record format is 4,90 %, whereas the EER for compact card format is 2,00 %. From [9] there was no differentiation of algorithm types. However, the results that were obtained from implementations in this study are comparable to the field.

Comparing these results to the 5-byte data on the same database, the alignment algorithm shows an accuracy change of about 2,90 %. That means the 5-byte format has significantly larger amounts of accuracy loss than the 3-byte format. This is almost double the EER for the compact-card format. This is also not as expected from the assumption and report from [9] which claims better results when using the record format.

Table 6-1 shows the match score differences across the original implemented algorithms and the minutia points format. The PDF graphs for the 3-byte data have produced a smaller EER than the 5-byte data. However, the distribution of match scores presents the 3-byte data as causing accuracy loss. The match scores from the 3-byte data are quite

high even for impostors and that makes it hard to select a suitable matching threshold when using the algorithm in an application. That means although EER is a good measure, it should not be used alone to judge the performance of algorithms.

Table 6-1: Match scores in original implementation

ISO/IEC format	Alignment algorithm (%)	Nearest-neighbourhood algorithm (%)
3-byte	2,00	0,54
5-byte	4,90	0,18
Difference	2,90	0,37

With the x , y and θ variables being converted for the 3-byte representation, and the 5-byte representation only having the θ value converted, under normal circumstances the 5-byte representation would be expected to perform better than the 3-byte representation. In this case the 3-byte representation gave better EERs than the 5-byte representation for the alignment algorithm, but the performance of both the algorithms for the 5-byte data is indeed better than it is for the 3-byte data.

Table 6-2: Match scores on the improved algorithms

ISO/IEC format	Alignment algorithm (%)	Nearest-neighbourhood algorithm (%)
3-byte	4,900	0,35

Comparing the results according to matchers, the nearest-neighbourhood algorithm has a better overall performance across the minutiae formats; whereas the alignment algorithm has the worst overall performance. The evidence of this is seen in Table 6-2 when the nearest-neighbourhood algorithm only has a 0,36 % accuracy drop when changing from 5- to 3-byte data, but the alignment algorithm has a drop of accuracy twice as bad on 5-byte than it does with the 3-byte data.

Improvement of the algorithms - by adjusting the comparison thresholds with the same scaling factor that was used to adjust the minutia points for the on-card comparison format and the record format - has significantly improved the algorithm performance, especially

the alignment algorithm as the EER of the algorithm with 5-byte data is exactly the same as the EER for 3-byte data after modification of the algorithm.

Indeed, knowing that the algorithm implementation is for smart-card use, which consequently means the 3-byte data will be used, helps with making the algorithm performance approaching that of off-card matching [6].

Comparing the outcome on formats with the ANOVA results, there is evidence that the x and y variables although not having significantly high values of the f -statistic, the F of 0,008 and 0,01, are visibly higher than the orientation f -statistic of 0,0000902. That means the theta variable alone would not make a major difference but it also exposes the 5-byte data as displaying a bit of inconsistency, which will in turn affect the overall performance.

6.5 Summary

In this chapter, the alignment and nearest-neighbourhood results were presented, discussed and validated against literature.

The ANOVA test results showed no significant difference between the minutia points' formats with 95 % confidence. This result pointed to the algorithms as the culprit for accuracy loss, and implementation of the algorithms has proved that. The nearest-neighbourhood algorithm performed better than the alignment algorithm. The alignment algorithm's adaptation to on-card comparison format data showed significant improvement of the accuracy which performed close to the record format. The next chapter presents a conclusion to the study.

Chapter 7: Conclusion

South Africa introduced smart ID cards in 2013 for identification and authentication of South African citizens [62]. This came as an improvement on the old green bar-coded ID book in which authentication was not secure enough. However, the use of the smart-card for authentication comes with new problems. The minutia points that are stored in the smart-card, due to standardisation, appear to cause accuracy problems when used with on-card minutiae comparison algorithms.

7.1 Research synopsis

This study set out to investigate how the compact-card and record formats of minutia points affect the comparison accuracy and to also find out the algorithm that performs better with the minutiae format developed for use in smart-cards. The study begins with an analysis of the current state of the art with regard to minutiae-matching algorithms for on-card comparison and also presents an analysis on how problems pertaining to the minutiae formats have been addressed.

This was followed by identification of a problem which had not been studied in depth before, pertaining to the formats of minutia points used for on-card comparison. The methodology of the study was then presented and algorithm designs were presented. The design of analysis of variance is also shown for the analysis of minutia points, and in doing so it was revealed that the minutia points are not significantly different from one another; although differences exist, they are not significant. This led to a detailed study of the MOC algorithms, where two algorithms were selected due to their distinct basic structures.

The two categories of matching algorithms used are the local and global features based algorithms. The nearest-neighbourhood algorithm and the alignment algorithm each fall into these categories respectively. These matching algorithms were implemented and also verified through unit tests to ensure that they are within their specifications in addition to implementing matching algorithms, the minutia points were also encoded in the 3-byte and 5-byte formats. The minutiae were also verified to be correctly converted to either formats.

After the implementations, thorough comparisons of minutia points were made through ANOVA, and a thorough analysis of the matching algorithms performance was made

through matching procedures. The results from the algorithms were verified through past literature and found to be comparable.

The aim of these implementations and analyses was to be able to reach the research objectives stated in Chapter 1. Additionally, to satisfy the research goal which is to investigate how the ISO/IEC 19797-2:2011 template for on-card matching affects the accuracy of different MOC algorithm approaches. The results showed that the template does affect the accuracy and the effect is not the same on each of the algorithms. However, through manipulation of the resolutions that cause some loss of accuracy, the algorithms' performance with 3-byte can be significantly improved.

The next section addresses each research objective and outlines how the study has contributed in each of the research objectives, leading to the fulfilment of the research goal and ultimately answering the research question. The following conclusions have been made with regard to the minutiae formats and with regard to the matching algorithms.

7.2 Significant contributions of the study

The significant contributions of this study are explained in detail through the following subsections.

7.2.1 Analysis of different MOC algorithms

At this point, different MOC algorithms and their accuracy when used with the ISO/IEC 19794-2:2011 defined template for on-card comparison are analysed. The analysis reveals that the neighbourhood algorithm performs better than the alignment algorithm. This occurs with all cases of minutiae formats. The alignment algorithm has inconsistencies in terms of accepting and rejecting genuine and impostor attempts, especially with the CC data format.

When the algorithm implementation is conformed to operating with the on-card comparison format, the alignment algorithm has a better EER difference in the adapted version. Although the EERs in the alignment algorithm are not better than they are in the nearest-neighbourhood algorithm, there is a cooperation from the alignment algorithm for adapting to use with the on-card comparison format.

Both algorithms, in the original and adapted implementations, perform better with the 5-byte data, which is in line with the hypothesis which was motivated by results from [9] that the 5-byte data make the algorithms perform better.

During experimentation, it was revealed that primary knowledge of the data format is what makes the algorithm perform better with that data format. That is, primarily implementing an algorithm with 3-byte data in mind may cause the algorithm to perform worse with 5-byte data.

7.2.2 Investigation of the accuracy change

The investigation here focuses on the accuracy change when using the ISO/IEC 19794-2:2011 on-card comparison format and the ISO/IEC 19794-2:2005 record format for on-card comparison. This investigation is conducted to determine whether the accuracy change is caused by the inefficiency of the algorithm, or exacerbated by the template format.

The accuracy results in terms of EER show a significant drop when the format changes from 5- to 3-byte minutia points. When moving from 3-byte format to 5-byte format the EER gets lower on the nearest-neighbourhood algorithm, which indicates a better performance of the algorithm – in terms of EER. However, for the alignment algorithm, when moving from 3-byte format to 5-byte format the EER gets higher but through the PDF, it was proven that it does not necessarily mean the 3-byte performance was better. Explanation of the 3-byte data having lower EER and higher match scores is due to the fact that the resolution change to 3-byte causes points to seem closer to one another than they really are and therefore get covered by comparison thresholds.

The various algorithms do not handle the format change in a similar way. The neighbourhood algorithm shows an EER drop when the format changes from the 5- to the 3-byte data format. The alignment algorithm, in addition to being the worst-performing algorithm, shows the bigger accuracy drop when the minutiae format is changed from 5- to 3-byte. This shows that the algorithm cannot handle the 3-byte data and cannot be trusted for use in fingerprint-secured applications.

According to the ANOVA tests that were run on the data, it can be concluded with 95 % confidence that there is no significant difference between the minutia points. That would raise an expectation of having no changes when running the data through the algorithms,

although the algorithms do show a significant difference in accuracy when changing from 5-byte to 3-byte minutiae formats.

The alignment algorithm with an EER of 2,00 % – which is 1,46 more than the EER from the nearest-neighbourhood algorithm of 0,54 % – is evidence that the nearest-neighbourhood algorithm is performing better than the alignment algorithm with the 3-byte data format. Even the number of false positives at this EER is lower for the nearest-neighbourhood algorithm, and so is the number of false negatives.

Similarly to the previous analysis, the performance of the algorithms with the record-format data shows that the nearest-neighbourhood algorithm performs better than the alignment algorithm. Additionally, the difference in EER for 5-byte data on the alignment algorithm is 4,72 more than it is in the nearest-neighbourhood algorithm. This is over 12 times more than the EER in the nearest-neighbourhood algorithm.

Although the algorithms have their setbacks when it comes to handling the format changes, and even though the minutiae templates display a very small insignificant difference between the two formats according to ANOVA, the minutia points do exacerbate the loss of accuracy.

7.2.3 Recommendation for a suitable MOC algorithm

The third significant contribution that came out of this study concerns the recommendation of a suitable MOC algorithm that can be used with the ISO template format and at the same time can yield better results in terms of accuracy.

Overall, the nearest-neighbourhood algorithm performs better than the alignment algorithm. However, the alignment algorithm with improvements for handling compact-card data performs better than the nearest-neighbourhood algorithm. With proper improvements on the overall performance of the alignment algorithm, it will be able to have a good performance even for the 3-byte data since the resolution issue will be properly handled.

The recommended algorithm for on-card use is the nearest-neighbourhood algorithm because it generally and naturally handles the format change better. However, if the alignment algorithm can be improved to have accuracies similar to the nearest-

neighbourhood algorithm, it will yield performance results that approach the 5-byte format performance.

7.3 Future studies

Future studies may look into better algorithms for the trigonometric functions that are needed by the matching algorithms, other than the CORDIC and the Newton-Raphson methods which are time-consuming and hence make the matching algorithms slower than they should be. In real-time systems, matching algorithms need to be faster in order for the authentication and identification procedure to be realistic.

Due to time constraints, the algorithms were implemented using Python. A worthwhile future study would be to implement them on the actual smart-card to test, utilise and also take advantage of the performance of the algorithms in a real-life application.

Further, since the alignment algorithm used already has successors which have been improved to perform better, those successor algorithms can further be improved for on-card use, keeping in mind the 3-byte data format and assessing whether the performance improvement also approaches that of the 5-byte data format.

REFERENCES

- [1] A. K. Jain, J. Feng, and K. Nandakumar, "Fingerprint matching". *Computer (Long Beach, Calif.)*, vol. 43, no. 2, pp. 36–44, 2010.
- [2] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. Springer, 2009.
- [3] K. M. Shelfer and J. D. Procaccino, "Smart card evolution," *Commun. ACM*, vol. 45, no. 7, pp. 83–88, 2002.
- [4] E. Tabassi, P. Grother, and W. Salamon, "Minutiae interoperability", *BIOSIG 2009 Biometrics Electron. Signatures*, pp. 13–30, 2009.
- [5] F. Benhammadi and K. B. Bey, "Embedded Fingerprint Matching on Smart Card", *Int. J. Pattern Recognit. Artif. Intell.*, vol. 27, no. 2, p. 1350006, Mar. 2013.
- [6] P. Grother and W. Salamon, "An Assessment of ISO/IEC 7816 Card-Based Match-on-Card Capabilities", NIST, 2007.
- [7] C. Tai Pang, Y. Wei Yun, J. Xudong, T. P. Chen, Y. W. Yau, and X. Jiang, "On-Card Matching", in *Encyclopedia of Biometrics*, no. February, Z. L. Stan and A. Jain, Eds. Springer, pp. 1014–1021, 2009.
- [8] A. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition", *Circuits Syst. Video ...*, vol. 14, no. 1, pp. 4–20, 2004.
- [9] T. Yoshida and S. Hangai, "A Study on Accuracy and Problems in using ISO / IEC 19794-2 Finger Minutiae Formats for Automated Fingerprint Verification", 2010.
- [10] Y. H. B. Yahaya, M. R. M. B. M. Isa, and M. I. Bin Aziz, "Fingerprint biometrics authentication on smart card", in *2009 International Conference on Computer and Electrical Engineering, ICCEE 2009*, vol. 2, pp. 671–673, 2009.
- [11] S. Liu and M. Silverman, "A practical guide to biometric security technology", *IT Prof.*, 2001.
- [12] R. Dhamija and A. Perrig, "Déjà Vu: A User Study Using Images for Authentication", no. 9th USENIX Security Symposium, 2000.
- [13] Y. W. Yun and C. T. P. Lawrence, "An Introduction to Biometric Match-On-Card". pp. 35–42, 2005.
- [14] A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of biometrics*, 2007.
- [15] A. Narayanan and G. Chhabra, "Reducing Cyber Threats: via a Multimodal Biometric System", vol. 3, no. 2, pp. 10–14, 2013.
- [16] L. Cardoso, A. Barbosa, F. Silva, A. M. G. Pinheiro, and H. Proenca, "Iris biometrics: Synthesis of degraded ocular images", *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 7, pp. 1115–1125, 2013.

- [17] M. Betzel, "Privacy Year in Review: Recent Changes in the Law of Biometrics", *ISJLP*, 2004.
- [18] S. Alliance, "Smart Cards and Biometrics", *available from www.smartcardalliance.org*, 2011.
- [19] G. Bella, S. Bistarelli, and F. Martinelli, "Biometrics to Enhance Smartcard Security", *Secur. Protoc. 11th Int. Work. Cambridge, UK, April 2-4, 2003, Revis. Sel. Pap.*, vol. 3364 LNCS, pp. 324–332, 2005.
- [20] A. Matthews, "Low cost attacks on smart cards: the electromagnetic sidechannel", 2006.
- [21] E. T. Schneiderman, "Historical Examination of Data Breaches in New York State", State of New York, 2014.
- [22] A. Jain, L. Hong, and S. Pankanti, "Biometric identification", *Commun. ACM*, 2000.
- [23] R. M. Bolle, A. W. Senior, N. K. Ratha, and S. Pankanti, "Fingerprint Minutiae: A Constructive Definition", *Lect. Notes Comput. Sci.*, vol. 2359, pp. 58–66, 2006.
- [24] I. Standard, *ISO IEC 19794-2:2011 Information Technology - Biometrics Data Interchange Formats - Part 2*. 2011.
- [25] "ISO/IEC 19794-2:2011 - Information technology -- Biometric data interchange formats -- Part 2: Finger minutiae data." [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=50864. [Accessed: 13-Aug-2015].
- [26] P. Grother and W. Salamon, "Performance of Fingerprint Match-on-Card Algorithms Evaluation Plan", NIST, 2007.
- [27] P. Grother, W. Salamon, C. Watson, M. Indovina, and P. Flanagan, "MINEX II Performance of Fingerprint Match-on-Card Algorithms Phase IV Report", vol. 7477, no. Revision II, 2011.
- [28] S. W. P. Marios Savvides, Jingu Heo, *Handbook of Biometrics*. 2008.
- [29] A. K. Jain, L. Hong, S. Pankanti, and R. Bolle, "An identity-authentication system using fingerprints", *Proc. IEEE*, vol. 85, no. 9, pp. 1365–1388, 1997.
- [30] A. Jain, L. Hong, and R. Bolle, "On-Line Fingerprint Verification", *Pattern Anal. Mach. ...*, vol. 19, no. 4, pp. 302–314, 1997.
- [31] R. Sanchez-rei, C. Sanchez-avila, R. Sanchez-Reillo, and C. Sanchez-avila, "Fingerprint Verification Using Smart Cards for Access Control Systems", in *2001 IEEE 35th International Carnahan Conference on Security Technology*, vol. 38, no. 12, pp. 250–253, 2002.
- [32] Sung Bum Pan *et al.*, "An ultra-low memory fingerprint matching algorithm and its implementation on a 32-bit smart card", *IEEE Trans. Consum. Electron.*, vol. 49, no. 2, pp. 453–459, May 2003.

- [33] Y. Gil, D. Moon, S. Pan, and Y. Chung, "Fingerprint Verification System involving Smart Card", *Moon*, pp. 1–16, 2002.
- [34] Y. S. Moon, K. F. Fong, and K. C. Chan, "Secure and Fast Fingerprint Authentication on Smart Card.pdf", pp. 3–7, 2005.
- [35] T. Chouta and J. Danger, "A small and high-performance coprocessor for fingerprint match-on-card", in *2012 15th Euromicro Conference on Digital System Design (DSD)*, 2012, pp. 915–922.
- [36] H. K. Lam, W. Y. Yau, T. P. Chen, Z. Hou, and H. L. Wang, "Fingerprint Pre-Alignment for Hybrid Match-On-Card System", In *Information, Communications & Signal Processing, 2007 6th International Conference on IEEE*, Singapore, pp. 3–6, 2007.
- [37] T. Chen, W. Yau, and F. Way, "ISO / IEC standards for on-card biometric comparison Xudong Jiang", *Int. J. Biom.*, vol. 5, no. 1, pp. 30–52, 2013.
- [38] S. Bistarelli, F. Santini, and A. Vaccarelli, "An asymmetric fingerprint matching algorithm for Java Card TM", *Pattern Anal. Appl.*, vol. 9, no. 4, pp. 359–376, 2006.
- [39] T. Cucinotta, M. Di Natale, and R. Brigo, "A fingerprint matching algorithm for programmable smart cards", *Inf. Secur. Bull. J.*, 2005.
- [40] South Africa Protection of Personal Information Act, "South Africa Protection of Personal Information Act, 2013", *Natl. Gazettes, No 37067*, no. 10505, pp. 1–148, 2013.
- [41] D. Carnegie, *How to win friends and influence people*. Simon and Schuster, 1981.
- [42] D. Maio, D. Maltoni, and R. Cappelli, "FVC2000: Fingerprint verification competition", *IEEE Trans.*, 2002.
- [43] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, "FVC2002: Fingerprint verification competition", *Proc. Int. Conf. Pattern Recognit.*, pp. 744–747, 2002.
- [44] D. Maio, D. Maltoni, R. Cappelli, and J. Wayman, "FVC2004: Third fingerprint verification competition", *Biometric*, 2004.
- [45] R. Cappelli, D. Maio, D. Maltoni, J. L. Wayman, and A. K. Jain, "Performance evaluation of fingerprint verification systems", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 1, pp. 3–17, 2006.
- [46] M. Barrenechea, J. Altuna, and M. Mendicute, "A low-cost FPGA-based embedded fingerprint verification and matching system", *Intell. Tech.*, 2009.
- [47] R. M. McCabe and M. D. Garris, "Summary of April 2005 ANSI / NIST Fingerprint Standard Update Workshop Summary of April 2005 ANSI / NIST Fingerprint Standard Update Workshop", no. April 2005, 2005.
- [48] F. L. Podio *et al.*, "CBEFF - Common Biometric Exchange File Format", NIST, 2001.
- [49] I. Standard, "ISO/IEC 19794-2:2005 - Information technology -- Biometric data

interchange formats -- Part 2: Finger minutiae data", *ISO/IEC 19794-2:2005 - Information technology -- Biometric data interchange formats -- Part 2: Finger minutiae data*, 2005. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38746. [Accessed: 27-May-2016].

- [50] J. V Freeman and M. J. Campbell, "One-Way Analysis Of Variance", *Collection Of Statistical Papers First Published in Scope*, Scope papers, pp 73-77, 2007.
- [51] B. Illowsky and S. Dean, *Introductory Statistics*. Houston, Texas: Openstax College, 2013.
- [52] W. Penny and R. Henson, "Analysis of variance", *Parametr. Mapp. Anal. ...*, 2006.
- [53] H. R. Lindman, "Analysis of Variance, One-Way, Fixed Effects", in *Analysis of Variance in Experimental Design*, New York, NY: Springer New York, pp. 13–45, 1992.
- [54] B. Winer, "Statistical principles in experimental design". Vol. 2, New York: McGraw-Hill, 1971.
- [55] H. R. Lindman, *Analysis of variance in experimental design*. Springer New York, 1992.
- [56] S. Pan, Y. Gil, D. Moon, Y. Chung, and C. Park, "A memory-efficient fingerprint verification algorithm using a multi-resolution accumulator array", *ETRI J.*, 2003.
- [57] Michael Still, "The Definitive Guide to ImageMagick - Michael Still - Google Books." [Online]. Available: https://books.google.co.za/books?hl=en&lr=&id=6KVZ8Ya6a8cC&oi=fnd&pg=PR15&dq=imagemagick&ots=CfASLmBLz0&sig=M6E-yumkSN9_L3X6JanVcLLZ90M#v=onepage&q=imagemagick&f=false. [Accessed: 09-May-2017].
- [58] Sun Microsystems, "Java Card Applet Developer's Guide", 1998.
- [59] M. Cornea-Hasegan and R. Golliver, "Correctness proofs outline for Newton-Raphson based floating-point divide and square root algorithms", *14th IEEE Symp. ...*, 1999.
- [60] L. Wang and M. Schulte, "Decimal floating-point square root using Newton-Raphson iteration", *Appl. Syst.*, 2005.
- [61] K. Sharat, B. V Uma, and D. M. Sagard, "Calculation of Sine and Cosine of an Angle using the CORDIC Algorithm", *IJITR) Int. J. Innov. Technol. Res.*, no. 2, pp. 891–895.
- [62] "The smart ID card is here! | Vuk'uzenzele." [Online]. Available: <http://www.vukuzenzele.gov.za/smart-id-card-here>. [Accessed: 10-Feb-2017].

APPENDICES

Appendix A: Nearest-neighbourhood algorithm

```
dissimilarity.py x
166 #Step 3 § 4
167 def globalDissimilarity(edDiff, rcDiff, draDiff, odDiff):
168     """Calculate Global dissimilarity value
169     Args:
170     Returns:
171     Raises:
172     """
173
174     #1. Calculate weights
175     # 1.1. declare bounding Box (feature acceptance threshold)
176     edDiffThr = 5
177     rcDiffThr = 3
178     draDiffThr = 8
179     odDiffThr = 10
180
181     #unknown weight value
182     w = 5
183
184     edWght = float((w / edDiffThr))
185     rcWght = float(float(w) / float(rcDiffThr))
186     draWght = (float(w) / float(draDiffThr))
187     odWght = (float(w) / float(odDiffThr))
188
189     #2. Calculate corresponding weight values
190     edWghtDiff = edDiff * edWght
191     rcWghtDiff = rcDiff * rcWght
192     draWghtDiff = draDiff * draWght
193     odWghtDiff = odDiff * odWght
194
195     #3. Calculate the global dissimilarity vale
196     neighDissimilarity = edWghtDiff + rcWghtDiff + draWghtDiff + odWghtDiff
197
198     return (neighDissimilarity)
```

Appendix B: Alignment algorithm

```
alignment_algorithm4.py
27
28 #global variables
29 bucketX = [0] * ((imHeight/bucketIncrements) * 2)
30 bucketY = [0] * ((imHeight/bucketIncrements) * 2)
31 bucketT = [0] * 36
32 x3 = []
33 y3 = []
34 t3 = []
35
36 minutia1 = sys.argv[1]
37 directory = sys.argv[2]
38
39 def spatialDiff( x1,y1, x2, y2 ):
42
43 def directionDiff( theta1, theta2):
53
54 def transParam(xi, yi, xj, yj, chgTheta):
58
59
60 def map(changeX, changeY, changeTheta, xpj ,ypj ,theta):
81
82 def mm(xi, yi, zi, xj, yj, zj):
109
110 def readCSIRMinutiaeFile(filePath):
150
151 def plottingPts (x,y, imgPath):
180
181 def bucketing(value, bucketLength):
189
190 def matchScore (nm,nq,nr) :
197
198 def findingBuckets(x1,y1,t1,x2,y2,t2,flag=1):
265
266
267 def alignment(path1, path2):
316
317
318 def multipleMathes(argv):
335
336 def oneMatch(argv):
342
343 #multipleMathes(sys.argv)
344 oneMatch(sys.argv)
345
```

```

alignment_algorithm4.py
267 def alignment(path1, path2):
268
269     img1Path = 'img.jpg'
270
271     x1, y1, t1, matched1 = readCSIRMinutiaeFile(path1)
272     x2, y2, t2, matched2 = readCSIRMinutiaeFile(path2)
273     m33=0
274     m33=findingBuckets(x1,y1,t1,x2,y2,t2)
275
276     # print "parameters not found"
277     if (max(bucketX) == 0):
278         m33 = findingBuckets(x1,y1,t1,x2,y2,t2,0)
279         print
280
281     #obtain
282     changeX = (bucketX.index(max(bucketX))-adjuster) *10
283     changeY = (bucketY.index(max(bucketY))-adjuster) *10
284     changeI = bucketI.index(max(bucketI)) *10
285
286     print ("The changeX is %s, changeY is %s and changeI is %s"%(changeX, changeY, changeI))
287     print("the ", m33)
288
289     #map minutia point into new values
290     for q in range(0,len(x2) ):
291         a, b, c = map(changeX, changeY, changeI, x2[q], y2[q], t2[q] )
292         x3.append(a)
293         y3.append(b)
294         t3.append(c)
295
296     print x1
297     print x3
298     print y1
299     print y3
300
301     #perform mm to get minutiae pairs
302     matchCount = mm(x1, y1, t1, x3, y3, t3)
303     print "count", matchCount
304
305     matchedX = max(bucketX)
306     matchedY = max(bucketY)
307     matchscore = matchScore(matchCount, len(x1), len(x2))
308
309     plottingPts(x1,y1,img1Path)
310     return matchscore

```

Appendix C: ANOVA F distribution values

T-12 Tables

Table entry for p is the critical value F^* with probability p lying to its right.

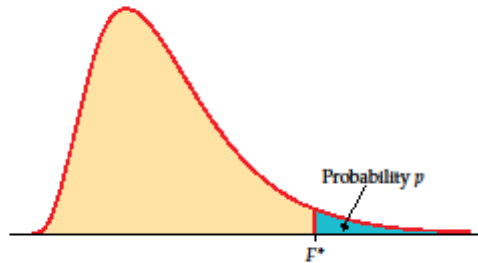


TABLE E										
F critical values										
		Degrees of freedom in the numerator								
p		1	2	3	4	5	6	7	8	9
Degrees of freedom in the denominator	1	.100	39.86	49.50	53.59	55.83	57.24	58.20	58.91	59.44
		.050	161.45	199.50	215.71	224.58	230.16	233.99	236.77	238.88
		.025	647.79	799.50	864.16	899.58	921.85	937.11	948.22	956.66
		.010	4052.2	4999.5	5403.4	5624.6	5763.6	5859.0	5928.4	5981.1
		.001	405284	500000	540379	562500	576405	585937	592873	598144
	2	.100	8.53	9.00	9.16	9.24	9.29	9.33	9.35	9.37
		.050	18.51	19.00	19.16	19.25	19.30	19.33	19.35	19.37
		.025	38.51	39.00	39.17	39.25	39.30	39.33	39.36	39.37
		.010	98.50	99.00	99.17	99.25	99.30	99.33	99.36	99.37
		.001	998.50	999.00	999.17	999.25	999.30	999.33	999.36	999.37
	3	.100	5.54	5.46	5.39	5.34	5.31	5.28	5.27	5.25
		.050	10.13	9.55	9.28	9.12	9.01	8.94	8.89	8.85
		.025	17.44	16.04	15.44	15.10	14.88	14.73	14.62	14.54
		.010	34.12	30.82	29.46	28.71	28.24	27.91	27.67	27.49
		.001	167.03	148.50	141.11	137.10	134.58	132.85	131.58	130.62
	4	.100	4.54	4.32	4.19	4.11	4.05	4.01	3.98	3.94
		.050	7.71	6.94	6.59	6.39	6.26	6.16	6.09	6.04
		.025	12.22	10.65	9.98	9.60	9.36	9.20	9.07	8.98
		.010	21.20	18.00	16.69	15.98	15.52	15.21	14.98	14.80
		.001	74.14	61.25	56.18	53.44	51.71	50.53	49.66	48.47
	5	.100	4.06	3.78	3.62	3.52	3.45	3.40	3.37	3.34
		.050	6.61	5.79	5.41	5.19	5.05	4.95	4.88	4.82
		.025	10.01	8.43	7.76	7.39	7.15	6.98	6.85	6.76
		.010	16.26	13.27	12.06	11.39	10.97	10.67	10.46	10.29
		.001	47.18	37.12	33.20	31.09	29.75	28.83	28.16	27.65
	6	.100	3.78	3.46	3.29	3.18	3.11	3.05	3.01	2.98
		.050	5.99	5.14	4.76	4.53	4.39	4.28	4.21	4.15
		.025	8.81	7.26	6.60	6.23	5.99	5.82	5.70	5.60
		.010	13.75	10.92	9.78	9.15	8.75	8.47	8.26	8.10
		.001	35.51	27.00	23.70	21.92	20.80	20.03	19.46	19.03
	7	.100	3.59	3.26	3.07	2.96	2.88	2.83	2.78	2.75
		.050	5.59	4.74	4.35	4.12	3.97	3.87	3.79	3.73
		.025	8.07	6.54	5.89	5.52	5.29	5.12	4.99	4.90
		.010	12.25	9.55	8.45	7.85	7.46	7.19	6.99	6.84
		.001	29.25	21.69	18.77	17.20	16.21	15.52	15.02	14.63

Table entry for p is the critical value F^* with probability p lying to its right.

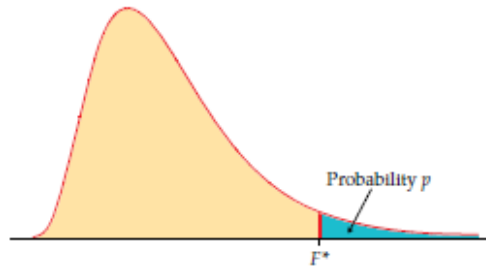


TABLE E

F critical values (continued)

Degrees of freedom in the numerator										
10	12	15	20	25	30	40	50	60	120	1000
60.19	60.71	61.22	61.74	62.05	62.26	62.53	62.69	62.79	63.06	63.30
241.88	243.91	245.95	248.01	249.26	250.10	251.14	251.77	252.20	253.25	254.19
968.63	976.71	984.87	993.10	998.08	1001.4	1005.6	1008.1	1009.8	1014.0	1017.7
6055.8	6106.3	6157.3	6208.7	6239.8	6260.6	6286.8	6302.5	6313.0	6339.4	6362.7
605621	610668	615764	620908	624017	626099	628712	630285	631337	633972	636301
9.39	9.41	9.42	9.44	9.45	9.46	9.47	9.47	9.47	9.48	9.49
19.40	19.41	19.43	19.45	19.46	19.46	19.47	19.48	19.48	19.49	19.49
39.40	39.41	39.43	39.45	39.46	39.46	39.47	39.48	39.48	39.49	39.50
99.40	99.42	99.43	99.45	99.46	99.47	99.47	99.48	99.48	99.49	99.50
999.40	999.42	999.43	999.45	999.46	999.47	999.47	999.48	999.48	999.49	999.50
5.23	5.22	5.20	5.18	5.17	5.17	5.16	5.15	5.15	5.14	5.13
8.79	8.74	8.70	8.66	8.63	8.62	8.59	8.58	8.57	8.55	8.53
14.42	14.34	14.25	14.17	14.12	14.08	14.04	14.01	13.99	13.95	13.91
27.23	27.05	26.87	26.69	26.58	26.50	26.41	26.35	26.32	26.22	26.14
129.25	128.32	127.37	126.42	125.84	125.45	124.96	124.66	124.47	123.97	123.53
3.92	3.90	3.87	3.84	3.83	3.82	3.80	3.80	3.79	3.78	3.76
5.96	5.91	5.86	5.80	5.77	5.75	5.72	5.70	5.69	5.66	5.63
8.84	8.75	8.66	8.56	8.50	8.46	8.41	8.38	8.36	8.31	8.26
14.55	14.37	14.20	14.02	13.91	13.84	13.75	13.69	13.65	13.56	13.47
48.05	47.41	46.76	46.10	45.70	45.43	45.09	44.88	44.75	44.40	44.09
3.30	3.27	3.24	3.21	3.19	3.17	3.16	3.15	3.14	3.12	3.11
4.74	4.68	4.62	4.56	4.52	4.50	4.46	4.44	4.43	4.40	4.37
6.62	6.52	6.43	6.33	6.27	6.23	6.18	6.14	6.12	6.07	6.02
10.05	9.89	9.72	9.55	9.45	9.38	9.29	9.24	9.20	9.11	9.03
26.92	26.42	25.91	25.39	25.08	24.87	24.60	24.44	24.33	24.06	23.82
2.94	2.90	2.87	2.84	2.81	2.80	2.78	2.77	2.76	2.74	2.72
4.06	4.00	3.94	3.87	3.83	3.81	3.77	3.75	3.74	3.70	3.67
5.46	5.37	5.27	5.17	5.11	5.07	5.01	4.98	4.96	4.90	4.86
7.87	7.72	7.56	7.40	7.30	7.23	7.14	7.09	7.06	6.97	6.89
18.41	17.99	17.56	17.12	16.85	16.67	16.44	16.31	16.21	15.98	15.77
2.70	2.67	2.63	2.59	2.57	2.56	2.54	2.52	2.51	2.49	2.47
3.64	3.57	3.51	3.44	3.40	3.38	3.34	3.32	3.30	3.27	3.23
4.76	4.67	4.57	4.47	4.40	4.36	4.31	4.28	4.25	4.20	4.15
6.62	6.47	6.31	6.16	6.06	5.99	5.91	5.86	5.82	5.74	5.66
14.08	13.71	13.32	12.93	12.69	12.53	12.33	12.20	12.12	11.91	11.72

(Continued)

TABLE E

F critical values (continued)

		Degrees of freedom in the numerator								
<i>p</i>		1	2	3	4	5	6	7	8	9
Degrees of freedom in the denominator	8	.100	3.46	3.11	2.92	2.81	2.73	2.67	2.62	2.56
		.050	5.32	4.46	4.07	3.84	3.69	3.58	3.50	3.39
		.025	7.57	6.06	5.42	5.05	4.82	4.65	4.53	4.36
		.010	11.26	8.65	7.59	7.01	6.63	6.37	6.18	5.91
		.001	25.41	18.49	15.83	14.39	13.48	12.86	12.40	11.77
	9	.100	3.36	3.01	2.81	2.69	2.61	2.55	2.51	2.44
		.050	5.12	4.26	3.86	3.63	3.48	3.37	3.29	3.18
		.025	7.21	5.71	5.08	4.72	4.48	4.32	4.20	4.03
		.010	10.56	8.02	6.99	6.42	6.06	5.80	5.61	5.35
		.001	22.86	16.39	13.90	12.56	11.71	11.13	10.70	10.11
	10	.100	3.29	2.92	2.73	2.61	2.52	2.46	2.41	2.35
		.050	4.96	4.10	3.71	3.48	3.33	3.22	3.14	3.02
		.025	6.94	5.46	4.83	4.47	4.24	4.07	3.95	3.78
		.010	10.04	7.56	6.55	5.99	5.64	5.39	5.20	4.94
		.001	21.04	14.91	12.55	11.28	10.48	9.93	9.52	8.96
	11	.100	3.23	2.86	2.66	2.54	2.45	2.39	2.34	2.27
		.050	4.84	3.98	3.59	3.36	3.20	3.09	3.01	2.90
		.025	6.72	5.26	4.63	4.28	4.04	3.88	3.76	3.59
		.010	9.65	7.21	6.22	5.67	5.32	5.07	4.89	4.63
		.001	19.69	13.81	11.56	10.35	9.58	9.05	8.66	8.12
	12	.100	3.18	2.81	2.61	2.48	2.39	2.33	2.28	2.21
		.050	4.75	3.89	3.49	3.26	3.11	3.00	2.91	2.80
		.025	6.55	5.10	4.47	4.12	3.89	3.73	3.61	3.44
		.010	9.33	6.93	5.95	5.41	5.06	4.82	4.64	4.39
		.001	18.64	12.97	10.80	9.63	8.89	8.38	8.00	7.48
	13	.100	3.14	2.76	2.56	2.43	2.35	2.28	2.23	2.16
		.050	4.67	3.81	3.41	3.18	3.03	2.92	2.83	2.71
		.025	6.41	4.97	4.35	4.00	3.77	3.60	3.48	3.31
		.010	9.07	6.70	5.74	5.21	4.86	4.62	4.44	4.19
		.001	17.82	12.31	10.21	9.07	8.35	7.86	7.49	6.98
	14	.100	3.10	2.73	2.52	2.39	2.31	2.24	2.19	2.12
		.050	4.60	3.74	3.34	3.11	2.96	2.85	2.76	2.65
		.025	6.30	4.86	4.24	3.89	3.66	3.50	3.38	3.21
		.010	8.86	6.51	5.56	5.04	4.69	4.46	4.28	4.03
		.001	17.14	11.78	9.73	8.62	7.92	7.44	7.08	6.58
	15	.100	3.07	2.70	2.49	2.36	2.27	2.21	2.16	2.09
		.050	4.54	3.68	3.29	3.06	2.90	2.79	2.71	2.64
		.025	6.20	4.77	4.15	3.80	3.58	3.41	3.29	3.12
		.010	8.68	6.36	5.42	4.89	4.56	4.32	4.14	3.89
		.001	16.59	11.34	9.34	8.25	7.57	7.09	6.74	6.26
	16	.100	3.05	2.67	2.46	2.33	2.24	2.18	2.13	2.06
		.050	4.49	3.63	3.24	3.01	2.85	2.74	2.66	2.59
		.025	6.12	4.69	4.08	3.73	3.50	3.34	3.22	3.05
		.010	8.53	6.23	5.29	4.77	4.44	4.20	4.03	3.78
		.001	16.12	10.97	9.01	7.94	7.27	6.80	6.46	5.98
	17	.100	3.03	2.64	2.44	2.31	2.22	2.15	2.10	2.03
		.050	4.45	3.59	3.20	2.96	2.81	2.70	2.61	2.55
		.025	6.04	4.62	4.01	3.66	3.44	3.28	3.16	3.06
		.010	8.40	6.11	5.19	4.67	4.34	4.10	3.93	3.79
		.001	15.72	10.66	8.73	7.68	7.02	6.56	6.22	5.75

TABLE E*F* critical values (continued)

Degrees of freedom in the numerator										
10	12	15	20	25	30	40	50	60	120	1000
2.54	2.50	2.46	2.42	2.40	2.38	2.36	2.35	2.34	2.32	2.30
3.35	3.28	3.22	3.15	3.11	3.08	3.04	3.02	3.01	2.97	2.93
4.30	4.20	4.10	4.00	3.94	3.89	3.84	3.81	3.78	3.73	3.68
5.81	5.67	5.52	5.36	5.26	5.20	5.12	5.07	5.03	4.95	4.87
11.54	11.19	10.84	10.48	10.26	10.11	9.92	9.80	9.73	9.53	9.36
2.42	2.38	2.34	2.30	2.27	2.25	2.23	2.22	2.21	2.18	2.16
3.14	3.07	3.01	2.94	2.89	2.86	2.83	2.80	2.79	2.75	2.71
3.96	3.87	3.77	3.67	3.60	3.56	3.51	3.47	3.45	3.39	3.34
5.26	5.11	4.96	4.81	4.71	4.65	4.57	4.52	4.48	4.40	4.32
9.89	9.57	9.24	8.90	8.69	8.55	8.37	8.26	8.19	8.00	7.84
2.32	2.28	2.24	2.20	2.17	2.16	2.13	2.12	2.11	2.08	2.06
2.98	2.91	2.85	2.77	2.73	2.70	2.66	2.64	2.62	2.58	2.54
3.72	3.62	3.52	3.42	3.35	3.31	3.26	3.22	3.20	3.14	3.09
4.85	4.71	4.56	4.41	4.31	4.25	4.17	4.12	4.08	4.00	3.92
8.75	8.45	8.13	7.80	7.60	7.47	7.30	7.19	7.12	6.94	6.78
2.25	2.21	2.17	2.12	2.10	2.08	2.05	2.04	2.03	2.00	1.98
2.85	2.79	2.72	2.65	2.60	2.57	2.53	2.51	2.49	2.45	2.41
3.53	3.43	3.33	3.23	3.16	3.12	3.06	3.03	3.00	2.94	2.89
4.54	4.40	4.25	4.10	4.01	3.94	3.86	3.81	3.78	3.69	3.61
7.92	7.63	7.32	7.01	6.81	6.68	6.52	6.42	6.35	6.18	6.02
2.19	2.15	2.10	2.06	2.03	2.01	1.99	1.97	1.96	1.93	1.91
2.75	2.69	2.62	2.54	2.50	2.47	2.43	2.40	2.38	2.34	2.30
3.37	3.28	3.18	3.07	3.01	2.96	2.91	2.87	2.85	2.79	2.73
4.30	4.16	4.01	3.86	3.76	3.70	3.62	3.57	3.54	3.45	3.37
7.29	7.00	6.71	6.40	6.22	6.09	5.93	5.83	5.76	5.59	5.44
2.14	2.10	2.05	2.01	1.98	1.96	1.93	1.92	1.90	1.88	1.85
2.67	2.60	2.53	2.46	2.41	2.38	2.34	2.31	2.30	2.25	2.21
3.25	3.15	3.05	2.95	2.88	2.84	2.78	2.74	2.72	2.66	2.60
4.10	3.96	3.82	3.66	3.57	3.51	3.43	3.38	3.34	3.25	3.18
6.80	6.52	6.23	5.93	5.75	5.63	5.47	5.37	5.30	5.14	4.99
2.10	2.05	2.01	1.96	1.93	1.91	1.89	1.87	1.86	1.83	1.80
2.60	2.53	2.46	2.39	2.34	2.31	2.27	2.24	2.22	2.18	2.14
3.15	3.05	2.95	2.84	2.78	2.73	2.67	2.64	2.61	2.55	2.50
3.94	3.80	3.66	3.51	3.41	3.35	3.27	3.22	3.18	3.09	3.02
6.40	6.13	5.85	5.56	5.38	5.25	5.10	5.00	4.94	4.77	4.62
2.06	2.02	1.97	1.92	1.89	1.87	1.85	1.83	1.82	1.79	1.76
2.54	2.48	2.40	2.33	2.28	2.25	2.20	2.18	2.16	2.11	2.07
3.06	2.96	2.86	2.76	2.69	2.64	2.59	2.55	2.52	2.46	2.40
3.80	3.67	3.52	3.37	3.28	3.21	3.13	3.08	3.05	2.96	2.88
6.08	5.81	5.54	5.25	5.07	4.95	4.80	4.70	4.64	4.47	4.33
2.03	1.99	1.94	1.89	1.86	1.84	1.81	1.79	1.78	1.75	1.72
2.49	2.42	2.35	2.28	2.23	2.19	2.15	2.12	2.11	2.06	2.02
2.99	2.89	2.79	2.68	2.61	2.57	2.51	2.47	2.45	2.38	2.32
3.69	3.55	3.41	3.26	3.16	3.10	3.02	2.97	2.93	2.84	2.76
5.81	5.55	5.27	4.99	4.82	4.70	4.54	4.45	4.39	4.23	4.08
2.00	1.96	1.91	1.86	1.83	1.81	1.78	1.76	1.75	1.72	1.69
2.45	2.38	2.31	2.23	2.18	2.15	2.10	2.08	2.06	2.01	1.97
2.92	2.82	2.72	2.62	2.55	2.50	2.44	2.41	2.38	2.32	2.26
3.59	3.46	3.31	3.16	3.07	3.00	2.92	2.87	2.83	2.75	2.66
5.58	5.32	5.05	4.78	4.60	4.48	4.33	4.24	4.18	4.02	3.87

(Continued)

TABLE E

F critical values (continued)

		Degrees of freedom in the numerator								
<i>p</i>		1	2	3	4	5	6	7	8	9
Degrees of freedom in the denominator	18	.100	3.01	2.62	2.42	2.29	2.20	2.13	2.08	2.04
		.050	4.41	3.55	3.16	2.93	2.77	2.66	2.58	2.51
		.025	5.98	4.56	3.95	3.61	3.38	3.22	3.10	3.01
		.010	8.29	6.01	5.09	4.58	4.25	4.01	3.84	3.71
		.001	15.38	10.39	8.49	7.46	6.81	6.35	6.02	5.76
	19	.100	2.99	2.61	2.40	2.27	2.18	2.11	2.06	2.02
		.050	4.38	3.52	3.13	2.90	2.74	2.63	2.54	2.48
		.025	5.92	4.51	3.90	3.56	3.33	3.17	3.05	2.96
		.010	8.18	5.93	5.01	4.50	4.17	3.94	3.77	3.63
		.001	15.08	10.16	8.28	7.27	6.62	6.18	5.85	5.59
	20	.100	2.97	2.59	2.38	2.25	2.16	2.09	2.04	2.00
		.050	4.35	3.49	3.10	2.87	2.71	2.60	2.51	2.45
		.025	5.87	4.46	3.86	3.51	3.29	3.13	3.01	2.91
		.010	8.10	5.85	4.94	4.43	4.10	3.87	3.70	3.56
		.001	14.82	9.95	8.10	7.10	6.46	6.02	5.69	5.44
	21	.100	2.96	2.57	2.36	2.23	2.14	2.08	2.02	1.98
		.050	4.32	3.47	3.07	2.84	2.68	2.57	2.49	2.42
		.025	5.83	4.42	3.82	3.48	3.25	3.09	2.97	2.87
		.010	8.02	5.78	4.87	4.37	4.04	3.81	3.64	3.51
		.001	14.59	9.77	7.94	6.95	6.32	5.88	5.56	5.31
	22	.100	2.95	2.56	2.35	2.22	2.13	2.06	2.01	1.97
		.050	4.30	3.44	3.05	2.82	2.66	2.55	2.46	2.40
		.025	5.79	4.38	3.78	3.44	3.22	3.05	2.93	2.84
		.010	7.95	5.72	4.82	4.31	3.99	3.76	3.59	3.45
		.001	14.38	9.61	7.80	6.81	6.19	5.76	5.44	5.19
	23	.100	2.94	2.55	2.34	2.21	2.11	2.05	1.99	1.95
		.050	4.28	3.42	3.03	2.80	2.64	2.53	2.44	2.37
		.025	5.75	4.35	3.75	3.41	3.18	3.02	2.90	2.81
		.010	7.88	5.66	4.76	4.26	3.94	3.71	3.54	3.41
		.001	14.20	9.47	7.67	6.70	6.08	5.65	5.33	5.09
	24	.100	2.93	2.54	2.33	2.19	2.10	2.04	1.98	1.94
		.050	4.26	3.40	3.01	2.78	2.62	2.51	2.42	2.36
		.025	5.72	4.32	3.72	3.38	3.15	2.99	2.87	2.78
		.010	7.82	5.61	4.72	4.22	3.90	3.67	3.50	3.36
		.001	14.03	9.34	7.55	6.59	5.98	5.55	5.23	4.99
	25	.100	2.92	2.53	2.32	2.18	2.09	2.02	1.97	1.93
		.050	4.24	3.39	2.99	2.76	2.60	2.49	2.40	2.34
		.025	5.69	4.29	3.69	3.35	3.13	2.97	2.85	2.75
		.010	7.77	5.57	4.68	4.18	3.85	3.63	3.46	3.32
		.001	13.88	9.22	7.45	6.49	5.89	5.46	5.15	4.91
	26	.100	2.91	2.52	2.31	2.17	2.08	2.01	1.96	1.92
		.050	4.23	3.37	2.98	2.74	2.59	2.47	2.39	2.32
		.025	5.66	4.27	3.67	3.33	3.10	2.94	2.82	2.73
		.010	7.72	5.53	4.64	4.14	3.82	3.59	3.42	3.29
		.001	13.74	9.12	7.36	6.41	5.80	5.38	5.07	4.83
	27	.100	2.90	2.51	2.30	2.17	2.07	2.00	1.95	1.91
		.050	4.21	3.35	2.96	2.73	2.57	2.46	2.37	2.31
		.025	5.63	4.24	3.65	3.31	3.08	2.92	2.80	2.71
		.010	7.68	5.49	4.60	4.11	3.78	3.56	3.39	3.26
		.001	13.61	9.02	7.27	6.33	5.73	5.31	5.00	4.76

TABLE E

F critical values (continued)

Degrees of freedom in the numerator										
10	12	15	20	25	30	40	50	60	120	1000
1.98	1.93	1.89	1.84	1.80	1.78	1.75	1.74	1.72	1.69	1.66
2.41	2.34	2.27	2.19	2.14	2.11	2.06	2.04	2.02	1.97	1.92
2.87	2.77	2.67	2.56	2.49	2.44	2.38	2.35	2.32	2.26	2.20
3.51	3.37	3.23	3.08	2.98	2.92	2.84	2.78	2.75	2.66	2.58
5.39	5.13	4.87	4.59	4.42	4.30	4.15	4.06	4.00	3.84	3.69
1.96	1.91	1.86	1.81	1.78	1.76	1.73	1.71	1.70	1.67	1.64
2.38	2.31	2.23	2.16	2.11	2.07	2.03	2.00	1.98	1.93	1.88
2.82	2.72	2.62	2.51	2.44	2.39	2.33	2.30	2.27	2.20	2.14
3.43	3.30	3.15	3.00	2.91	2.84	2.76	2.71	2.67	2.58	2.50
5.22	4.97	4.70	4.43	4.26	4.14	3.99	3.90	3.84	3.68	3.53
1.94	1.89	1.84	1.79	1.76	1.74	1.71	1.69	1.68	1.64	1.61
2.35	2.28	2.20	2.12	2.07	2.04	1.99	1.97	1.95	1.90	1.85
2.77	2.68	2.57	2.46	2.40	2.35	2.29	2.25	2.22	2.16	2.09
3.37	3.23	3.09	2.94	2.84	2.78	2.69	2.64	2.61	2.52	2.43
5.08	4.82	4.56	4.29	4.12	4.00	3.86	3.77	3.70	3.54	3.40
1.92	1.87	1.83	1.78	1.74	1.72	1.69	1.67	1.66	1.62	1.59
2.32	2.25	2.18	2.10	2.05	2.01	1.96	1.94	1.92	1.87	1.82
2.73	2.64	2.53	2.42	2.36	2.31	2.25	2.21	2.18	2.11	2.05
3.31	3.17	3.03	2.88	2.79	2.72	2.64	2.58	2.55	2.46	2.37
4.95	4.70	4.44	4.17	4.00	3.88	3.74	3.64	3.58	3.42	3.28
1.90	1.86	1.81	1.76	1.73	1.70	1.67	1.65	1.64	1.60	1.57
2.30	2.23	2.15	2.07	2.02	1.98	1.94	1.91	1.89	1.84	1.79
2.70	2.60	2.50	2.39	2.32	2.27	2.21	2.17	2.14	2.08	2.01
3.26	3.12	2.98	2.83	2.73	2.67	2.58	2.53	2.50	2.40	2.32
4.83	4.58	4.33	4.06	3.89	3.78	3.63	3.54	3.48	3.32	3.17
1.89	1.84	1.80	1.74	1.71	1.69	1.66	1.64	1.62	1.59	1.55
2.27	2.20	2.13	2.05	2.00	1.96	1.91	1.88	1.86	1.81	1.76
2.67	2.57	2.47	2.36	2.29	2.24	2.18	2.14	2.11	2.04	1.98
3.21	3.07	2.93	2.78	2.69	2.62	2.54	2.48	2.45	2.35	2.27
4.73	4.48	4.23	3.96	3.79	3.68	3.53	3.44	3.38	3.22	3.08
1.88	1.83	1.78	1.73	1.70	1.67	1.64	1.62	1.61	1.57	1.54
2.25	2.18	2.11	2.03	1.97	1.94	1.89	1.86	1.84	1.79	1.74
2.64	2.54	2.44	2.33	2.26	2.21	2.15	2.11	2.08	2.01	1.94
3.17	3.03	2.89	2.74	2.64	2.58	2.49	2.44	2.40	2.31	2.22
4.64	4.39	4.14	3.87	3.71	3.59	3.45	3.36	3.29	3.14	2.99
1.87	1.82	1.77	1.72	1.68	1.66	1.63	1.61	1.59	1.56	1.52
2.24	2.16	2.09	2.01	1.96	1.92	1.87	1.84	1.82	1.77	1.72
2.61	2.51	2.41	2.30	2.23	2.18	2.12	2.08	2.05	1.98	1.91
3.13	2.99	2.85	2.70	2.60	2.54	2.45	2.40	2.36	2.27	2.18
4.56	4.31	4.06	3.79	3.63	3.52	3.37	3.28	3.22	3.06	2.91
1.86	1.81	1.76	1.71	1.67	1.65	1.61	1.59	1.58	1.54	1.51
2.22	2.15	2.07	1.99	1.94	1.90	1.85	1.82	1.80	1.75	1.70
2.59	2.49	2.39	2.28	2.21	2.16	2.09	2.05	2.03	1.95	1.89
3.09	2.96	2.81	2.66	2.57	2.50	2.42	2.36	2.33	2.23	2.14
4.48	4.24	3.99	3.72	3.56	3.44	3.30	3.21	3.15	2.99	2.84
1.85	1.80	1.75	1.70	1.66	1.64	1.60	1.58	1.57	1.53	1.50
2.20	2.13	2.06	1.97	1.92	1.88	1.84	1.81	1.79	1.73	1.68
2.57	2.47	2.36	2.25	2.18	2.13	2.07	2.03	2.00	1.93	1.86
3.06	2.93	2.78	2.63	2.54	2.47	2.38	2.33	2.29	2.20	2.11
4.41	4.17	3.92	3.66	3.49	3.38	3.23	3.14	3.08	2.92	2.78

TABLE E

F critical values (continued)

		Degrees of freedom in the numerator								
p		1	2	3	4	5	6	7	8	9
Degrees of freedom in the denominator	28	.100	2.89	2.50	2.29	2.16	2.06	2.00	1.94	1.90
		.050	4.20	3.34	2.95	2.71	2.56	2.45	2.36	2.29
		.025	5.61	4.22	3.63	3.29	3.06	2.90	2.78	2.69
		.010	7.64	5.45	4.57	4.07	3.75	3.53	3.36	3.23
		.001	13.50	8.93	7.19	6.25	5.66	5.24	4.93	4.69
	29	.100	2.89	2.50	2.28	2.15	2.06	1.99	1.93	1.89
		.050	4.18	3.33	2.93	2.70	2.55	2.43	2.35	2.28
		.025	5.59	4.20	3.61	3.27	3.04	2.88	2.76	2.67
		.010	7.60	5.42	4.54	4.04	3.73	3.50	3.33	3.20
		.001	13.39	8.85	7.12	6.19	5.59	5.18	4.87	4.64
	30	.100	2.88	2.49	2.28	2.14	2.05	1.98	1.93	1.88
		.050	4.17	3.32	2.92	2.69	2.53	2.42	2.33	2.27
		.025	5.57	4.18	3.59	3.25	3.03	2.87	2.75	2.65
		.010	7.56	5.39	4.51	4.02	3.70	3.47	3.30	3.17
		.001	13.29	8.77	7.05	6.12	5.53	5.12	4.82	4.58
	40	.100	2.84	2.44	2.23	2.09	2.00	1.93	1.87	1.83
		.050	4.08	3.23	2.84	2.61	2.45	2.34	2.25	2.18
		.025	5.42	4.05	3.46	3.13	2.90	2.74	2.62	2.53
		.010	7.31	5.18	4.31	3.83	3.51	3.29	3.12	2.99
		.001	12.61	8.25	6.59	5.70	5.13	4.73	4.44	4.21
	50	.100	2.81	2.41	2.20	2.06	1.97	1.90	1.84	1.80
		.050	4.03	3.18	2.79	2.56	2.40	2.29	2.20	2.13
		.025	5.34	3.97	3.39	3.05	2.83	2.67	2.55	2.46
		.010	7.17	5.06	4.20	3.72	3.41	3.19	3.02	2.89
		.001	12.22	7.96	6.34	5.46	4.90	4.51	4.22	4.00
	60	.100	2.79	2.39	2.18	2.04	1.95	1.87	1.82	1.77
		.050	4.00	3.15	2.76	2.53	2.37	2.25	2.17	2.10
		.025	5.29	3.93	3.34	3.01	2.79	2.63	2.51	2.41
		.010	7.08	4.98	4.13	3.65	3.34	3.12	2.95	2.82
		.001	11.97	7.77	6.17	5.31	4.76	4.37	4.09	3.86
	100	.100	2.76	2.36	2.14	2.00	1.91	1.83	1.78	1.73
		.050	3.94	3.09	2.70	2.46	2.31	2.19	2.10	2.03
		.025	5.18	3.83	3.25	2.92	2.70	2.54	2.42	2.32
		.010	6.90	4.82	3.98	3.51	3.21	2.99	2.82	2.69
		.001	11.50	7.41	5.86	5.02	4.48	4.11	3.83	3.61
	200	.100	2.73	2.33	2.11	1.97	1.88	1.80	1.75	1.70
		.050	3.89	3.04	2.65	2.42	2.26	2.14	2.06	1.98
		.025	5.10	3.76	3.18	2.85	2.63	2.47	2.35	2.26
		.010	6.76	4.71	3.88	3.41	3.11	2.89	2.73	2.60
		.001	11.15	7.15	5.63	4.81	4.29	3.92	3.65	3.43
	1000	.100	2.71	2.31	2.09	1.95	1.85	1.78	1.72	1.68
		.050	3.85	3.00	2.61	2.38	2.22	2.11	2.02	1.95
		.025	5.04	3.70	3.13	2.80	2.58	2.42	2.30	2.20
		.010	6.66	4.63	3.80	3.34	3.04	2.82	2.66	2.53
		.001	10.89	6.96	5.46	4.65	4.14	3.78	3.51	3.30

TABLE E

F critical values (continued)

Degrees of freedom in the numerator										
10	12	15	20	25	30	40	50	60	120	1000
1.84	1.79	1.74	1.69	1.65	1.63	1.59	1.57	1.56	1.52	1.48
2.19	2.12	2.04	1.96	1.91	1.87	1.82	1.79	1.77	1.71	1.66
2.55	2.45	2.34	2.23	2.16	2.11	2.05	2.01	1.98	1.91	1.84
3.03	2.90	2.75	2.60	2.51	2.44	2.35	2.30	2.26	2.17	2.08
4.35	4.11	3.86	3.60	3.43	3.32	3.18	3.09	3.02	2.86	2.72
1.83	1.78	1.73	1.68	1.64	1.62	1.58	1.56	1.55	1.51	1.47
2.18	2.10	2.03	1.94	1.89	1.85	1.81	1.77	1.75	1.70	1.65
2.53	2.43	2.32	2.21	2.14	2.09	2.03	1.99	1.96	1.89	1.82
3.00	2.87	2.73	2.57	2.48	2.41	2.33	2.27	2.23	2.14	2.05
4.29	4.05	3.80	3.54	3.38	3.27	3.12	3.03	2.97	2.81	2.66
1.82	1.77	1.72	1.67	1.63	1.61	1.57	1.55	1.54	1.50	1.46
2.16	2.09	2.01	1.93	1.88	1.84	1.79	1.76	1.74	1.68	1.63
2.51	2.41	2.31	2.20	2.12	2.07	2.01	1.97	1.94	1.87	1.80
2.98	2.84	2.70	2.55	2.45	2.39	2.30	2.25	2.21	2.11	2.02
4.24	4.00	3.75	3.49	3.33	3.22	3.07	2.98	2.92	2.76	2.61
1.76	1.71	1.66	1.61	1.57	1.54	1.51	1.48	1.47	1.42	1.38
2.08	2.00	1.92	1.84	1.78	1.74	1.69	1.66	1.64	1.58	1.52
2.39	2.29	2.18	2.07	1.99	1.94	1.88	1.83	1.80	1.72	1.65
2.80	2.66	2.52	2.37	2.27	2.20	2.11	2.06	2.02	1.92	1.82
3.87	3.64	3.40	3.14	2.98	2.87	2.73	2.64	2.57	2.41	2.25
1.73	1.68	1.63	1.57	1.53	1.50	1.46	1.44	1.42	1.38	1.33
2.03	1.95	1.87	1.78	1.73	1.69	1.63	1.60	1.58	1.51	1.45
2.32	2.22	2.11	1.99	1.92	1.87	1.80	1.75	1.72	1.64	1.56
2.70	2.56	2.42	2.27	2.17	2.10	2.01	1.95	1.91	1.80	1.70
3.67	3.44	3.20	2.95	2.79	2.68	2.53	2.44	2.38	2.21	2.05
1.71	1.66	1.60	1.54	1.50	1.48	1.44	1.41	1.40	1.35	1.30
1.99	1.92	1.84	1.75	1.69	1.65	1.59	1.56	1.53	1.47	1.40
2.27	2.17	2.06	1.94	1.87	1.82	1.74	1.70	1.67	1.58	1.49
2.63	2.50	2.35	2.20	2.10	2.03	1.94	1.88	1.84	1.73	1.62
3.54	3.32	3.08	2.83	2.67	2.55	2.41	2.32	2.25	2.08	1.92
1.66	1.61	1.56	1.49	1.45	1.42	1.38	1.35	1.34	1.28	1.22
1.93	1.85	1.77	1.68	1.62	1.57	1.52	1.48	1.45	1.38	1.30
2.18	2.08	1.97	1.85	1.77	1.71	1.64	1.59	1.56	1.46	1.36
2.50	2.37	2.22	2.07	1.97	1.89	1.80	1.74	1.69	1.57	1.45
3.30	3.07	2.84	2.59	2.43	2.32	2.17	2.08	2.01	1.83	1.64
1.63	1.58	1.52	1.46	1.41	1.38	1.34	1.31	1.29	1.23	1.16
1.88	1.80	1.72	1.62	1.56	1.52	1.46	1.41	1.39	1.30	1.21
2.11	2.01	1.90	1.78	1.70	1.64	1.56	1.51	1.47	1.37	1.25
2.41	2.27	2.13	1.97	1.87	1.79	1.69	1.63	1.58	1.45	1.30
3.12	2.90	2.67	2.42	2.26	2.15	2.00	1.90	1.83	1.64	1.43
1.61	1.55	1.49	1.43	1.38	1.35	1.30	1.27	1.25	1.18	1.08
1.84	1.76	1.68	1.58	1.52	1.47	1.41	1.36	1.33	1.24	1.11
2.06	1.96	1.85	1.72	1.64	1.58	1.50	1.45	1.41	1.29	1.13
2.34	2.20	2.06	1.90	1.79	1.72	1.61	1.54	1.50	1.35	1.16
2.99	2.77	2.54	2.30	2.14	2.02	1.87	1.77	1.69	1.49	1.22