# User Authentication Incorporating Feature Level Data Fusion of Multiple Biometric Characteristics

Mark Abernethy

---

This thesis is presented for the degree of

Doctor of Philosophy

Murdoch University, January 2011.

---

# Declaration

I declare that this thesis is my own account of my research and contains as its main content work which has not previously been submitted for a degree at any tertiary education institution.

.....

Mark Abernethy

# Abstract

This PhD research project developed and evaluated innovative approaches to computer system user authentication, using biometric characteristics. It involved experiments with a significant number of participants and development of new approaches to biometric data representation and analysis.

The initial authentication procedure, that we all perform when we log onto a computer system, is considered to be the first line of protection for computer systems. The password is the most common verification token used in initial authentication procedures. Unfortunately, passwords are subject to numerous attack vectors (loss, theft, guessing or cracking), and as a result unauthorised persons may gain access to the verification token and be incorrectly authenticated. This has led to password-based authentication procedures being responsible for a large proportion of computer network security breaches.

In recent years, the use of biometrics has been increasingly researched as an alternative to passwords in the initial authentication procedure. Biometrics concerns the physical traits and behavioural characteristics that make each individual unique. Biometric authentication involves the use of biometric technologies in authentication systems, with the aim to provide accurate verification (based on biometric characteristics).

Research has demonstrated that uni-modal biometric authentication (that is, authentication based on a single biometric characteristic) makes it difficult for an impostor to impersonate a legitimate user. More recent research is finding that multi-modal biometric authentication (that is, authentication based on the combination of multiple biometric characteristics) can make it even more difficult for an impostor to impersonate a legitimate user. Thus multi-modal biometrics claims improved accuracy and robustness.

Multi-modal biometrics requires consideration of various aspects of data integration, known to the field of data fusion. Multi-modal biometric research has, until recently, focused on the fusion of data (from multiple sources) at the decision level or the confidence score level. It has been proposed that fusion of data at the feature level will produce more accurate and reliable verification.

However, fusion of data at the feature level is a more difficult task than fusion at the other two levels. For decision level fusion, 'accept' or 'reject' results from the different data sources are fused. For confidence score level fusion, confidence scores (typically in the continuous interval $[0, 1]$) from the different data sources are fused. That is, for the aforementioned levels, the data from the multiple sources are of the same nature. Feature level fusion combines feature vectors, where the data from the different sources are most likely to consist of different units of measurement.

Data fusion literature formally specifies that data may be combined according to three paradigms: competitive, complementary, and cooperative. Competitive data fusion assesses data from all available sources, and bases classification upon the 'best' source. Complementary data fusion combines all available data from all sources, and bases classification upon this combined data. Cooperative data fusion involves the selection of the best features of each individual data source, and then combines the selected features prior to classification.

The objectives of the current study were to investigate the use of two individual biometric characteristics (keystroke dynamics and fingerprint recognition). For keystroke dynamics, feature selection was employed to reduce the variability associated with data from this characteristic. For fingerprint recognition, a new method was developed to represent fingerprint features. This was done to assist classification by Artificial Neural Networks, and to meet the requirement to facilitate fusion with the keystroke dynamics data at the feature level.

Whilst feature level data fusion was the primary objective, investigation of the two individual characteristics was conducted to enable comparison of results with the data fusion results. For the data fusion investigation, the complementary and cooperative paradigms were adopted, with the cooperative approach involving four stages.

The feature selection method chosen to filter keystroke dynamics data was based on normality statistics, and returned results comparable to many other research efforts. The fingerprint feature representation method developed for this experiment demonstrated an innovative and effective technique, which could be applicable in a uni-modal or a multi-modal context.

As the new fingerprint representation method resulted in a standard length feature vector for each fingerprint, data alignment and subsequent feature level data fusion was efficiently and practicably facilitated.

The experiment recruited 90 participants to provide typing and fingerprint samples. Of these, 140 keystroke dynamics samples and 140 fingerprint samples (from each participant) were utilised for the first two phases of the experiment. Phase three of the experiment involved the fusion of the samples from the first two phases, and thus there were 140 combined samples. These quantities provided 100 samples for false negative testing and 10,500 samples for false positive testing (for each participant for each phase of the experiment). These figures are similar or better than virtually all previous research studies in this field.

The results of the three phases of the experiment were calculated as the two performance variables, the false rejection rate (FRR)—measuring the false negatives—and the false acceptance rate (FAR)—measuring the false positives.

The keystroke dynamics investigation returned an average FAR of 0.02766095 and an average FRR of 0.0862, which were at least comparable with other research in the field.

The fingerprint recognition investigation returned an average FAR of 0.0 and an average FRR of 0.0022, which were as good as (or better than) other research in the field.

The feature level data fusion adopting the complementary approach returned an average FAR of 0.0 and an average FRR of 0.0004. Feature level data fusion adopting the cooperative approach returned respective average FAR and FRR results of 0.00000381 and 0.0004 for stage 1, 0.0 and 0.0006 for stage 2, 0.0 and 0.001 for stage 3, and 0.0 and 0.001 for stage 4.

The research demonstrated that uni-modal biometric authentication systems provide an accurate alternative to traditional password-based authentication methods. Additionally, the keystroke dynamics investigation demonstrated that filtering 'noisy' data from raw data improved accuracy for this biometric characteristic (though other filtering methods than that used in this research may improve accuracy further). Also, the newly developed fingerprint representation method demonstrated excellent results, and indicated that its use for future research (in representing two dimensional data for classification by Artificial Neural Networks) could be advantageous.

The data fusion investigation demonstrated that multi-modal biometric authentication systems provide additional accuracy improvement (as well as a perceived robustness) compared to uni-modal biometric authentication systems. Feature level data fusion demonstrated improved accuracy compared with confidence score level and decision level data fusion methods. The new fingerprint representation method (which provided an innovative technique for representing data from any two dimensional data source) facilitated feature level data fusion with keystroke dynamic data, and the results validate the importance of using feature rich data.

# Contents

# List of Tables

# List of Figures

# Acknowledgments

I would like to express my sincere gratitude to my primary supervisor, Dr Andrew Turk, whose analytical skills proved most valuable and insightful during the course of this project. Dr Turk has been inspirational in encouraging me to strive for the highest standards, whilst undertaking rigorous and honest research. He has also been very helpful in regard to maintaining self-discipline (in relation to one's research work ethic), and a congenial and cooperative attitude with one's colleagues.

This research involved many technical aspects, and thanks are extended to my secondary supervisor Mr Shri M. Rai for his assistance with these matters. Shri Rai supervised my honours research and was therefore familiar with previous work in the area of keystroke dynamics. However, his proficiency in mathematics and science became truly valuable when working with fingerprint data and data fusion issues.

My thanks to friend and colleague, Dr Christian Payne. Christian first kindled my interest in computer security, and this area of research remains as interesting and challenging today as ever.

Thanks to Dr Lance Fung for suggesting the inclusion of fingerprint recognition for this project. His suggestion was inspirational and opened up new areas of investigation. Dr Fung also arranged for the purchase of the fingerprint scanner and the software development kit that were used for the experiment.

To my dear friend Dr R. (Chandra) Chandrashekhar of SwanLotus, thank you for your assistance with interpreting the point pattern matching algorithm used in the experiment to align fingerprint features. Your contribution was invaluable, and your friendship is priceless.

To my family and friends, thank you for the support that only loved ones can give.

To my dearest friend Paramahansa Yogananda, thank you so much for showing me a purpose to life and guiding me through its many, and varied, joys and tribulations.

# Chapter 1

# Introduction

## 1.1 Context Of The Study

The subject matter of this dissertation concerns the field of computer security. There are many definitions for computer security suggested in the literature; some are essentially theoretical, whilst others are more practical. Effectively, a computer system should be available to work correctly and reliably under all circumstances, and thus be resistant to abuse of any description. In practice, this may be impossible to achieve. Therefore, the goal should be to make a computer system as secure as it can practicably be made.

To achieve this goal, conscious decisions need to be made about what data and/or resources are to be protected and from whom (Schneier, 2000). Some questions to be posed are: What preventative measures can be taken? Will the security system concentrate on prevention only, or will detection and recovery also be considered? That is, if preventative measures fail, can the intrusion be detected and what will be done to stop the attack and recover from it?

The answers to these questions (and others) need to be defined when planning the security policy. The policy should be tailored to the specific security requirements of the system under consideration. Once determined, the security policy can be implemented using appropriate mechanisms.

Bruce Schneier (2000) has stated that computer security is a process, not a product. That is, security can not be purchased in a box. Commercially available

security systems may not suit a specified security policy, and trying to adapt one could be seriously detrimental to both product and policy. Also, it should always be assumed that there will be someone who will attempt to circumvent any security mechanism, so a security system (policies and mechanisms) must undergo continual revision to try to stay ahead of would be attackers. Therefore, it is important to develop processes that allow for this evolution.

Adopting an evolutionary approach is unlikely to solve all computer security problems, but should reduce the incidents of security breaches. Failure to do so will lead to inadequate security, which will undoubtedly leave a computer system vulnerable to attack.

Historically, there are three basic aspects upon which a computer security system rests (Bishop, 2003; Garfinkel et al., 2003):

1. Confidentiality: ensuring that data can only be viewed, wholly or in part, by those authorised to do so. Confidentiality therefore relates to the concealment of sensitive data, and concerns the ability to read or copy data.

2. Integrity: ensuring that data or resources cannot be modified by unauthorised persons. Those who are permitted to modify data or resources must only be able to do so according to their level of authorisation. Integrity therefore relates to the trustworthiness of data or resources, and involves ensuring the prevention of improper or unauthorised modification; it concerns the ability to write or delete data.

3. Availability: ensuring that a non-authorised user cannot prevent authorised users from access to the information or resources they require. This alludes to the maintenance of data or resources; protecting them from degradation.

In addition to the above points, Garfinkel et al. (2003) recommend the following three aspects for practical implementation:

1. Consistency: ensuring that the system behaves as expected by authorised users. User expectation is subjective; however administratively, consistency

means ensuring that the system performs correctly and reliably day after day. It is therefore correlative to the integrity and availability of data and resources.

2. Access control: ensuring that authorised persons are able to do what they are authorised to do, and that everyone else is not (Schneier, 2000). Access control enforces data confidentiality, integrity, and availability according to user authorisation levels, by restricting access to all data and resources to those authorised to access them. Mechanisms for implementing access control include access control lists, cryptography, and authentication.

3. Non-repudiation: preventing denial of abuse of the system. This involves providing undeniable proof that a user (legitimate or not) has performed an action they are not authorised to perform, and emphasizes the importance of logging and auditing.

Computer security has become a serious concern for companies, corporations, governments, academia, and private users. Though there may be numerous reasons for this concern, among them are the following:

- The seemingly limitless variety of tasks for which computer systems are being used, which inevitably open up more avenues of attack.

- The exponential rate of increase in the incidents of reported breaches to computer systems (CERT/CC, 2004). This is demonstrated by CERT/CC[1] survey figures for the number of reported breaches from 1988 to 2003 (refer Appendix A, Table A.1 and Figure A.1). It is reasonable to suspect that the number of security breaches is far greater than those reported, because companies and corporations are often reluctant to publicly acknowledge that they have security problems (CERT/CC, 2004).

- The number of reported vulnerabilities by which computer systems have been compromised (CERT/CC, 2004). This is demonstrated by CERT/CC survey figures for the number of reported vulnerabilities from 1995 to 2008 (refer

[1] CERT/CC is an acronym for Computer Emergency Response Team/Coordination Center. It is an organization that specialises in computer security and conducts surveys on network security.

Appendix A, Table A.2 and Figure A.2). Again, it is reasonable to suspect that there are an unknown number of vulnerabilities yet to be discovered and possibly exploited.

- Total company and corporate losses resulting from e-crime were estimated at $666 million for 2003 (CSO, 2004). In a recent survey, average losses due to security breaches was $234,244 per respondent (CSI, 2009)[2].

In the majority of cases, these problems result when computer systems are networked. A computer network is a collection of separate but interconnected computers, where interconnection means that computers share an agreed upon method of communication and information exchange (Tanenbaum, 1996). A standalone computer system is a single, separate computer that is not connected to other computers, and can only be accessed if one is in physical proximity to it. Networked computers, however, can be accessed remotely and an attacker can remain relatively anonymous.

Computer networks are complex systems which have several properties that impact on security (Schneier, 2000):

- They are complicated, with possibly thousands of components doing different tasks; any one of which could malfunction or be subverted.
- They are interactive; individual components work together with any combination of other components.
- They evolve. That is, they do things that they were not originally designed to do.
- They have bugs. That is, they misbehave in possibly unexplainable ways.
- They accept user input from a large number of people, sometimes all at the same time. This property coupled with the above properties introduces the seemingly endless possibility of vulnerability.

Given these properties and the ability for remote access, securing a computer network is extremely difficult (if at all possible). It is reasonable to conclude that

[2]The survey involved 443 respondents.

the larger and more complex the network is, the more vulnerable it will be to attack and the more intractable the task of securing it becomes.

The network that most of us are aware of is the Internet. The Internet is a worldwide collection of interconnected networks that cooperate with each other using an agreed upon protocol suite (Tanenbaum, 1996).

The popularity of the Internet has dramatically increased since its inception, particularly since the introduction of the World Wide Web in 1993. An NUA[3] survey reported that the number of Internet users in 2002 was 580,780,000 (Nua, 2002). Table A.3 and Figure A.3 (in Appendix A) present NUA survey figures from 1995 to 2002[4] to demonstrate the ever increasing number of Internet users.

As the Internet is a network of computer networks, it suffers from all the problems of any network. In fact, because the Internet is so large and complex, any individual network connected to the Internet is more at risk because there are so many users and points of access. CERT/CC (2004) stated that "given the widespread use of automated attack tools, attacks against Internet-connected systems have become so commonplace that counts of the number of incidents reported provide little information with regard to assessing the scope and impact of attacks"[5].

It is therefore imperative that security policies and mechanisms continue to develop and evolve. Schneier (2000) believes that authentication across computer networks is the most important security problem to be solved. Inadequate user authentication is one of the major causes of security breaches, and is thus responsible for many intrusions to computer systems. Authentication is the major focus of the current study.

The next section 1.2 considers the motivation and objectives for the current study by discussing authentication and the weaknesses of existent traditional methods.

[3]NUA Internet Surveys is an organization that conducts surveys of Internet usage.

[4]NUA no longer conduct surveys on the number of Internet users; the numbers are now so large that it seems the surveys provide little significant information.

[5]As of 2004, CERT/CC no longer publish the number of reported computer system breaches. Instead, they are working with others in the community to develop and report on more meaningful metrics.

## 1.2 Motivation And Objectives For This Research

### 1.2.1 Motivation For The Study

Authentication is required when it is necessary to know if a person is who they claim to be. It is a procedure that involves a person making a claim about their identity, and then providing evidence to prove it.

This study focuses on the initial authentication procedure that most computer users are accustomed to performing when they log onto a computer system. The initial authentication procedure is considered to be the first line of protection for computer systems (Garfinkel et al., 2003). It therefore stands to reason that this procedure should be made as accurate and reliable as feasibly possible.

Authentication is an access control mechanism that is based on user identity. It comprises two processes (Garfinkel et al., 2003):

1. Identification: the naming or labeling of an identity, providing the means to distinguish that identity from among a set of similar identities. For example on a computer system, legitimate users are given a unique username by which the system differentiates them from other legitimate users of the system.

2. Verification: the process of confirming the veracity of a claimed identity. For example on a computer system, a unique verification token (with direct correspondence to each username) is intended to verify the identity of a legitimate user. The verification process entails comparison of a stored or registered token, for a legitimate user, with a query token provided by the claimant during the authentication procedure.

Only when identity is established and confirmed—via the verification process—is authentication granted.

For the initial authentication procedure to be trusted, it must achieve **both** of the following goals (Schneier, 2000):

1. It must grant all authorised identities access to the system.

2. It must deny all unauthorised identities access to the system.

Traditional methods for the initial authentication procedure are based on (Umphress and Williams, 1985):

- What a user knows (i.e. knowledge of a verification token such as a password).
- What a user has (i.e. possession of a verification token such as a key or an ID card).

The password is by far the most common verification token used in initial authentication procedures (Joyce and Gupta, 1990; Garfinkel et al., 2003; Zhang et al., 2009), with many organisations widely adopting (and being heavily dependent upon) this method. This is because a password-based authentication procedure is cost effective to implement, making use of the existing infrastructure of the computer system. Also, the verification process involved is a simple comparison between character strings (more precisely, the hashed values of the passwords).

Unfortunately, passwords suffer from the following major weaknesses:

1. Users typically choose passwords that are low in entropy (i.e. randomness), because they are easier to remember; high entropy passwords are difficult for users to remember (Schneier, 2000). Passwords that are low in entropy become more predictable, and thus more guessable. Also, if there are an insufficient number of characters (of low entropy), the password may be easily cracked by brute force processing. Attackers using powerful computers and sophisticated cracking methods, might conduct offline attacks on stolen password files and crack weak passwords very quickly.

2. Even when users choose passwords that are higher in entropy, it is common for them to be written on paper and left in the vicinity of their computer desk. Though this makes it more convenient for users to remember high entropy passwords, it makes them vulnerable to loss or theft.

3. Very often, users will use the same password for accounts on different computer systems (Zhang et al., 2009). So if a legitimate user password on one system is compromised, an attacker may be able to use it to access other systems on which that user has an account.

In the CSI Computer Crime and Security Survey (2009), password related vulnerabilities rated $6^{th}$, of 7 major types of attack experienced by respondents, accounting for 17.3%. Other types of attacks were malware infections (64.3%), laptop theft (42.2%), and insider abuse (29.7%) (CSI, 2009). Though not the largest percentage, password related vulnerabilities remain a significant cause of security system breaches.

Other verification tokens such as keys and ID cards can also be easily lost or stolen. Therefore, the token-based authentication procedure continues to be one of the most widely exploited methods of compromising computer systems (Garfinkel et al., 2003). Once an attacker gains access to a computer system using a lost, stolen or cracked token, they will inevitably attempt to elevate their authorisation level. If successful, the confidentiality and integrity of data on the system is open to compromise.

As demonstrated in the CSI Computer Crime and Security Survey (2009), circumventing the initial authentication procedure is not the only way for an attacker to gain unauthorised access to a computer system. However, use of other methods to gain unauthorised access to computer systems is outside the scope of this discussion (refer section 1.4).

The limitations of the traditional initial authentication procedure mean that unauthorised persons may gain access to the verification token and be incorrectly authenticated. Therefore, there is no guarantee that knowledge or possession of the verification token truly confirms identity (Monrose and Rubin, 2000). So contrary to popular belief, initial authentication procedures do not actually verify identity; they only verify the holder of the token, who may or may not be the legitimate identity (Schneier, 2000).

Also of concern is that numerous computer system breaches are not discovered. Even if they are, it is often difficult to determine when the breach occurred or what data may have been compromised. Attackers may even leave a 'back-door' so that they can gain access to the system again at a later date (using legitimate users credentials).

To remedy the deficiencies associated with the traditional initial authentication procedure, a method that is being increasingly researched is based on 'what the user is' (Umphress and Williams, 1985). This method is referred to as biometrics, and may prove effective in improving upon the traditional authentication methods.

Biometrics are the physical traits and behavioural characteristics that are unique to each individual. Familiar examples of biometric characteristics are a person's signature, fingerprints, and DNA. Less familiar examples are retinal/iris patterns, voice/speech patterns, facial image patterns, hand geometry, and keystroke dynamics. A detailed discussion of biometrics is presented in Chapter 2 section 2.2.

Human beings use biometrics for person recognition every day. When we meet someone we know, we automatically assess features about the person. Firstly, we identify them according to our recollection of certain features, then we verify by closer scrutiny of those features. When we meet someone we have not met before, we memorise features about the person, so that when we next meet them we have information with which to make the comparison.

A computer based system using biometrics can only simulate, or model, human capabilities for person recognition, and may not attain the same accuracy as a human being does. The reason for this is that human beings have learned brain and memory functions, and can easily integrate information from multiple senses. Computers are limited in these capabilities.

In computer terms, the integration of information from multiple sources belongs to the field of study known as data fusion. There are three paradigms and three major level of fusion available for the integration of data from multiple sources. The issues surrounding data fusion, including the paradigms and fusion levels, are discussed in detail in Chapter 2 section 2.3.

In relation to biometrics, incorporating multiple sources of data for authentication purposes is known as multi-modal biometric authentication. Until recently, most research in multi-modal biometrics have concentrated on combining data at the decision or confidence score levels (refer Chapter 2 section 2.3.2).

The intention of this study is to fuse multi-modal biometric data at the feature level. It has been proposed in the literature that feature level data is richer in meaningful information, and should provide a more accurate and robust verification process. However, there are certain operational issues associated with this level of fusion that require careful consideration and further research.

This study attempts to offer some solutions to some of these operational issues, in order to improve the accuracy and robustness of both uni-modal and mutli-modal biometric authentication. It must be stated though, that there may be other viable methods for dealing with these issues.

### 1.2.2 Objectives of the Study

As traditional authentication methods are responsible for a significant proportion of computer security breaches, this study attempts to improve the initial authentication procedure by utilising biometrics in the verification process.

Three research objectives were identified:

1. To investigate the effective use of the biometric characteristic keystroke dynamics (refer Chapter 3 and Chapter 5 section 5.4), and to assess the results in comparison to previous research in this field. To achieve this objective, raw data are pre-processed to reduce the high degree of variability associated with this biometric characteristic.

2. To investigate the effective use of the biometric characteristic fingerprint recognition (refer Chapter 4 and Chapter 5 section 5.5), and to assess the results in comparison to previous research in this field. To achieve this objective (using Artificial Neural Networks for classification; refer section 1.5), and also to facilitate the third objective of the experiment (feature level data fusion), a new representation method for fingerprint features was developed to attain feature vectors of a standard length for all fingerprint samples for all participants.

3. To investigate the feature level fusion of data from the previously stated two sources (refer Chapter 2 section 2.3 and Chapter 5 section 5.6), and to assess the results in comparison to previous research in this field. To achieve this objective, data alignment of the two data sets was necessary, as they have different units of measurement. A simple data alignment method was applied to the fingerprint data, to bring them into alignment with the keystroke dynamics data.

These three objectives became the three phases of the experiment, as discussed in section 1.5.

### 1.2.3 Research Questions

The research objectives outlined in section 1.2.2 give rise to the following research questions:

1. Can the biometric characteristic keystroke dynamics (using feature selection to remove noisy data) demonstrate enough accuracy—in comparison to previous research in this field—to be considered an effective verification token in a uni-modal biometric authentication system?

2. Can the biometric characteristic fingerprint recognition (using a new feature representation method to provide standard length feature vectors) demonstrate enough accuracy—in comparison to previous research in this field—to be considered an effective verification token in a uni-modal biometric authentication system?

3. Can feature level data fusion (using the feature vectors gained from points 1 and 2) demonstrate enough accuracy—in comparison to previous research in this field—to be considered a better alternative in a multi-modal biometric authentication system compared to such a system where decision or confidence score level fusion is utilised.

The next section 1.3 discusses the significance or relevance of the research.

## 1.3 Significance Of The Research

This study is one of many into biometrics for authentication. The many biometric characteristics achieve varying degrees of accuracy because of variations in instrument accuracy, raw data format, data pre-processing, software accuracy, and the uniqueness (between different people) of the biometric characteristic under consideration. Though some standards for data collection methods and verification processes have been introduced in recent years[6], these efforts are taking some time to filter into the biometric community (research and commercial).

As discussed in Chapter 2 section 2.2, biometric verification systems rarely indicate that two samples, taken from the same person at different times, are a perfect match. Instead the system only indicates the probability that two samples are from the same person; that is, there is seldom absolute certainty. When making the final verification decision, it is crucial that the biometric verification system attains the most accurate probability score possible. This is particularly true when multiple characteristics are being used.

Early studies involving multiple characteristics focused on fusing data at the decision or confidence score levels. More recent studies have attempted data fusion at the feature level. The concepts of decision, confidence score and feature level fusion are discussed in Chapter 2 section 2.3.2.1.

Fusion at the feature level uses data closest to the raw data, and therefore is richer in feature information (compared with the other levels). To attain the best probability score, and thus take full advantage of the uniqueness of the biometric characteristics, it stands to reason that utilising the richness of these features would be advantageous. The experiment attempts to demonstrate this, and to develop a methodology which is efficient and scalable.

At the commencement of the current study, relatively little work had been done on data fusion at the feature level. Fusion of data at this level requires combining data from different biometric characteristics in a way that the individual features of

[6]Refer to The Biometrics Resource Center Website sponsored by the National Institute of Standards and Technology (NIST). Available at:
http://www.itl.nist.gov/div893/biometrics/standards.html

each characteristic are truly represented. From an operational perspective, this poses difficulties in relation to the additional processing required because of differences in data format between the characteristics.

The significance of the current investigation to the field of biometrics generally, and multi-modal biometrics specifically, is as follows:

- It investigates two biometric characteristics (keystroke dynamics and fingerprint recognition) independently of each other, to demonstrate that they can provide an accurate alternative to traditional authentication methods. In order to rigorously investigate these two biometric characteristics, the following issues need attention:

    1. A feature selection method should be employed to filter 'noise' from the collected data, because of the known high variability of keystroke dynamics raw data.

    2. To facilitate classification of fingerprint feature data by Artificial Neural Networks, a new fingerprint representation method needs development; this would also facilitate fusion of this data with keystroke dynamics data.

- It investigates multi-modal biometric authentication to demonstrate that it provides additional accuracy improvements as well as a perceived robustness to the verification process.

- It investigates feature level data fusion to demonstrate that fusion at this level offers improvement in accuracy over confidence score level and decision level data fusion, and thus the importance of using feature rich data.

What makes this research of particular significance is the fusion of data at the feature level. As discussed in Chapter 2 section 2.3.2.2, most research into multi-modal biometrics has concentrated on performing data fusion at the confidence score level, where valuable feature data is lost in the processing. Even though fusion at the confidence score level provides better accuracy and robustness than uni-modal biometric systems, the current study attempts to demonstrate that even better accuracy gains can be achieved by fusion of data at the feature level.

The methodology proposed in the current study should be generically applicable to any number (and any type) of biometric characteristics, provided that the pre-processing of raw data—to obtain the feature data—maintains data integrity.

## 1.4 Scope Of The Research

The scope of the research is restricted to biometric authentication. That is, to improve the initial authentication procedure and thus reduce the probability of successful attack via this mechanism. Other possible authentication procedures are not investigated.

The choice of the two biometric characteristics used in the experiment was made for demonstration of the rationale behind the experiment, and for convenience of implementation. As discussed in Chapter 2 section 2.2 there are many biometric characteristics available for this type of experiment. The results of this experiment are claimed only for those characteristics implemented, though it could reasonably be expected that other biometric characteristics may be used for the same purpose (with the accuracy attained dependent on the actual characteristics chosen).

Feature level data fusion was investigated because researchers in this field have long suspected that using feature level data would improve both accuracy and robustness. Again, the results of this experiment are claimed only for feature level data fusion of the two biometric characteristics utilised, though it could reasonably be expected that other biometric characteristics may be used for the same purpose (with the accuracy attained dependent on the actual characteristics chosen).

As mentioned previously, there exist other avenues of attack for those attempting to gain unauthorised access to computer systems. Some attack possibilities may be due to:

- Exploitable vulnerabilities. As mentioned in section 1.1, vulnerabilities exist on most computers and computer networks. In the case of computer networks, these vulnerabilities are exploitable remotely and with a degree of anonymity. As evident from the seemingly endless number posted on vulnerability websites[7], these vulnerabilities affect computers with different operating systems

[7]Such as: US-CERT available at http://www.kb.cert.org/vuls/ and Security Focus available at http://www.securityfocus.com/vulnerabilities

and many different components and applications. Very often patches are available to fix vulnerabilities, but users or system administrators are often not diligent in applying those patches.

- Human fallibility:

  - Social engineering remains one of the most prevalent means of information leakage. The celebrated case involving the "hacker" Kevin Mitnick provides a very good example of social engineering. Mr Mitnick attributes the majority of his success to social engineering skills rather than computer "hacking" skills (Mitnick and Simon, 2002). For example, he was able to obtain passwords and other secret information by impersonating someone else and just asking for it (Schneier, 2000).

  - Careless employees. Often employees do not realise the sensitivity of the data or resources that they are working with each day. Many have a lax attitude toward computer security because they are unaware of the importance of their actions in relation to the protection of their employers physical and intellectual property. Also, middle and higher level managerial staff may not be as careful as they should be with technological secrets, and leak important information to unauthorised employees in order to 'get the job done'. In a recent survey, 51% of respondents attributed security events to internal sources (CSO, 2010).

  - Disgruntled or mischievous employees (either current or past) can often be in possession of information about an employers sensitive property. If they believe they have cause, they could either attempt disruption themselves or pass on information to others who might have a vested interest in doing so.

The experiment performed for this dissertation makes no attempt to solve or discuss these issues beyond that already mentioned.

The next section summarises the research method used for the experiment.

## 1.5 Experimental Method And The Rationale For Its Selection

The research began with a thorough literature review to identify the most appropriate way to carry out a set of practical experiments testing biometric authentication approaches, with a relatively large number of participants.

The methodology adopted for all three phases of the experiment was strongly influenced by the intention to utilise Artificial Neural Networks (ANNs) for pattern recognition. ANNs and their suitability for pattern recognition are discussed in Chapter 2 sections 2.4.2 and 2.4.3 respectively. ANNs require input vectors of a consistent length, so for each phase feature vectors were required to be of a consistent length.

With this requirement in mind, feature selection was applied to the keystroke dynamics raw data to obtain feature vectors of the same length. Feature selection was employed to reduce the effects of the known variability that exists in keystroke dynamics raw data. Therefore, all keystroke dynamics feature vectors (for all samples for all participants) were processed to a length of 24 elements. The process is described in detail in Chapter 5 section 5.4.

For the fingerprint recognition phase, a new representation method for fingerprint features was developed to meet the requirement for consistent length feature vectors. This new representation method was also required to facilitate the data fusion phase, where the combined feature vectors (composed of the feature vectors from the previous 2 phases) needed to be of the same length.

The new representation method involved identifying (for each participant) 8 local fingerprint features common to each of their samples. Six attributes for each of the identified 8 local fingerprint features were recorded, two of which specified their location (i.e. the x and y coordinates in a two dimensional plane). Therefore, all fingerprint feature vectors (for all samples for all participants) were processed to a length of 52 elements. The process is described in detail in Chapter 5 section 5.5.

By simple normalisation, data alignment was achieved between the keystroke dynamics feature vectors and the fingerprint feature vectors. The combined feature vectors were then used in the data fusion phase of the experiment. Two approaches for the data fusion phase were investigated: the complementary paradigm and the cooperative paradigm.

The complementary approach utilised 100% of data in the combined feature vectors (i.e. all of the features from both biometrics). Therefore, all data fusion feature vectors (for all samples for all participants using the complementary paradigm) were processed to a length of 76 elements. The process is described in detail in Chapter 5 section 5.6.2.

The cooperative approach utilised the 'best' features from the combined feature vectors (i.e. selected features from both biometrics). This raises three questions (keeping in mind the requirement of a consistent length for the final combined feature vector):

1. What percentage of the 100% of data in the combined feature vector should be used?
2. What proportion of features from the original feature vectors of each biometric characteristic should be used?
3. What feature selection method should be used?

The processes involved in developing solutions to these questions are described in detail in Chapter 5 section 5.6.3.

The experiments were designed on this basis and successfully carried out using 90 participants (see Chapters 5 and 6), producing very satisfactory results (see Chapters 7 and 8).

## 1.6 Outline Of This Dissertation

The next chapter provides a review of three subject areas directly associated with the experiment conducted for this dissertation. Biometrics is discussed in Chapter 2 section 2.2. Data fusion and multi-modal biometrics are discussed in Chapter 2 section 2.3 (including a review of literature related to multi-modal biometrics in

section 2.3.2.2). Pattern recognition and Artificial Neural Networks (ANNs) are discussed in Chapter 2 section 2.4.

Chapter 3 provides a detailed discussion of the biometric characteristic keystroke dynamics (including a review of literature related to keystroke dynamics in section 3.4). Chapter 4 provides a detailed discussion of the biometric characteristic fingerprint recognition (including a review of literature related to fingerprint verification in section 4.5).

Chapter 5 describes in detail the research methodology and the implementation of the three phases of the experiment. Section 5.4 discusses the keystroke dynamics phase, section 5.5 discusses the fingerprint recognition phase, and section 5.6 discusses the feature level data fusion phase.

In Chapter 6 the method used to classify the outputs from ANN testing is discussed, and the results are then presented in section 6.4. Chapter 7 then discusses these results in depth, comparing each phase with the other and with other research efforts in their respective fields.

Chapter 8 provides a conclusion to the dissertation, including discussion of the key findings, limitations of the research, and potential future research directions.

## 1.7 Conclusion

In this chapter, some important aspects of computer security were highlighted in section 1.1. One of these aspects was authentication, and improving the initial authentication procedure was the overall focus of this dissertation.

Section 1.2.1 introduced the motivation for the research, which is related to problems associated with the traditional authentication procedure. Biometrics was introduced as offering a possible alternative to the traditional authentication procedure. The objectives of the study were then outlined in section 1.2.2, and the research questions were posed in section 1.2.3.

The significance or relevance of the study was discussed in section 1.3. Section 1.4 discussed the scope of the research, and section 1.5 summarised the research method.

Finally, section 1.6 provided a outline of the remainder of the dissertation.

# Chapter 2

# Background

## 2.1 Introduction

A number of associated areas of study impact on, or are utilised in, the work described in this thesis. These areas include biometrics, data fusion and multi-modal biometrics, and pattern recognition and Artificial Neural Networks (ANNs). This chapter discusses these areas of study, and provides background on them to help understand why certain choices were made during the experimental stage of the study.

Consequently, background will be given on the following topics:

- Biometrics: the personal characteristics used for authentication (section 2.2). This section discusses the following topics:
  - An overview of biometrics (section 2.2.1), where definitions for biometrics and biometric technologies are given.
  - The components of a biometric authentication system; what a biometric authentication system is intended to achieve; the requirements that it should meet; and how it operates (section 2.2.2).
  - Biometric system errors and performance variables used to present experimental results (section 2.2.3).
  - A description of well known biometric characteristics for possible use in a biometric authentication system is also given in section 2.2.4.

- Data Fusion and Multi-Modal Biometrics (section 2.3):
  - Data Fusion: the integration or merging of data from multiple sources (section 2.3.1).
  - Multi-Modal Biometrics: the use of multiple personal characteristics used for authentication (section 2.3.2).
- Pattern Recognition and Artificial Neural Networks (section 2.4):
  - Pattern Recognition: techniques for classifying data based on extracted information (section 2.4.1).
  - Artificial Neural Networks: a particular pattern recognition technique used in the experiment of the current study (section 2.4.2).

## 2.2 Biometrics

### 2.2.1 Overview of Biometrics

According to Webster's dictionary (Websters, 2010a), there are two definitions for biometrics. In biology, biometrics refers to the science and technology of measuring and statistically analysing biological data. In information technology, biometrics refers to measuring and analysing human body characteristics for authentication purposes. In this latter context, biometrics is often termed biometric authentication.

Refining this definition, biometric authentication concerns using the physical traits and behavioural characteristics that make each individual unique for authentication purposes, and encompasses any personal characteristic that can be used to uniquely verify a person's identity (Monrose and Rubin, 2000).

The distinction between establishing a person's true identity and verifying a person's claimed identity needs clarification. Establishing a person's true identity (i.e. lawful name) is the concern of law enforcement. For example, if a person assumes three identities to defraud the social security department, the department would strive to discover this, determine the person's real or lawful name, and take the appropriate legal action.

In information technology, verifying a person's claimed identity is the aim of the initial authentication procedure (Wayman et al., 2005). Traditionally, this procedure requires all legitimate users of a system to have possession or knowledge of a verification token. The token (for example, username and password) is presented to the system at the time of attempted authentication (that is, logon), and the system makes a comparison between the supplied token and a registered token for that user. If they match, the user is authenticated; if not, access is denied.

In order to use biometrics for authentication, biometric technologies have evolved to incorporate automated methods (usually performed by a computer) and the technical apparatus used to verify the identity of a person based on physiological and/or behavioural characteristics (Wayman et al., 2005).

Biometrics technologies involve (Liu and Silverman, 2001):

- The capture of biological data from an individual.
- The extraction of uniquely identifiable features from that data.
- The processing of these features into a format that can be stored and later retrieved. This format is referred to as the 'registered template' or 'reference template' for that individual.
- The comparison or matching of the registered template with a template processed from a sample provided at the time of authentication (referred to as a 'query sample').

Because biometric technologies are automated, when they are deployed into the authentication procedure it is referred to as a 'biometric authentication system'. These systems can be based on one or multiple biometric characteristics. When one biometric characteristic is employed, it is referred to as a 'uni-modal biometric authentication system' or more simply 'uni-modal biometric system'. Often the term is shortened further to 'uni-modal biometrics'. When multiple biometric characteristics are employed, it is referred to as a 'multi-modal biometric authentication system' or more simply a 'multi-modal biometric system' or 'multi-modal biometrics'.

Uni-modal biometrics based on a single personal characteristic normally make it difficult for an impostor to impersonate a legitimate user. Logically, multi-modal biometrics should make it even more difficult for an impostor to impersonate a legitimate user, as it requires impersonation of two or more characteristics.

The next section discusses biometric authentication systems.

### 2.2.2 Biometric Authentication Systems

A biometric authentication system is intended to authenticate people based on certain biometric characteristics. So, biometric authentication systems essentially operate as a pattern recognition system (Jain et al., 2004). That is, features of a biometric characteristic determine a distinct pattern for each person. Authentication is granted or denied on the basis of recognition of this pattern, in the data supplied at the authentication stage.

It should be noted, that there are two broad categories of biometric characteristics (Revett et al., 2007):

1. The physiological category pertains to physical attributes or traits that allow for the measurement of a person's physiological features. Examples of such biometric characteristics are local fingerprint configuration, retinal blood vessel pattern, and iris pattern. The measurements of the physical attributes are used to formulate quantifiable feature vectors.

2. The behavioural category pertains to a person's behavioural attributes or traits, that again require measurement and quantification. Examples of behavioural biometric characteristics are speech pattern, signature, and typing pattern.

When implemented into a biometric authentication system, the physiological biometric characteristics are generally considered to be much more accurate, reliable, and robust in comparison to the behavioural biometric characteristics (Revett et al., 2007). There is good evidence for this perception, as will be discussed in section 2.2.4. However, physical biometric characteristics have the following disadvantages (Revett et al., 2007):

- Measuring devices for physical biometric characteristics are subject to electromagnetic interference (noise), and natural wear and tear. This could reduce the efficiency and accuracy of their use.

- The measuring devices, and the software to interact with them, are typically costly to deploy. This makes it difficult to utilise physiological biometric characteristics for general remote transactions (for example, in web based applications such as online purchasing or banking).

- Some physiological biometric characteristics would be considered by users as intrusive. For example, to use iris and retinal patterns, photographing or scanning the eye in very close proximity is required, and users may find this uncomfortable and inconvenient.

Though the behavioural biometric characteristics have an advantage in that they do not suffer from the limitations associated with the physiological biometric characteristics, they do suffer from the following disadvantages (Revett et al., 2007):

- Their attributes or traits are more difficult to precisely measure and quantify than the physiological biometric characteristics. Thus they typically demonstrate much more variability.

- Their attributes or traits are not as enduring. That is, they are more subject to change over short periods of time, for example because of attitude or tiredness.

- Because of the two previous points, behavioural biometric characteristics are considered less accurate, reliable, and robust in comparison to the physiological biometric characteristics.

So both categories of biometric characteristics have their advantages and disadvantages, and the selection of which characteristic to utilise in a biometric authentication system will be dependent upon the intended application. Therefore, it is possible for any characteristic from either category to be effectively deployed, provided appropriate procedures are put in place to minimise the effects of any disadvantages, and thus ensure the security of the system.

A generic uni-modal biometric authentication system (illustrated in Figure 2.1) consists of four modules (Ross and Jain, 2004; Faundez-Zanuy, 2009):

1. The sensor or biometric capture module. This is the module responsible for capturing the physical or behavioural traits of the particular biometric characteristic under observation.

2. The feature extraction module. This module processes the raw captured data by extracting the features that represent the traits of the biometric characteristic. The extraction of features forms the feature vector that is output by this module.

3. The matching module. This module utilises a classifier to ascertain if the extracted features, of the biometric characteristic under observation, match those of the registered template in the database. The module typically outputs a match confidence score, being the likelihood or probability that the two samples match.

4. The decision module. This module uses the confidence score to determine the final classification decision. As biometric samples rarely match exactly, a threshold is typically applied in this module to make the final decision.

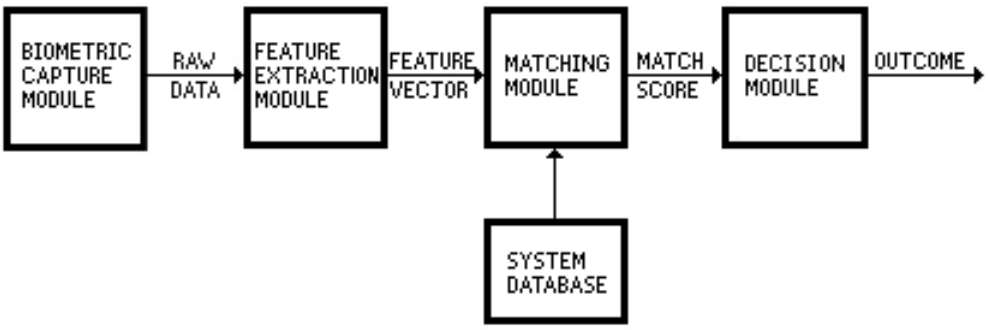


Figure 2.1: The Generic Biometric Authentication System
The image was sourced from Faundez-Zanuy, 2009.

To incorporate biometrics into an authentication system, the measurement of any biometric characteristic must meet the following requirements (Matyas Jr and Riha, 2000; Jain et al., 2004):

- Universality: each person (in the target user group) should have the characteristic. Some characteristics may be encumbered by physical disabilities, such as muteness or eye disease.

- Uniqueness or Distinctiveness: any two persons should be sufficiently different in terms of the characteristic. For example, identical twins each have unique fingerprints, yet they cannot be distinguished by their DNA.

- Permanence: the characteristic should be sufficiently invariant (with respect to the matching criterion) over a relevant period of time. For example, iris features remain stable for decades, whereas facial features change significantly with age and fingerprint features can be degraded by scaring, abrasion or prolonged use of cleaning chemicals.

- Collectability: obtaining the characteristics should be easy, and they must be quantitatively measurable. For example, retinal scanning requires the use of expensive apparatus and the collection process requires precise positioning of the eye. This makes collectability cumbersome, whereas keystroke dynamics only requires that the user has the ability to type (even with only one finger) on a standard computer keyboard.

The following are also required for a practical implementation of an authentication system:

- Performance: the system must have the resources needed to attain an achievable recognition accuracy and speed for the intended number of user profiles. The system must also compensate for operational and environmental factors that may affect accuracy and speed.

- Acceptability: the extent to which people are willing to accept the use of a biometric characteristic (for authentication purposes) in their daily lives. Facial recognition is considered an acceptable biometric because people are accustomed to having their photographs taken, and the process in non-intrusive.

  However, retinal scanning requires the eye to be held in a constant position while an infrared laser beam is directed through the cornea of the eye.

Most people would find this highly intrusive, and others would not want laser beams fired into their eyes.

- Circumvention: the system must be sufficiently robust to withstand various fraudulent methods of attack. From a security perspective, low circumvention would be considered highly desirable.

A biometric authentication system typically incorporates two phases (Jain et al., 2004):

- Enrolment phase: This phase involves the capture, feature extraction, and formation and storage of a registered template for each legitimate user of the system. Quality checks must be performed during the enrolment phase, to ensure that the acquired samples can be accurately and reliably processed.

- Validation phase: This phase has two modes:

  - Verification mode: the system validates a person's claimed identity, by comparing their pre-stored registered template with a template derived from a sample provided by the claimant at the time of attempted authentication (termed a 'query' or 'test' sample). The process of obtaining the query template must be the same as that employed in the formation of the registered template (during the enrolment phase). Verification is a one-to-one operation, aimed at preventing multiple people from using the same identity.

  - Identification mode: the system identifies an individual by searching a database of registered templates, and comparing their query template with all templates in the database. Identification is a one-to-many operation, aimed at preventing a single person from using multiple identities.

The next section discusses the performance variables used to measure—and present results—of a biometric authentication system. Also discussed, are the common system errors associated with a biometric authentication system.

### 2.2.3 Biometric Performance Variables and System Errors

In an empirical study where classification is being assessed (in the case of biometrics, verification accuracy), there are four possible outcomes[1] (Bradley, 1997; Fawcett, 2006; Flach, 2004):

1. True positive. The sample of a valid user is correctly accepted as belonging to that user.

2. False positive. The sample of a non-valid user, is incorrectly accepted as belonging to a valid user. This is referred to as a Type I error.

3. True negative. The sample of a non-valid user, is correctly rejected as not belonging to a valid user.

4. False negative. The sample of a valid user, is incorrectly rejected as not belonging to that user. This is referred to as a Type II error.

In experiments related to authentication, it is the error in classification that is of interest. Therefore, classification outcomes 2 and 4 above (i.e. Type I and II errors) are measured by the following rates:

- The rate at which non-valid users are falsely accepted as valid users. That is, false positives or Type I errors.

- The rate at which valid users are falsely rejected. That is, false negatives or Type II errors.

The two performance variables used to express these rates are generally termed the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) respectively[2]. Thus, the FAR is expressed as the ratio of samples from non-valid users that are falsely accepted, and the FRR is expressed as the ratio of samples from valid users that are falsely rejected.

[1] These classification outcomes are discussed in greater detail in Chapter 6 section 6.2.1.

[2] In some literature, these are termed the False Match Rate (FMR) and the False Non-Match Rate (FNMR) respectively.

In a typical authentication system, the primary concern is to minimise access to the system by non-valid users. The degree of minimisation will be dependent on the nature of the information being protected. For example, the military may have top secret information to be protected, and would insist on eliminating any unauthorised access. However, in achieving zero tolerance access, valid users may be inconvenienced by being falsely rejected numerous times.

In the case of a top secret system, this may be considered an acceptable trade-off. In contrast, a home computer used by family members would not necessarily require the same level of restriction, and numerous false rejections may be considered an unreasonable inconvenience.

In an experiment, the ultimate goal when evaluating verification accuracy is to achieve a FAR of 0%. Such a rate means that no non-valid user has been accepted as a valid user, and indicates that the experiment performed to the highest expectations. The goal for the FRR is to achieve a rate that is appropriately low without negatively impacting on the FAR. Previous research has shown that a FRR of 0% is difficult to attain without having some detrimental effect on the FAR (Maltoni et al., 2003; Qi and Wang, 2005).

As an example, a FAR of 0.4% and a FRR of 5% indicates that four in one thousand non-valid users could expect to be successful in gaining unauthorised access, while a valid user could expect to be rejected once in twenty attempts. Reducing the FRR to 1% would mean a valid user could expect to be rejected only once in one hundred attempts. However, this may, for instance, increase the FAR to about 2.5%, which means that twenty five in one thoudand non-valid users could expect to be successful in gaining unauthorised access. This would not generally be considered a wise trade-off.

In practice, no biometric authentication system can be expected to absolutely verify the identity of an individual (Matyas Jr and Riha, 2000). For example, a password system involves the comparison of the hashes of two passwords (one being the query sample and one being the registered template in a database). If there is an exact correspondence, verification is confirmed.

However, a biometric system can only indicate the likelihood or probability that two samples are from the same person. This is because biometric characteristics are determined by sensors, and there are various factors associated with human interaction with sensors that affect the accuracy of sensor readings. This means that two biometrics samples from the same person are most unlikely to be absolutely identical.

Some of the error rates that reflect or impact on this uncertainty are (Maltoni et al., 2003; Nandakumar, 2008):

- Failure To Capture Rate (FTCR): the percentage of times that a biometric capture device fails to automatically capture the intended biometric trait. This usually occurs as a result of poor quality or malfunctioning sensing devices.

- Failure To Enrol Rate (FTER): the percentage of times that users of the biometric authentication system are unable to enrol in the system. This may occur as a result of quality control checks on the enrolment procedure, a poor quality or malfunctioning sensor, inappropriate interaction between the user and the sensor, or other environmental factors (such as ambient conditions, background noise, etc.).

- Equal-Error Rate (EER): denotes the (classification) error rate—at a given threshold $t$—where the FAR and the FRR are equivalent. Though this may seem to be an ideal trade-off point (that is, an appropriate point of equal accuracy between the performance metrics), for authentication purposes this is seldom the case. Most often the threshold requires adjustment to provide a more stringent control over the FAR.

The next section provides an overview of the different biometric characteristics that are available for use in a biometric authentication system.

### 2.2.4 Biometric Characteristics

Any discussion of the validity and utility of different biometric characteristics should be carried out in relation to the requirements of any particular biometric authentication system (refer section 2.2.2). Table 2.1 summarises the common candidate biometric characteristics and nominates the usual levels of achievement (high, medium, or low) of the generic system requirements (discussed in section 2.2.2), for each characteristic. In Table 2.1, H represents a high rating, M represents a medium rating, and L represents a low rating.

| Biometric Identifier | Universality | Distinctiveness | Permanence | Collectability | Performance | Acceptability | Circumvention | Rating |
|---|---|---|---|---|---|---|---|---|
| DNA | H | H | H | L | H | L | L | 0.81 |
| Facial Recognition | H | L | M | H | L | H | H | 0.67 |
| Iris Pattern Recognition | H | H | H | M | H | L | L | 0.86 |
| Retinal Pattern Recognition | H | H | H | L | H | L | L | 0.81 |
| Speaker Recognition | M | L | L | M | L | H | H | 0.52 |
| Fingerprint Recognition | M | H | H | M | H | M | M | 0.81 |
| Palmprint Recognition | M | H | H | M | H | M | M | 0.81 |
| Hand Geometry | M | M | M | H | M | M | M | 0.71 |
| Keystroke Dynamics | M | M | L | H | M | M | M | 0.71 |
| Signature Recognition | M | L | L | H | L | H | H | 0.57 |
| Gait Recognition | M | L | L | H | L | H | M | 0.62 |
| Body Odor Recognition | H | H | H | L | L | M | L | 0.71 |

Table 2.1: Summary of Utility Biometric Characteristics for Authentication Systems

In most cases, the achievement levels presented for the candidate biometric characteristics are based on those proposed by Jain et al. (2004), although the following slight modifications have been made to some of their proposed achievement levels.

In the opinion of the author the universality achievement levels offered by Jain et al. (2004) in relation to keystroke dynamics and signature recognition are unfairly represented. As an example, it is true that the ability to type or write a signature would definitely be affected by injury to, or loss of, a finger. However, it is just as true that fingerprint recognition would be equally affected by such an occurrence. In both cases, this should not permanently affect a person's ability to perform the

necessary task involved. Therefore, the levels allocated to keystroke dynamics and signature recognition (for universality), in Table 2.1, have been modified to 'medium' rather than the 'low' value allocated by Jain et al. (2004).

Also in the opinion of the author, the 'low' level allocated to keystroke dynamics for distinctiveness and performance, by Jain et al. (2004), seems unfairly representative. The literature review in Chapter 3 section 3.4 demonstrates that this biometric characteristic is distinctive enough—and performs well enough if data is carefully pre-treated—to be allocated a higher level of achievement. Accordingly, a 'medium' level has been allocated to these system requirements for keystroke dynamics.

Collectability of keystroke dynamics should in the author's opinion be allocated a 'high' value, as a keyboard is a standard peripheral device on most computers and computer users would perform the majority of their interaction with a computer via a keyboard.

A rating column has been included in Table 2.1, to help assess the overall viability of each characteristic. The rating score for each characteristic was calculated as follows:

- A High rating was assigned a value of 3, Medium a value of 2, and Low a value of 1.
- The sum of assigned column values, for each biometric characteristic, was divided by the highest possible score of 21.

An exception to this scoring scheme was that applied to 'Circumvention'. As a low degree of circumvention is desirable, the complementary values are assigned to Low and High. That is, Low was assigned a value of 3, High a value of 1.

It is important to note that although the rating is provided to 2 decimal places (to permit close alternative methods to be distinguished), it is only approximate as the rating scale is crude and there has been no attempt to weight the criteria.

A description of each of the biometric characteristics identified in Table 2.1 follows, except for Keystroke Dynamics and Fingerprint Recognition which are only discussed briefly. These two characteristics were utilised in the current study, and consequently are discussed in detail in Chapters 3 and 4.

The first characteristic for discussion is provided as a counter example, because it rates highly in the requirements for a biometric authentication system, but to date has not (to the author's knowledge) been used in such a system.

#### 2.2.4.1 Deoxyribonucleic Acid (DNA)

According to Webster's dictionary (Websters, 2010b), DNA is a "complex molecule found in the chromosomes of almost all organisms, which acts as the primary genetic material; the part of the cell nucleus that is the repository of hereditary characteristics." That is, it contains the genetic instructions for the development and function of almost all living organisms. In animals and plants these genetic instructions are present in every cell nucleus.

DNA is unique to each individual and is therefore a unique identifier. However, DNA has mostly been used in forensic science and biological research, because the following limitations have restricted the use of DNA for biometric authentication systems (Jain et al., 2004):

1. A persons DNA can be unsuspectingly obtained. For example, from a drink can or glass. Another person could then use this DNA sample for false authentication.

2. Information about a person's genetic information could be abused. For example, if a prospective employee has a hereditary condition, an employer who gains knowledge of this may use that knowledge to discriminate against the person in their employment opportunity.

3. DNA is currently unsuitable for real-time applications because of the processing (chemical and machine) and time involved to match samples.

4. The human factor in processing opens up the possibility of sample contamination or degradation that could impact on the accuracy of the matching process.

DNA technology also differs from standard biometric technologies in several ways (International Biometric Group, 2006):

- It requires a tangible physical sample as opposed to an impression, image, or recording.
- The matching process is not done in real-time, and currently not all stages of comparison are automated.
- The matching process does not employ templates or feature extraction, but rather represents the comparison of actual samples.

More recently, the National Institute of Standards and Technology (NIST) has developed a research team to further research the applicability of DNA in biometric authentication (National Institute of Standards and Technology, 2010a).

#### 2.2.4.2 Facial Recognition

Facial recognition is used by humans each day to recognise acquaintances (Jain et al., 2004). In an authentication system, a camera is used to obtain an image of a face. From an acceptability perspective, this is non-intrusive because most people are accustomed to having their photographs taken.

A common approach to facial recognition, which is easily understood, is based on the shape, location and spatial relationship of facial attributes such as eyes, eyebrows, nose, lips, and chin (Jain et al., 2004). For a biometric authentication system, identifiable points associated with facial attributes are determined, and the distances between these points measured. There are about 80 measurements attainable, however, only 14 to 22 are required for the facial recognition system.

Some measurements include:

- Distance between eyes.
- Width and length of nose.
- Depth of eye sockets.
- Distance between cheekbones.

- Distance between points on the jaw line.
- Distance between points on the chin.

Another approach to facial recognition is based on the overall (global) analysis of the facial image, where the facial image is represented as a weighted combination of a number of formulary faces (Delac and Grgic, 2004).

Possible changes in the following aspects of image capture, are potential limitations associated with the capture of the facial images:

- Facial expression.
- Orientation of the face (in relation to the camera).
- Distance from the camera.
- Location background.
- Ambient lighting conditions.

Without suitable control of such contextual information, there is doubt if current facial recognition techniques can provide person recognition to an acceptable level of confidence (Jain et al., 2004). However, research in this field is yielding ever more robust and accurate systems.

#### 2.2.4.3 Iris Pattern Recognition

The iris is the coloured ring of textured tissue between the pupil and the white of the eye (Matyas Jr and Riha, 2000). The formation of each iris is stabilized by the age of two, and even twins have different iris patterns. Irises can only be altered by surgery, and it is therefore very difficult to impersonate someone based on this characteristic.

This textured tissue of the iris possesses a unique mesh-like structure of features forming a complex pattern (Daugman, 2004). The pattern can contain some of the following features: arching ligaments, ridges, crypts, corona, freckles, pits, furrows, straights, and rings.

Iris recognition uses pattern recognition techniques based on high-resolution images of an individual's eyes (Matyas Jr and Riha, 2000). The image is captured by a camera positioned directly in front of the eye at a distance of 10 to 40 cm. Subtle infrared illumination reduces specular reflection from the convex cornea to create images of the intricate structures of the iris.

Algorithms required for image acquisition and one-to-many matching (i.e. identification) were pioneered by John G. Daugman, PhD, OBE (University of Cambridge Computer Laboratory). After isolating the iris, each iris pattern is demodulated to extract its phase information using quadrature 2-D Gabor wavelets (Daugman, 2004). The encoding process amounts to a patch-wise phase quantization of the iris pattern, which determines an 'iriscode' that characterises the iris. This iriscode can be used for both identification and verification processes.

Iris recognition is considered reasonably non-intrusive, relatively simple to collect samples, and has demonstrated high levels of accuracy for both identification and verification purposes (Matyas Jr and Riha, 2000). The iris pattern remains stable over ones lifetime, however, the iris is subject to several diseases which would void the use of this biometric characteristic by those persons affected. One other possible limitation associated with iris recognition may be the expense of software for iris detection and pattern recognition; one company holds all world-wide patents on iris recognition concepts including those developed by Daugman.

#### 2.2.4.4 Retinal Pattern Recognition

According to Webster's dictionary (Websters, 2010c), the retina is the multilayered light-sensitive membrane lining the inner posterior chamber of the eyeball. The human retina is stable from birth to death, apart from the action of diseases. It receives images produced by the lens, converts them into chemical and nervous signals which reach the brain by way of the optic nerve.

The arrangement of blood vessels in the retina forms a rich pattern that is unique for each eye of each individual (Jain et al., 2004). This pattern is claimed to provide the most secure biometric, as it is very difficult to change or replicate the retinal pattern.

Retinal pattern recognition has the highest performance accuracy of any of the biometric characteristics, estimated to be in the order of 1:10,000,000. Unlike some biometric identifiers, such as fingerprint recognition, retinal pattern recognition cannot be fooled. The retina of a deceased person quickly decays and cannot be used to deceive a retinal scan.

The main limitation of retinal pattern recognition is that the retinal scanning process is highly intrusive (Matyas Jr and Riha, 2000). A subject must keep their eye in very close proximity to the scanner, and keep their vision focused on a specific point for the duration of scanning. During scanning an infrared laser beam is directed through the cornea of the eye. This level of intrusiveness is likely to be acceptable only in situations where very high security access is required.

#### 2.2.4.5 Speaker Recognition

Speech recognition or speech processing is a field of study concerned with recognising what a speaker is saying (Markowitz, 2000). Speaker recognition is concerned with the physiological characteristic of the human voice and the behavioural aspects of speaking, that are unique to each individual. Researchers in the field of speech processing consider the term voice recognition to be an erroneous term referring to speaker recognition. As biometrics is concerned with identifying or verifying a person's identity (in this case, the speaker), speaker recognition seems a more accurate term and is henceforth used in this document.

The physiological characteristic of voice is dependent on the spatial characteristic of the vocal tract, mouth and nasal cavities (Jain et al., 2004). A voice sample is captured by a conventional microphone, so this biometric characteristic is non-intrusive. The sample is then processed and stored as a template called a 'voiceprint'. Though commercial systems have been available since the 1970's, the accuracy attained by them still needs improvement (Matyas Jr and Riha, 2000).

The behavioural aspects of speaking are concerned with lip and jaw movement during the act of speaking (Jain et al., 2004). Behavioural components of variation can result from emotional state, medical condition, and changes as a result of aging.

#### 2.2.4.6 Fingerprint Recognition

A fingerprint is produced when the bulbous region of a finger tip makes contact with another surface, thus creating a duplicate impression of the finger tip (Faulds, 1880; Galton, 1892).

The most prominent traits of a fingerprint impression are caused by the papillary ridges and the consequent valleys or furrows of the epidermal layer of the finger. These traits form a pattern (known as the ridge pattern) that is distinguishable to the naked eye. There are also minute characteristics of the individual ridges (known as minutia points or minutiae) that are not easily distinguishable.

There has been a large amount of research, producing a large body of literature, concerning both identification (fingerprint classification) and verification techniques for fingerprint recognition. These issues are discussed in detail in Chapter 4 (Fingerprint Recognition).

#### 2.2.4.7 Palmprint Recognition

Like fingerprints, palmprints contain ridges and valleys that form unique patterns for each individual. As the palm is larger than a finger, it is expected that palmprints may consist of more features than fingerprints and consequently be more distinctive (Jain et al., 2004). However, palmprint scanners are much larger than fingerprint scanners, because they need to capture a larger area. This is a limiting factor for their use at workstations and mobile devices (Matyas Jr and Riha, 2000).

There are two popular approaches to palmprint recognition. One approach is to transform palmprint images into specific transformation domains. Techniques such as gabor filters, fourier transforms, and wavelets can be used to obtain salient features. Another approach is to extract principal lines and creases from the palm image, though this approach is not considered to be distinctive enough for authentication purposes (Connie et al., 2003). Also, extracting discriminatory line structures is complex and difficult; creases and ridges of the palm cross and overlap each other, which complicates the feature extraction task.

A recent research effort used the data from two and three dimensional palmprint scans (captured simultaneously) (Li et al., 2010). The principal line and palm shape features were extracted and used to accurately align the palmprint; matching rules were defined that used the 2D and 3D features for efficient recognition.

#### 2.2.4.8 Hand Geometry

Hand geometry involves the measurement and analysis of a number of spatial attributes of the human hand (Jain et al., 2004; Fong and Seng, 2009). Unlike fingerprints, the human hand is not unique. Individual hand features are not descriptive enough for identification. However, by combining various individual features and measurements of fingers and hands, it is possible to devise a method for verification purposes.

The hand attribute candidates include (Jain et al., 2004; Fong and Seng, 2009):

- The overall shape and dimensions of the hand and fingers.
- The size of the palm.
- The lengths and widths of the fingers.
- The location of joints on the fingers.

The shape of a person's hand does not normally change significantly after a certain age, so this characteristic has a good permanence rating. However, hand geometry is not known to be sufficiently distinctive to be considered suitable for a biometric authentication system with a large population database (Matyas Jr and Riha, 2000). Also, current hand geometry scanners can not detect whether a hand is attached to a living person or not; therefore if the correct pressure is applied to the scanning device, a fake hand can deceive the system (National Centre for State Courts, 2006).

More recently, research into hand geometry has investigated hand contours. This method matches registered and query templates according the finger or palm contour outline of a hand (Fong and Seng, 2009).

#### 2.2.4.9 Keystoke Dynamics

Keystroke dynamics is a behavioural characteristic that involves analysing a computer user's habitual typing pattern when interacting with a computer keyboard (Monrose and Rubin, 2000).

This biometric characteristic has been increasingly researched in the last 20 years, because data collection is simpler and less expensive when compared to other characteristics which rate higher in Table 2.1. The characteristic is not without its downfalls however, and as keystroke dynamics is utilised in the research described in this thesis, a detailed review of into this biometric characteristic is presented in Chapter 3 section 3.4.

#### 2.2.4.10 Signature Recognition

A signature is a handwritten, often stylized, depiction of someone's name that a person writes on documents as a proof of identity. Signatures act as a seal, and form a basis for non-repudiation in contract law.

In biometric technology, signature recognition is based on the dynamics of making the signature (Matyas Jr and Riha, 2000). That is, verification is not based on what the signature looks like, but on the data collected about the writing process. The signature is captured using a tablet and a special stylus which measures the pressure, direction, acceleration and length of strokes, the number of strokes and the duration.

Signatures are behavioural, so they can vary substantially between successive instances, and may change over time (Jain et al., 2004). A factor that may limit the acceptability of this biometric characteristic, is that the digitalised signature looks slightly different from a person's hand written signature. Also, the person cannot see each character as they are writing.

More recent approaches do not require the use of a tablet and stylus, but employ a video camera that is focused on the user writing on a piece of paper with a normal pen (Impedovo and Pirlo, 2007). Parameters that may be used for verification (adopting this approach) are the signature image area, signature height and width, length to width ratio, middle zone width to signature width ratio, and the number of characteristic points.

#### 2.2.4.11 Gait Recognition

Gait analysis is the process of quantification and interpretation of animal (including human) locomotion (Boyd and Little, 2005). In the pathological field, the study of gait analysis may reflect bodily compensations for underlying symptoms, allowing orthopaedic diagnoses to be made.

Gait analysis commonly involves the measurement of the movement of the body in space (kinematics) and the forces involved in producing these movements (kinetics), and is widely used in professional sports training to optimise and improve athletic performance.

In biometrics, gait is both a physiological and behavioural characteristic that attempts to identify the specific way an individual walks (Jain et al., 2004). By studying the variations in gait style this biometric can be used as an identifier to differentiate between individual people.

The classification of gait style can be grouped into two categories of measurement:

1. Spatial-temporal. That is, step length, step width, walking speed, cycle time.
2. Kinematic. That is, the joint rotation of the hip, knee and ankle; the mean joint angles of the hip, knee and ankle; and the thigh, trunk and foot angles.

Although human beings may use gait to help recognise people (often subconsciously), it is not considered to be distinctive enough for an authentication system (Jain et al., 2004). A limitation with this biometric is the complexity of dealing with the spatial, temporal and kinematic measurements (in different environments). Also, it scores low in permanence because of changes to the physiological condition of an individual's body over time.

#### 2.2.4.12 Body Odor Recognition

Any organic body that exudes an odor constantly produces tiny quantities of molecules that evaporate to produce the odorant (Korotkaya, 2003). So, body odor is a characteristic of the changing chemical composition of the body.

In biometrics, measurement is achieved by recording data as a stream of odor laden air is blown over an array of chemical sensors (Jain et al., 2004). Each sensor is

sensitive to the different properties of groups of aromatic compounds , and produces a characteristic pattern for each odorant (Korotkaya, 2003). This is a non-intrusive process, as the odor stream is usually taken from the back of the hand (Matyas Jr and Riha, 2000).

A limitation with this characteristic is that there is variability in the quantity of molecules produced at any given time; the body does not produce a steady odor stream like sound waves. Also, collectability of this characteristic may be affected by the use of soap or hand wash, and environmental conditions. Illness is also known to affect body odor, which may cause occasional problems in terms of permanence.

#### 2.2.4.13 More Detailed Discussion

As previously stated, Chapter 3 (Keystroke Dynamics) and Chapter 4 (Fingerprint Recognition) discuss these two biometric characteristics in greater detail, because they were used in the author's investigation into a multi-modal biometric system, and so a more in-depth discussion was warranted.

The next section 2.3 provides an overview of data fusion and then discusses these concepts in relation to multi-modal biometrics.

## 2.3 Data Fusion And Multi-Modal Biometrics

### 2.3.1 Data Fusion

Data fusion is the integration (merging or combining) of data from multiple sensors or sources (Hall and Llinas, 1997). Multi-sensor data fusion is a relatively new science, though the ideas and concepts behind it are not.

An analogy can be drawn between multi-sensor data fusion and the use of senses by human beings and animals, who develop the ability to combine information from multiple senses from an early age. Whereas the use of a single sense may provide some useful information, the use of multi-sensory data can provide more meaningful information, to more accurately assess environmental conditions for survival purposes.

From a survival perspective, reliance on a single sense would be imprudent because (Hall and Llinas, 1997):

1. Input from several senses provides more information that can be used for the task at hand. For example: when assessing the quality of edible substances, using a combination of taste, sight, smell and touch gives more meaningful information than using sight or smell alone.

2. If any one of the senses becomes damaged, the others must be able to compensate. An example of the development of compensatory skills is visually impaired people, who are known to have more highly developed hearing skills than the average person.

Thus, multi-sensory data fusion is naturally performed by humans and animals to accurately assess their environment for threat identification, to improve their chances of survival. In the same way, merging data from multiple sensors/sources improves accuracy of a data fusion system that can be applied to industrial processes, medical diagnosis, and military applications (Hall and Llinas, 2001).

Using data from multiple independent sensors/sources, makes a system less vulnerable to the failure of a single source (Brooks and Iyengar, 1998). Provided the fusion method is appropriate to the types of data, and is performed correctly, the system should become more robust.

Combining data from several different sources not only enables a system to attain more accurate information than would otherwise be possible—and therefore achieve more specific inferences than could be achieved by using a single independent source—it also makes the system less sensitive to noisy data.

Decision making is most often dependent on different sources of data related to a given situation. This can introduce the problem of too much data and can lead to information overload. A fusion system must combine data in such a way as to remove the influence of inconsistent or irrelevant data, so that the best interpretation of information is achieved. The resultant information can be stored in one coherent structure.

When properly processed, fusion combines inputs from multiple independent sources of particular accuracy and reliability, to provide information of known accuracy and proven reliability.

For data fusion systems utilising more than one data source, the fusion process generally involves (Brooks and Iyengar, 1998):

- A sensor: an electrical or mechanical device that maps the values of some environmental attribute to a quantitative measurement. In the current study, the devices used were the standard computer keyboard and the fingerprint scanner[3]. Every sensor detects some aspect or attribute of the state of its environment. This involves direct interaction with the environment. When no change in environment is detected, the sensor is inactive; if change in environment is detected by the sensor, it becomes active and records the appropriate data. Individual sensors are not totally reliable interfaces for several reasons:
  - They have limited accuracy.
  - They are subject to the effects of some type of 'noise'.
  - Under some conditions they will function incorrectly.
  - Sensor interaction generally causes wear and tear on the device, eventually leading to component failure, inaccuracy or poor precision.
- Pre-processing: involves the processing of raw data—read from a sensor—into a form that facilitates the fusion process. Tasks include; sensor re-calibration, noise reduction, re-aligning skewed individual readings, edge detection, feature detection, and possible treatment of features to produce a more manageable form.
- Fusion: this process involves integrating readings from multiple independent sources into a single reading/vector. The fused reading/vector will be in the same form as the pre-processed data. The actual values utilised in the fusion process may differ from many or all inputs. The fusion process must be based

[3]For keystroke dynamics, time intervals in milliseconds between keystroke events were measured. For fingerprints, global and local features from the scanned grey-scale fingerprint images were extracted.

on mathematically rigorous methods that avoid naively propagating errors in the system. Statistical and deterministic fusion methods exist for this purpose.

- Interpretation: this process is task-specific and consists of finding the best fit possible for the data within the information requirements of the system. Application areas include: aeronautics, biometrics, hazardous environments, manufacturing, medicine, remote sensing, robotics, and traffic control.

Each of the above steps is dependent on the results of the preceding step. Data fusion attempts to overcome the limitations stated above—making the most of existing technology—by finding ways to integrate data from multiple independent sources and by appropriate handling of any discrepancies encountered.

The next section (2.3.1.1) discusses the paradigms for fusing data from multiple independent sources.

#### 2.3.1.1 Paradigms of Data Fusion

When fusing data from multiple sensors/sources, there are three standard data fusion configuration paradigms available for consideration (Brooks and Iyengar, 1998):

1. The Complementary Data Fusion paradigm is utilised where sensors or independent data sources do not depend on one another directly, but can be combined to provide a more complete mapping of a region, physical attribute, or aspect of an object.

    - The sensors provide data covering *different* regions, physical attributes, or aspects of an object.
    - Merged data provides an overall view of the whole region or object.
    - Merging data can be simply achieved since no conflicting information is present in the data (unless the regions overlap).
    - Multiple sensors of the same type, or of different types, may be deployed.

    As an example, Figure 2.2 illustrates four radar sensors placed on each side of a rectangular structure, with the sensors facing away from the structure.

Each radar provides data regarding the region within its own scope. Their views are in different directions and although the data may be of similar nature, they will illustrate different geographical attributes. By merging or fusing all views, a more complete view of the entire environment is attained.

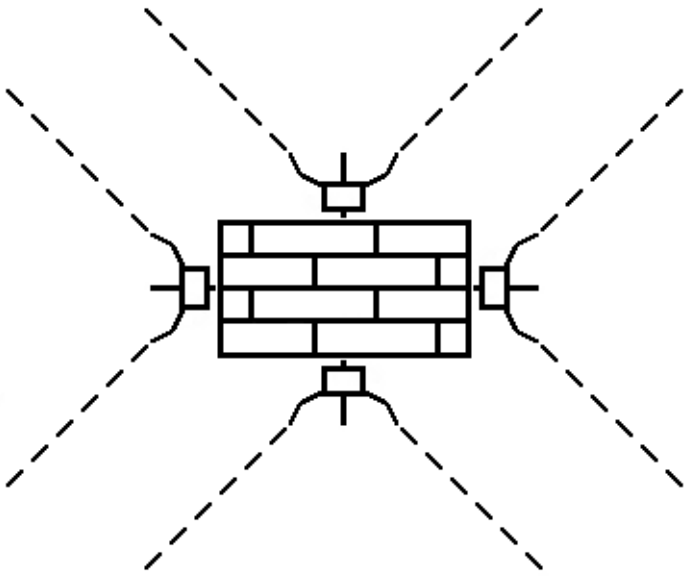

Figure 2.2: Complementary Data Fusion Paradigm
The image was sourced from Brooks and Iyengar, 1998.

2. The Competitive Data Fusion paradigm is utilised where sensors or data sources provide independent measurements of the *same* information about a region, physical attribute, or aspect of an object. That is, they observe the same region (or object) and provide their own data from those observations. They are competitive in that a data fusion system needs to decide which sensor's data is to be accepted as the most correct. That is, has the least discrepancies.

   - Theoretically, the data from all sensors should be of identical nature.
   - Merging data is challenging since is involves interpreting conflicting data.
   - Competitive configurations are often used in mission critical situations to provide better reliability or fault-tolerance in a system.

- Competitive configurations generally cause accuracy and reliability to increase.

As an example, Figure 2.3 illustrates four radar sensors placed on each side of a (hollow) rectangular structure. However, this time the sensors are placed in such a way as to be observing the inside area within the structure (though each radar will observe from a different perspective). Each sensor's data should describe their own perspective of the structure. The data with the least number of discrepancies is chosen as the most representative. Sensor accuracy, sensor perspective, and the time between readings make fusion problematic and complex.

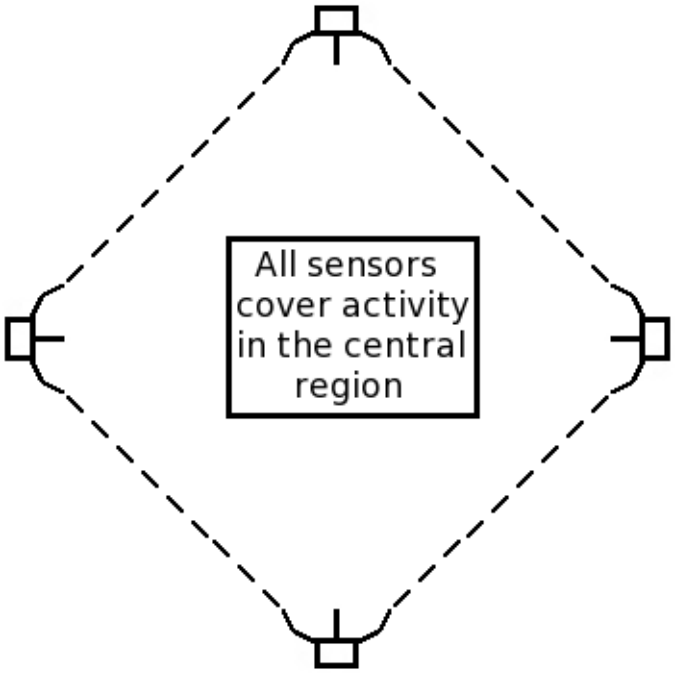


Figure 2.3: Competitive Data Fusion Paradigm
The image was sourced from Brooks and Iyengar, 1998.

3. The Cooperative Data Fusion approach is related to the competitive paradigm, in that it is utilised where sensors or data sources provide independent measurements of the *same* information about a region, physical attribute, or aspect of an object. However instead of competing, data is combined in such a way that information can be derived, that would be otherwise unattainable from individual sensors/data sources.

- This is a difficult system to design because information sought is sensitive to the inaccuracies associated with individual sensors.
- Cooperative configurations typically require determination of the specific features (from each of the data sources) that best represent the region, physical attribute, or aspect of an object. Thus feature selection may be required to determine the most appropriate features (from each of the data sources) to combine.
- Cooperative configurations may cause accuracy and reliability to decrease, if steps are not taken to ensure rigorous feature selection.

As an example, Figure 2.4 illustrates 2 video cameras observing the same object from different locations. The data from each will give information in two dimensions. However, if the cameras' angle of separation is sufficient, the combined data can provide information that infers three dimensions. That is, gives the impression of depth.

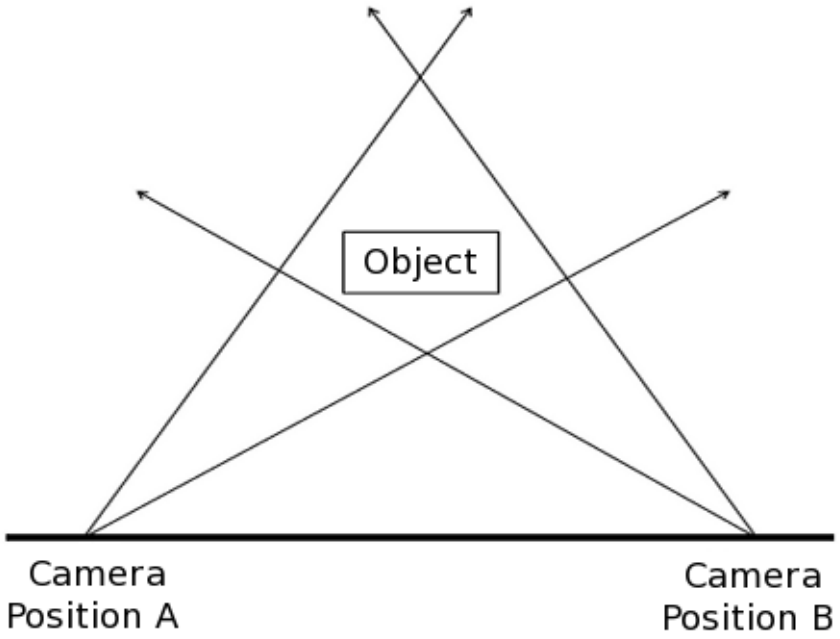


Figure 2.4: Cooperative Data Fusion Paradigm
The image was sourced from Brooks and Iyengar, 1998.

In the current study, the complementary and cooperative paradigms were implemented and tested (refer Chapter 5 section 5.6). The competitive paradigm was not investigated because it was considered contrary to the purpose of the experiment.

The fundamental premise for fusing data from multiple sources was to improve accuracy and robustness in the authentication procedure. That is, a multi-modal biometric authentication system should be harder to circumvent than a uni-modal biometric authentication system.

In the competitive paradigm, a single data source is selected to represent a region, physical attribute, or aspect of an object. Therefore, if this paradigm was implemented in an authentication system, only one data source would be used—resulting in a uni-modal biometric authentication system—which is at variance with the intention of multi-modal biometric authentication.

The next section (2.3.1.2) discusses the levels at which data fusion may be performed.

#### 2.3.1.2 Formal Levels of Fusion

The field of multi-sensor data fusion has been applied primarily in the mathematical and engineering fields (Hall and Llinas, 1997). As such, the literature has seen development of some formal specifications.

The discussion in this section deals with the formal specification of three levels at which multi-sensor data may be fused (Hall and Llinas, 2001):

1. Sensory Data Level. If the sensor/source data (from multiple sensors/sources) are comparable—that is, sensors/sources are measuring the same physical attribute/phenomena—then the raw data can be directly merged. Techniques include classic estimation methods (such as Kalman filtering). If the sensor/source data are non-commensurate, then they must be fused at the feature level or decision level.

2. Feature Level. This fusion level involves the extraction of representative features of each sensor/data source. Feature data is processed into an appropriate format and stored in individual feature vectors; these feature vectors are subsequently fused. Pre-processing may entail the selection and extraction of the 'most' representative features from sensor/source data, before combining them into a single feature vector. The ultimate success relies on the selection of

'good' (appropriate) features (such features should provide excellent class separability in feature space). Selection of 'poor' (inappropriate) features results in large overlapping feature space areas for class objects. The resultant feature vector serves as input into pattern recognition techniques such as neural networks, clustering algorithms, or template methods.

3. Decision Level. This fusion level involves the processing of the sensor/source data from each source to achieve high level inferences, which are subsequently combined. Sensor/source information are combined after each sensor/source has made a preliminary determination based on independent data. That is, this level of fusion entails the integration of decisions coming from different sensors/sources (Varshney, 1997). Decisions or inferences can be made based on raw data or extracted features. Techniques include weighted decision methods (voting techniques), classical inference, and Bayesian inference.

Fusion levels applied to biometrics vary slightly in interpretation to the three levels just discussed. The specific details are provided in the discussion in section 2.3.2.1.

As specified in the title of this thesis, the aim of the study was to fuse data at the feature level because it has been proposed that data fused at this level maintains more discriminative information (than fusion at the decision level), and should thus provide improved accuracy when applied to the authentication procedure.

The next section (2.3.1.3) discusses the differences in the nature of data, and the treatment that needs to be performed on that data, for an acceptable fusion of data.

#### 2.3.1.3 Data Alignment

Data alignment refers to the processing required to modify data, received from multiple sources, such that it permits that data to be reasonably compared and associated (Hall and Llinas, 2001). Prior to fusion, data from individual sources must be transformed into a consistent format that is suitable for subsequent processing. That is, data from multiple sources can only be effectively combined if the data are compatible in format and consistent in their frame of reference.

There are five possible processes involved in data alignment (Bowman and Steinberg, 2001):

1. Common Formatting: transforming data to standard data types and/or units of measure. Typically performed by unit adjustment algorithms.

2. Time Propagation: extrapolating old time values to current time. Typically performed by temporal adjustment algorithms.

3. Coordinate Conversion: translating data from various coordinate systems to a common spatial referencing system. Typically performed by spatial reference adjustment algorithms.

4. Mis-alignment Compensation: correcting displacement errors in observations.

5. Evidential Conditioning: assigning confidence values associated with data attributes.

In the current study, there were two data sources (keystroke dynamics metrics and fingerprint features) that were not compatible in format or frame of reference. The keystroke dynamics metrics consisted of numerical values only, whilst the fingerprint features consisted of numerical values and two dimensional coordinate values. Thus data alignment was necessary, and the processes involved are discussed in detail in Chapter 5 section 5.5.5.

The next section (2.3.2) provides a discussion of multi-modal biometrics, which utilises data fusion.

### 2.3.2 Multi-Modal Biometrics

In recent times, research in the field of biometrics for authentication purposes, has increasingly investigated the use of multiple biometric characteristics (multi-modal biometrics). This section defines multi-modal biometrics, and discusses the benefits (which have lead to the increased interest) of a multi-modal biometric system over a uni-modal biometric system.

The levels of data fusion as applied to multi-modal biometrics are discussed in section 2.3.2.1, and a brief review of experimental work in this area of research is provided in section 2.3.2.2.

Multi-modal biometrics refers to the use of more than one biometric characteristic for person recognition. A multi-modal biometric system encompasses the necessary processing required to incorporate the chosen multiple biometric characteristics into the authentication procedure.

Research into multi-modal biometric systems has eventuated because uni-modal biometric systems may suffer from any of the following limitations:

- Noise in sensed data (Ross and Jain, 2004). This could be a result of a defective or mal-adjusted sensing device, or because of physical or behavioural changes (either temporary or permanent) to the biometric of the person involved.

- Intra-class variations (Ross and Jain, 2004). These typically result from incorrect or inconsistent interaction between the user and the sensing device.

- Inter-class similarity (Frischholz and Dieckmann, 2000; Ross and Jain, 2004). A single characteristic may sometimes fail to be exact enough for identification (or verification). For example, identical twins have facial features that could prove impossible for a biometric system to consistently differentiate.

- Non-universality (Frischholz and Dieckmann, 2000; Ross and Jain, 2004). The chosen characteristic may not always be readable, resulting in an inability to acquire meaningful biometric data. For example, about 2% of people have fingerprints that cannot be recorded because they are obscured by cuts or scars, or they are too fine to show up well during acquisition (due to aging or the effects of chemical re-agents).

If for any of the preceding reasons, accuracy is not attained by one modality, the other modes may still lead to accurate authentication (Frischholz and Dieckmann, 2000). A multi-modal biometric system makes it very difficult for an impostor to simultaneously impersonate the multiple character traits of a legitimate user (Ross et al., 2001).

For example, impersonating facial features, vocal features and fingerprint features in real time (via the authentication procedure) would be much more difficult than impersonating only one character trait.

Thus the limitations of a uni-modal biometric system may be overcome or mitigated by using a multi-modal biometric system. Researchers believe that the use of multi-modal biometrics will provide a more reliable and robust system (Frischholz and Dieckmann, 2000; Ross et al., 2001; Ross and Jain, 2004).

#### 2.3.2.1 Levels of Fusion In Multi-Modal Biometrics

This section discusses the levels of fusion at which multi-modal biometric systems may typically operate. During the discussion, the relationship between the formal levels of data fusion covered in section 2.3.1.2 and those used in a multi-modal biometric system will be noted.

As discussed in section 2.2.2 (and illustrated in Figure 2.1), a generic uni-modal biometric system consists of four modules (Ross and Jain, 2004; Faundez-Zanuy, 2009). For a generic multi-modal biometric system, a fusion module must be added (refer Figures 2.6, 2.7, and 2.8).

A description of the roles for the four modules in a generic uni-modal biometric system was provided in section 2.2.2. The fusion module (added to the generic multi-modal biometric system) is responsible for combining the data from multiple sources, and the proposed level of fusion determines its location in the system.

Figure 2.5 demonstrates that fusion can occur before the matching process or after the matching process (Poh and Kittler, 2008). If fusion occurs before the matching process, data may be fused at either the sensor or feature level. If fusion occurs after the matching process, data may be fused at either the confidence score, rank, or decision levels.

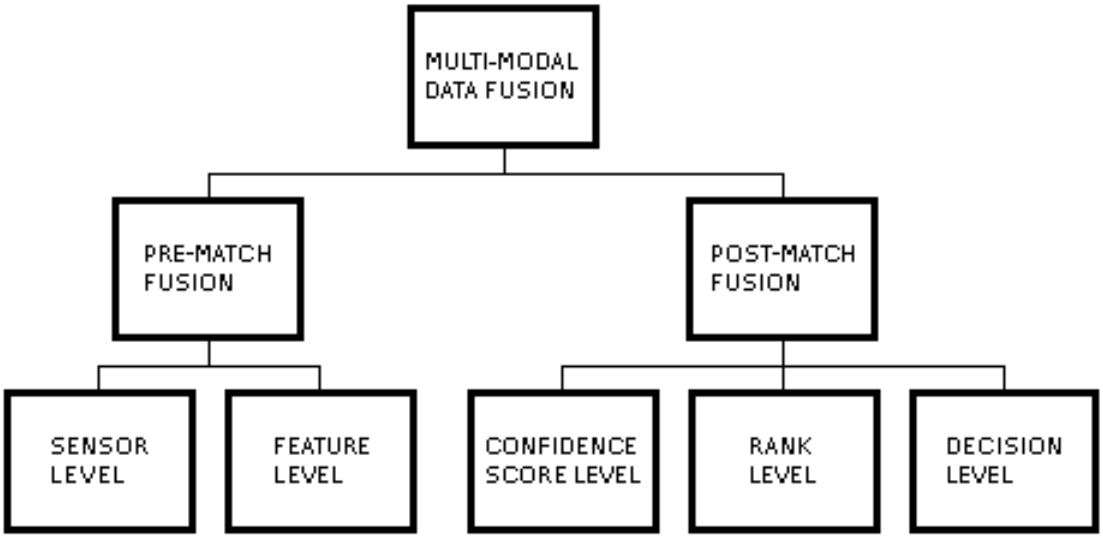


Figure 2.5: Data Fusion Levels In Multi-Modal Biometrics
The image was sourced from Poh and Kittler, (2008).

The data fusion levels in a multi-modal biometric system, are described as follows (Ross and Govindarajan, 2005; Poh and Kittler, 2008):

- Sensor Level. Raw data represents the richest source of information (though it may possibly be contaminated by noise). Raw data can be processed such that a new, single vector, consisting of the integrated raw data, is obtained. The newly created raw data vector may be processed directly or features may be extracted from it. An important caveat regarding sensor level fusion is that it is only possible to fuse data when samples of the same biometric trait are used (Nandakumar, 2008). That is, where raw data from multiple instances using the same sensor, or samples from multiple sensors, provide readings of the same biometric. Consequently, this level of fusion is rarely attempted in multi-modal biometrics because raw data (from multiple modes) can not be meaningfully combined.

  Sensor level fusion in multi-modal biometrics relates to the first level of the formal data fusion levels covered in section 2.3.1.2.

- Feature Level. This fusion level uses data collected from the feature extraction process (Osadciw et al., 2003). Researchers believe that feature level fusion will result in accurate and robust authentication, because data at this level

is closer to raw data—than the subsequent fusion levels—and maintains more discriminatory information than those levels (Ross and Jain, 2004).

However, in order to achieve a high level of system performance, extensive processing is typically required (refer section 2.3.2.2). Feature extraction typically requires the selection of salient features, from the independent data sources, that best represent the entity and can provide recognition accuracy (Poh and Kittler, 2008).

Figure 2.6 illustrates the process flow recommended for feature level fusion. Firstly, the biometric capture modules are responsible for acquisition of the biometric characteristics. In Figure 2.6 there are two such capture modules, indicating a dual mode biometric system.

Following acquisition, is the feature extraction module (for each mode). At this stage, the features relating to each individual mode remain separate (that is, they remain separate feature vectors). Data alignment—to bring the features from multiple independent sources into alignment (as discussed in section 2.3.1.3)—will usually be required prior to fusion. Feature selection may also be required—prior to fusion—to reduce the size of the final feature set; otherwise the fused feature set may suffer from the 'curse of dimensionality' (Ross and Govindarajan, 2005).

After feature extraction and data alignment (and possibly feature selection), the fusion module combines the feature vectors corresponding to each independent source. If data alignment was necessary, and had been successfully performed, the fusion process becomes relatively simple. In some literature related to feature level fusion, the feature vectors are simply concatenated (Ross et al., 2001; Ross and Jain, 2004; Faundez-Zanuy, 2009; Nandakumar, 2008).

The fused vector of features is then passed to the matching module, where it is compared to a registered template. The template feature vector must have been processed in the same manner, and result in the same format, as the query feature vector. The output from the matching module is then passed to the decision module, where the final classification decision is made.

It should be apparent that the complex and intensive processing required for feature level fusion (to avail the system of the advantages of data fused at this level) is mainly associated with feature pre-processing for the appropriate representation of features, feature selection, and data alignment.

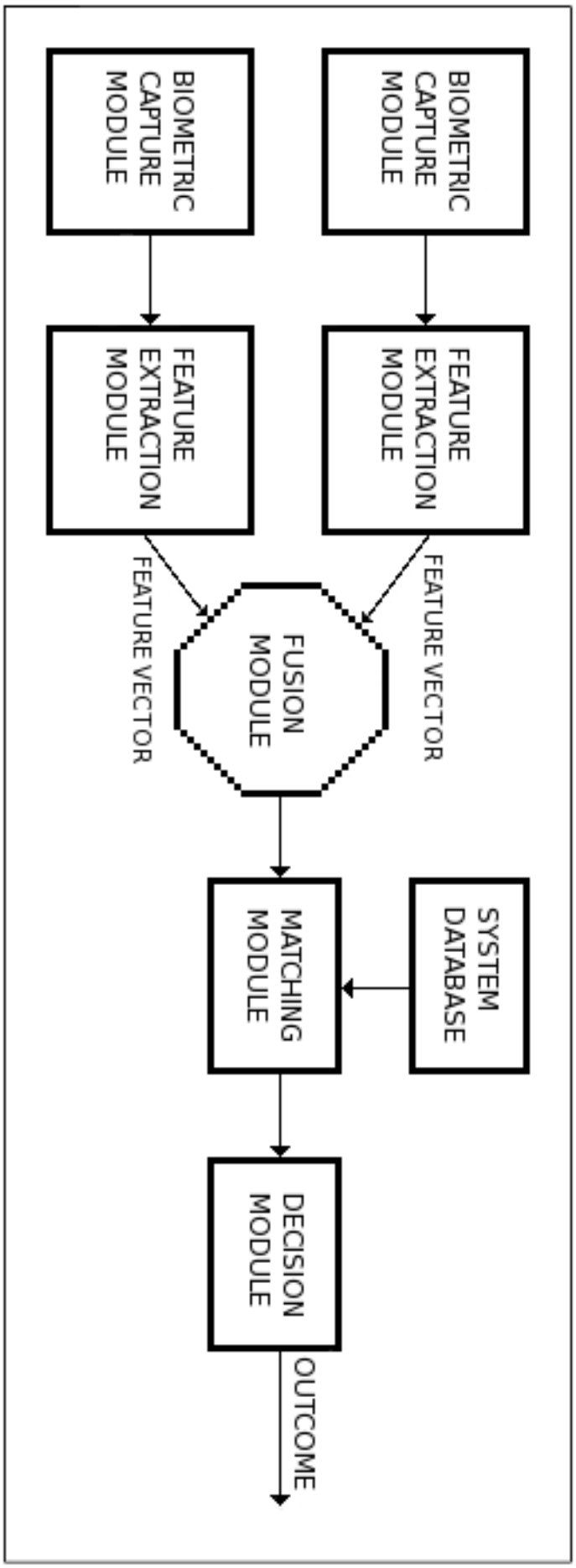


Figure 2.6: Feature Level Data Fusion
The image was sourced from Maltoni et al., (2003).

Feature level fusion in multi-modal biometrics relates to the second level of the formal data fusion levels covered in section 2.3.1.2.

- Confidence (or Matching) Score Level. Fusion at this level occurs after a matching module (for each mode) has determined a confidence or matching score (Osadciw et al., 2003; Indovina et al., 2003). The confidence score is a measure of similarity between the query and registered biometric feature vectors (Nandakumar, 2008). This level of fusion is the most commonly employed and researched of all fusion levels in multi-modal biometrics (Poh and Kittler, 2008; Nandakumar, 2008).

  Figure 2.7 illustrates the process flow recommended for confidence score level fusion. The biometric capture and feature extraction modules perform their tasks as they would for feature level fusion. However, fusion at the confidence score level does not involve the integration of the feature vectors. The individual feature vectors are passed to the separate matching modules for comparison with registered templates. That is, instead of a one vector comparison (as with feature level fusion), there are multiple vectors compared with their corresponding templates (in the appropriate matching modules).

  A confidence score is calculated in each matching module, according to the above comparison, and passed to the fusion module. The confidence score is essentially a probability score in the continuous domain (typically in the interval $[0, 1]$). The scores from the individual matching modules are combined in the fusion module, into a single scalar value (typically in the intervals $[0, 1]$ or $[0, 100]$), which is then passed to the decision module. The decision module makes the final classification decision, based on this fused confidence score. Fusion at this level requires less processing (than feature level fusion) in order to achieve an appropriate level of system performance.

  Confidence score level fusion in multi-modal biometrics is not specifically defined among the formal data fusion levels covered in section 2.3.1.2. One could conjecture that it may fit into the decision level of the formal data fusion levels.

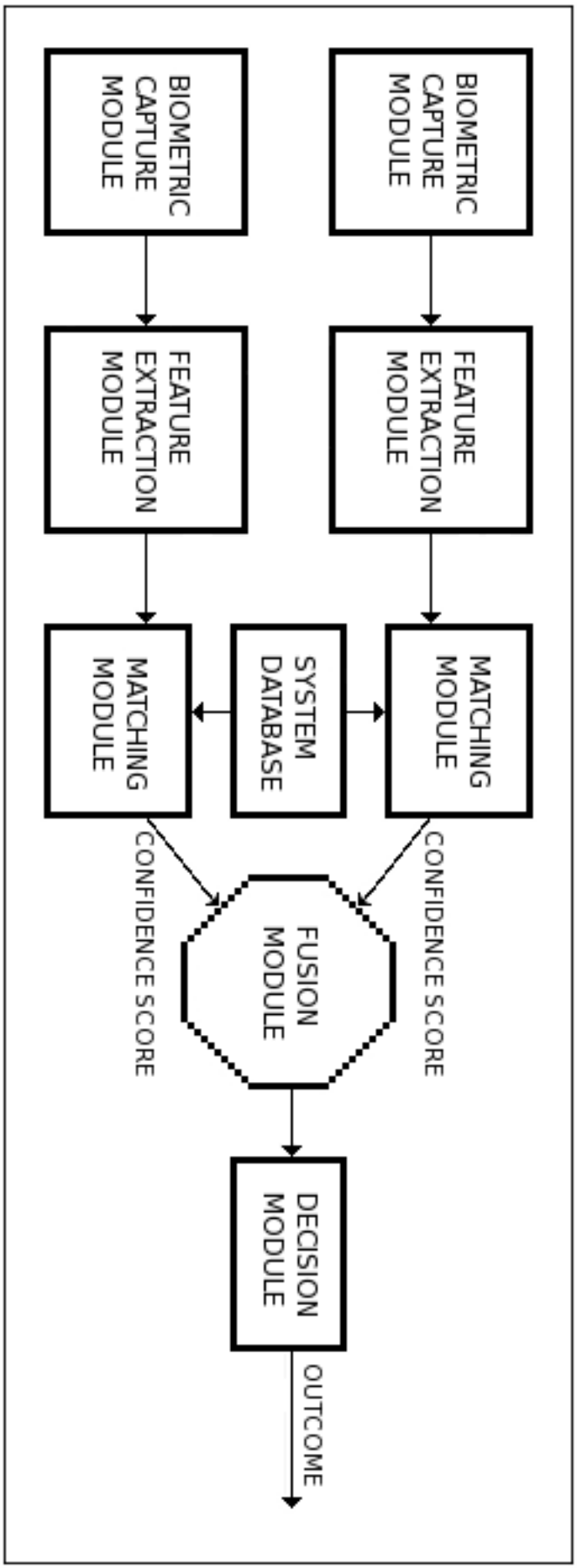


Figure 2.7: Confidence Score Level Data Fusion
The image was sourced from Maltoni et al., (2003).

- Rank Level. Fusion at this level is more relevant to identification than verification (Ross and Govindarajan, 2005). For every enrolled identity in a database, the confidence scores for each mode are sorted in descending order and ranked such that the lowest rank indicates the 'worst' match and the highest rank indicates the 'best' match (with all other confidence scores appropriately ranked). A fusion method is used to consolidate the individual confidence scores.

  Rank level fusion in multi-modal biometrics is not specifically defined among the formal data fusion levels covered in section 2.3.1.2, though being a subset of confidence score level, it may fit into the decision level of the formal data fusion levels.

- Decision Level. Fusion at this level is the most abstract, where accept/reject decisions—from multiple data sources—are consolidated into one final classification decision (Osadciw et al., 2003).

  Figure 2.8 illustrates the process flow recommended for decision level fusion. The biometric capture, feature extraction, and matching modules perform their tasks as they would for confidence score level fusion. However fusion at the decision level, does not involve the integration of the confidence scores. The individual confidence scores are passed to the decision modules, where accept/reject decisions (output as boolean values) are made for each mode.

  Each decision module makes its classification decision (based on the confidence score passed to it), and passes its decision to the fusion module. The multiple boolean values are integrated to generate the final classification decision.

  Fusion at this level takes advantage of the processing performed by each mode's matching and decision modules to arrive at an individual decision. This makes decision level fusion more scalable than the other levels.

  Fusion at this level requires the least amount of processing (compared to the other levels) in order to achieve an appropriate level of system performance. Decision level fusion in multi-modal biometrics relates to the third level of the formal data fusion levels covered in section 2.3.1.2.

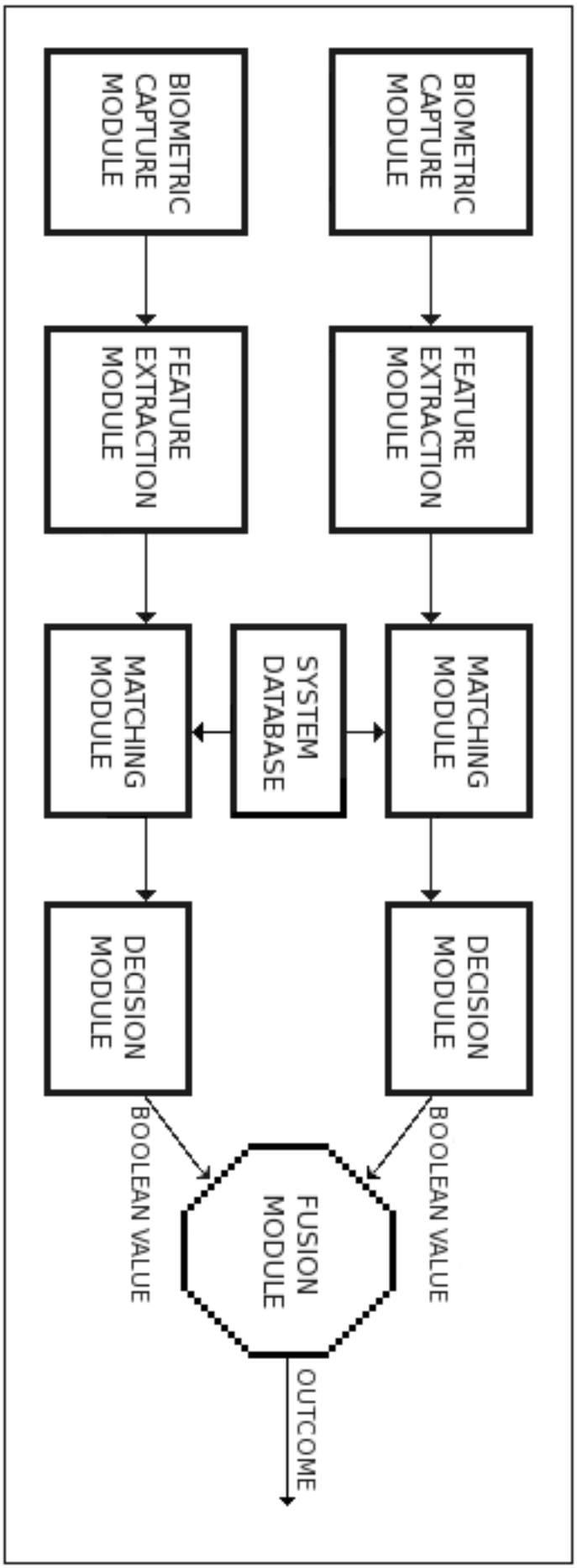


Figure 2.8: Decision Level Data Fusion
The image was sourced from Maltoni et al., (2003).

The next section presents a review of some experimental efforts in multi-modal biometric authentication.

#### 2.3.2.2 Review of Multi-Modal Biometrics Research

This section provides a review of some research efforts in the field of multi-modal biometrics[4]. As will be discovered, early work predominantly involved fusion at the confidence score or decision levels. More recently, work involving fusion at the feature level has been undertaken.

The review will focus on the data fusion related issues (such as fusion paradigms, fusion levels, and fusion methods). Other experimental factors (such as data treatment) and methodological issues, are not considered in this review for the following reasons:

- Such issues—related to the particular biometric characteristics used in this study—are discussed in the literature reviews in Chapter 3 section 3.4 (for keystroke dynamics) and Chapter 4 section 4.5 (for fingerprint recognition).

- For the purpose of comparing results between the reviewed papers and the current study, the interest is primarily concerned with data fusion issues.

The research findings and data fusion issues relating to the experiments involving multi-modal biometrics, as conducted by the authors of the papers reviewed, are summarised in Table 2.2[5]. As well a being useful as a quick reference for the following discussion, the information in Table 2.2 will be used in Chapter 7 to compare results achieved in the current study with those of previous research.

[4]It should be noted that the review is by no means a comprehensive coverage of all work done in this field. Rather the research efforts reviewed here were chosen to provide an overview of the current status in this field, and to provide figures with which to compare the results of the current experiment.

[5]Note that in Table 2.2, the performance variables (FAR and FRR in columns 7 and 8 respectively) are denoted as actual rates (i.e. a percentage divided by 100), even though some authors expressed them as a percentage. During the discussion, the performance variables will be presented as both the actual rates and their corresponding percentages (in parenthesis).

| Reviewed Paper | Number of Participants | Number of Modalities | Paradigm | Fusion Level | Fusion Method | FAR | FRR |
|---|---|---|---|---|---|---|---|
| Jain et al., 1999 | 50 | 3 | Complementary | Decision | JPDF | 0.0001 | 0.14 |
| Chatzis et al., 1999 | 37 | 3 | Complementary | Decision | FDKM | 0.0039 | 0.0 |
| Ross et al., 2001 | 50 | 3 | Complementary | Confidence Score | SR | 0.0003 | 0.0178 |
| Wang et al., 2003 | 90 | 2 | Complementary | Confidence Score | RBFN | 0.0 | 0.0 |
| Son and Lee, 2005 | $140 + 200$ | 2 | Cooperative | Feature | RJFV | 0.0 | 0.0 |
| Ross and Govindarajan, 2005 | 100 | 2 | Cooperative | Feature | CMS | 0.0001 | 0.13 |
| Rattani et al., 2007 | 50 | 2 | Cooperative | Feature | DT | 0.0102 | 0.0195 |
| Yao et al., 2007 | 119 | 2 | Cooperative | Feature | NN | na | na |
| Nandakumar, 2008 | $517 + 295$ | 2 | Complementary | Confidence Score | CLRT | 0.0001 | 0.009 |

| LEGEND | FUSION METHOD DESCRIPTION |
|---|---|
| na | Not available—information was not provided by the authors |
| JPDF | Joint class-conditional Probability Density Function |
| FDKM | Fuzzy Data K-Means algorithm |
| SR | Sum Rule |
| RBFN | Radial Based Function Network |
| RJFV | Reduced Joint Feature Vector |
| CMS | Combined Match Score |
| NN | Nearest Neighbour classifier |
| DT | Delauney Triangulation |
| CLRT | Complete Likelihood Ratio Test |

Table 2.2: Summary of Reviewed Literature Involving Multi-Modal Biometics

One of the earliest research efforts in multi-modal biometrics was by Jain et al., (1999a). The following is a brief description of the extraction and matching operations for the three modalities used in their experiment:

1. Fingerprint Recognition. The position and orientation of local fingerprint features (refer Chapter 4 section 4.3.2) were determined from a live-scan fingerprint image. Matching involved comparison of the two-dimensional minutiae patterns in a template with those of a query fingerprint image. A confidence score—indicating the likelihood that the two fingerprints belong to the same finger—was calculated.

2. Face Recognition. Feature extraction involved the use of the eigenface approach to obtain a feature vector of a facial image. A set of orthonormal vectors that best described the distribution of a facial image—in a lower dimensional subspace (eigenspace)—was computed. The feature vector represented the projection of the original facial image on the reduced eigenspace. Matching involved the projection of a query facial image onto the template eigenspace, and a confidence score computed based on a similarity function.

3. Speaker Recognition. Here, the acoustic properties of the speech signal were extracted. In the matching module, analysis of the identified properties (linear prediction coefficients of the cepstrum) was performed by hidden markov model (HMM). A confidence score was computed based on a similarity function.

The authors employed decision level fusion, where each matching or verification module determined a similarity or confidence score. The scores were then passed to the corresponding decision modules, where individual accept/reject decisions were made. These individual decisions were then passed to the fusion module. Fusion was achieved using the joint class-conditional probability density function, which established the final classification decision.

For the training stage (to obtain the templates), 50 participants provided 10 fingerprint samples each (a total of 500 samples), 9 facial images each (a total of 450 samples), and 12 speech samples each (a total of 600 samples).

For testing, 25 participants (a subset of the 50 participants from which the training samples were collected) provided 15 fingerprint samples each (a total of 375 samples), 15 facial images each (a total of 375 samples), and 15 speech samples each (a total of 375 samples).

From the ROC graph[6], the best results for the fusion of all three modalities were a false acceptance rate (FAR) of approximately 0.0001 (0.01%) and a false rejection rate of approximately 0.14 (14%). These results demonstrate a very good FAR, with a less than satisfactory FRR. It should be noted that the ROC curve demonstrated that the fusion of all three modalities performed considerably better than each individual modality.

Chatzis et al., (1999) used two biometric characteristics for their experiment, but employed the following five modalities (four facial feature extraction methods and one speech authentication algorithm) from these characteristics:

1. Morphological dynamic link architecture (MDLA): a facial recognition technique that employs both gray-scale and shape information.

2. Profile shape matching (PSM): another facial recognition technique that employs profile shape information.

3. Grey level matching (GLM): another facial recognition technique that uses the grey level values of an image.

4. Gabor filters to create feature vectors and implementing the dynamic link architecture (GDLA).

5. Maximum signal processing (MSP): a speech authentication algorithm based on hidden markov models (HMM).

Data fusion was performed at the decision level. The following six different decision functions were used to integrate the individual decision outputs from each of the modalities:

1. k-Means Algorithm (KM): classifies each training vector as tending to a certain cluster based on a minimum distance measure.

[6]ROC graphs are explained in Chapter 6 section 6.2.2.

2. Fuzzy k-Means Algorithm (FKM): classifies each training vector to all clusters, and assigns a membership value—in the continuous interval [0,1]—indicating the strength/weakness of the association between the vector and each of the $k$ clusters.

3. Fuzzy Vector Quantization Algorithm (FVQ). Vector quantization techniques are usually applied to lossy image compression[7] (Tsekouras, 2005). The central issue in vector quantization applications is the generation of the appropriate codebook. Most vector quantization methods perform the codebook design by employing crisp decision-making procedures (by assigning each training vector to only one cluster). The fuzzy approach treats each cluster as a fuzzy set and therefore, the codebook design is a soft decision making process.

4. Fuzzy Data k-Means Algorithm (FDKM): the inclusion of a quality measure (based on a fuzzy vector distance measure) to modify the data supplied to the fuzzy k-Means algorithm.

5. Fuzzy Data Vector Quantization Algorithm (FDVQ): the inclusion of a quality measure (based on a fuzzy vector distance measure) to modify the data supplied to the fuzzy vector quantization algorithm.

6. Median Radial Based Function Network (MRBF). A RBF network is a two layered neural network for classification or functional approximation. The hidden or middle layer neurons implement a Gaussian function to model the location and spread of clusters. An output layer neuron applies the weighted sum of the hidden layer neuron to a sigmoidal function (refer section 2.4.2.1). MRBF networks are not influenced by outliers due to the use of a median operator.

The experiment applied different combinations of modalities to the above decision functions. The best results were achieved when the following three modalities were combined:

[7]The implementation of vector quantization in image compression requires the decomposition of the image into a number of rectangular blocks. Each block forms a vector (i.e. training vector). The objective is to classify all the training vectors into a number of clusters, by minimizing a distortion measure. The centers of the resultant clusters provides the 'codebook' vectors. The image is reconstructed by replacing each training vector by its closest codebook vector.

1. Morphological dynamic link architecture (MDLA).

2. Profile shape matching (PSM).

3. Maximum signal processing (MSP).

Using the fuzzy data k-means (FDKM) decision function, the results (for the above three modalities) were a FAR of 0.0039 (0.39%) and a FRR of 0.0. These results demonstrate an excellent FRR score, with a less than satisfactory FAR.

Ross et al., (2001) experimented using three modalities:

1. Face Recognition. As with Jain et al., (1999a), feature extraction involved the use of the eigenface approach to obtain a feature vector of a facial image. Matching involved computing the Euclidean distance between the eigenface coefficients in a template feature vector with those calculated for the query feature vector.

2. Fingerprint Recognition. The features (position and orientation) associated with local fingerprint features were determined from a live-scan fingerprint image. Matching involved comparison of the two-dimensional minutiae patterns in a template with those of a query fingerprint image.

3. Hand Geometry. Fourteen features of a subject's right hand were extracted from a captured image, to obtain a feature vector. The features included the length and widths of fingers, and the width of the palm at various places. Matching involved computing the Euclidean distance between the metrics in the template feature vector with those of the query feature vector.

The matching modules corresponding to each of the modalities returned their own confidence score for each comparison. Four hundred and fifty samples (9 samples provided by 50 participants) of each of the three biometric characteristics were available for testing. Therefore, 450 genuine scores (true positives) and 22,050 impostor scores (false positives) could be generated for each modality. Confidence scores were mapped to the continuous interval $[0, 100]$.

Data fusion was performed at the confidence score level, using three different fusion methods:

1. Sum Rule: this was simply the weighted average of all individual confidence scores.

2. Decision Tree: generated from training data of genuine and impostor confidence scores, and then tested against independent genuine and impostor confidence scores in the test data.

3. Linear Discriminant Function: analysis of the training sets by this function minimised the inter-class separation. Testing set vectors were then classified using the minimum Mahalanobis distance rule.

The best results were returned when the simple Sum Rule was employed to fuse the confidence scores. With no bias applied to any of the modalities, the experiment achieved a FAR of 0.0003 (0.03%) and a FRR of 0.0178 (1.78%). The results demonstrate similar accuracy to the two previous research efforts, with a good FAR and an acceptable FRR.

Wang et al., (2003) used two biometric characteristics for their experiment. Face recognition and iris recognition were used because this combination allowed for simultaneous acquisition of the required images.

The following is a brief description of the extraction and matching operations for the two modalities used in their experiment:

1. Face Recognition. As with Jain et al., (1999a), feature extraction involved the use of the eigenface approach to obtain a feature vector of a facial image. A scatter matrix $S$—incorporating the mean vector—was calculated for each of the $N$ samples. The principal directions of $S$ are the eigenvectors corresponding to the $M$ largest eigenvalues of $S$. A vector Y was determined for each query facial image X, by projecting X onto the subspace obtained by the principal directions. Matching involved comparison between the feature vectors (from the registered and query samples) to produce a confidence score.

2. Iris Recognition. For feature extraction, the iris location was estimated by two circles, superimposed over the scanned image of the iris. The iris part was 'unwrapped' to form a rectangular region. An algorithm, to characterise the critical points of local variation in the rectangular region, was used to record features. The confidence score between registered and query samples was determined by comparing the features.

For the experiment, 5 samples from 90 participants' facial images[8]—consisting of a reasonable variation of expression, pose, and lighting—were selected. This provided 450 facial images. The same number of iris scan samples were also obtained from 90 participants.

Three different methods were used to fuse and then classify the confidence scores from the individual modalities:

1. Weighted Sum Rule. The individual confidence scores were summed with different weights, and calculated at different thresholds. At these differing thresholds, the Sum Rule calculations (which included the adjusted weights) were used in determining the performance variable (FAR and FRR) values.

2. Fisher Discriminant Analysis. The features from each modality were treated as elements of a vector, and the boundary between genuine and impostor vectors was then used to determine class membership.

3. Radial Based Function Network (RBFN). Again, the features from each modality were treated as elements of a vector, and the RBFN was then used for classification.

Best results were achieved by the RBFN which produced perfect recognition (that is, a FAR of 0.0 and a FRR of 0.0) at various threshold values between 0.2 and 0.8. These were excellent results, and demonstrated the suitability of certain artificial neural networks for classification purposes.

[8]Seventy one of which were taken from public domain face databases.

**Feature Level Data Fusion Issues**

Before discussing some the research efforts that have utilised feature level fusion, the following issues associated with this level of fusion are expounded below (Ross and Govindarajan, 2005):

1. The feature sets corresponding to each of the modalities under consideration may not be compatible for fusion. In the current study, this is the case with feature sets corresponding to keystroke dynamics and fingerprint recognition. As mentioned in section 2.3.2.1, data alignment is typically required for the fusion of feature sets to achieve meaningful representation. In some literature, data alignment is referred to as feature normalisation.

2. The correspondence among the different feature spaces (for each of the modalities) may be unknown. If an unknown correspondence between feature spaces exists, this may impact on data fusion design decisions. As an example, in Chapter 5 section 5.3 the point is made that there was presumed to be no relationship between an individual's typing behaviour and their fingerprint characteristics. With no perceived correspondence between their feature spaces, it was deemed unnecessary for data from the two types of data files to belong to the same participant.

3. As previously mentioned, most efforts in the field of feature level fusion simply concatenate the feature sets from the different modalities. This very often results in a very large dimensional feature vector; the resultant feature vector may contain noisy or redundant data, which may have very little or no significance to the pattern recognition task. This is commonly referred to as the 'curse of dimensionality' problem. As discussed in section 2.3.2.1, feature selection may help to alleviate or mitigate this problem to some degree by reducing data dimensionality.

4. A substantially more complex matching process will typically be required to utilise the features in the concatenated feature vector.

**Feature Level Data Fusion Research**

Son and Lee (2005) used two biometric characteristics (facial recognition and iris recognition) for their experiment. The reason given for the choice of these two biometric characteristics, was that they are both attainable from facial images. Wavelet transform was used to extract features from samples of both biometric characteristics.

The authors performed data fusion at the feature level by concatenated the feature sets from both sources. However, they noted that the extracted features demonstrated high dimensionality and thus resolved to attempt to reduce the size of the combined feature vector. For this process, the Direct Linear Discriminant Analysis (DLDA) technique was applied to attain what the authors named a 'Reduced Joint Feature Vector' (RJFV).

Typically, Linear Discriminant Analysis (LDA) methods first use Principle Component Analysis (PCA)[9] to lower the dimensionality of data, and then LDA[10] to further reduce dimensionality (Son and Lee, 2005). However, PCA will often remove components that may be useful for discriminating between classes. The DLDA method reduces dimensionality, by directly removing components that are less likely to be useful in discriminating between classes and maintaining components that are most likely to be useful in discriminating between classes.

As the end product of the DLDA was a RJFV (consisting of only useful information), the methodology achieved the same goals as a feature selection approach. Thus, the paradigm can be considered to be cooperative rather than complementary.

For the experiment, 10 samples each from 140 individuals were selected from two facial image databases (400 samples from one database and 1000 from the other).

[9]The central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set, while retaining as much as possible of the variation present in the data set. This is achieved by transforming the original data set to a new set of orthogonal variables; the principal components (PCs) (Jolliffe, 2002). The goal of PCA is to extract the important information from the data set, to represent it as a set of principal components, and to determine the pattern of similarity (Abdi and Williams, 2010). It is reasonable to expect that a relatively small number of features are sufficient to characterise a pattern or class (Swets and Weng, 1996).

[10]Although PCA is well-suited to object representation, the features produced are not necessarily good for discriminating among classes defined by the set of samples. PCA describes some major variations in the class, however these variations may well be irrelevant to how the classes are divided. Linear Discriminant Analysis is a method for providing clear linear separation of features that optimally discriminate among the classes represented in the data set (Swets and Weng, 1996).

Two iris image databases were compiled; one containing 1000 images (10 samples each from 100 individuals) of high quality, and the other containing 1000 images (10 samples each from 100 individuals) of low quality.

The best results were achieved when the high quality iris images were fused with the facial images from the database that contributed 400 samples. From the ROC curves presented by Son and Lee (2005) in their publication (Figure 5), a FAR and a FRR of 0.0 were achieved. Other very good results were also in evidence.

This research effort confirmed that feature level fusion of data from multiple sources can return accurate classification performance. It also demonstrated that feature selection not only improved efficiency (by reducing the dimension of combined feature vectors), but further improved accuracy by ensuring that the selected features provided the most meaningful discrimination between classes.

Ross and Govindarajan (2005) used two biometric characteristics (facial recognition and hand geometry) for their experiment. Feature extraction for facial images was performed by Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) on the R, G, B channels, and the I gray-scale rendition, of the colour image. The resultant coefficients consisted of 27 features. The hand geometry feature vectors consisted of a 9-byte value comprising different geometric measurements.

The feature vectors for both modalities were normalised (aligned) so that their respective components could equally contribute to the final matching. A median normalisation scheme was used to align the feature vectors, because this scheme was considered non-sensitive to outlier values. Let $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ be the feature vectors of the two modalities. Each $x_i$ and $y_i$ value was normalised according to Equation 2.1.

$$\xi' = \frac{\xi - median(F_\xi)}{median(|(\xi - median(F_\xi))|)} \tag{2.1}$$

where $\xi$ is the $i^{th}$ feature value being normalised, $F_\xi$ is the function that generated $\xi$, and $\xi'$ is the normalised value of $\xi$.

Thus after normalisation, the feature vectors corresponding to the two modalities are denoted $X' = \{x'_1, x'_2, \ldots, x'_m\}$ and $Y' = \{y'_1, y'_2, \ldots, y'_n\}$, and a merged feature vector—from these two sources—can be denoted $Z' = \{x'_1, x'_2, \ldots, x'_m, y'_1, y'_2, \ldots, y'_n\}$.

To reduce the influence of the curse of dimensionality (and to improve classification performance), feature selection was performed such that a minimal feature set, of size $k, k < (m + n)$, was obtained. The sequential forward floating selection technique was employed to obtain a reduced feature vector $Z = \{z_1, z_2, \ldots, z_k\}$. Thus, this research adopted a cooperative data fusion approach.

For matching, two instances of feature vectors $Z_i$ and $Z_j$ (derived from $\{X_i, Y_i\}$ and $\{X_j, Y_j\}$) were compared to obtain the normalised match (Euclidean distance) scores $s_X$ and $s_Y$. Using the simple sum rule, the fused match score ($s_{match} = (s_X + s_Y)/2$) was obtained.

Also, two different distance measures were calculated from the two feature vectors $Z_i$ and $Z_j$, according to Equations 2.2 and 2.3:

$$s_{euc} = \Sigma_{r=1}^{k}(z_{i,r} - z_{j,r})^2 \tag{2.2}$$

$$s_{tad} = \Sigma_{r=1}^{k} I(|z_{i,r} - z_{j,r}, t|) \tag{2.3}$$

where $s_{euc}$ is a Euclidean distance measure, $s_{tad}$ is a threshold absolute distance measure, and $t$ is a pre-specified threshold. $I(y, t) = 1$, if $y > t$, otherwise $I(y, t) = 0$.

Using the simple sum rule again, $s_{euc}$ and $s_{tad}$ were consolidated into one feature level score $s_{feat}$. Again using the simple sum rule, $s_{match}$ and $s_{feat}$ were combined to obtain a final match score. This methodology appears to be a combination of feature level and confidence score level fusion. For this review (as denoted in Table 2.2), the authors designation was noted. For the experiment, 5 facial image samples and 5 hand geometry samples were collected from 100 participants.

The best results, derived from the authors ROC graph for the two fused modalities, were a FAR of 0.0001 (0.01%) and a FRR of 0.13 (13%)[11]. The FAR demonstrates excellent accuracy, however the FRR may be considered less than satisfactory.

[11] Refer Ross and Govindarajan (2005), page 202 Figure 6.

As face and fingerprint recognition are now being used for biometric identification traits by US Immigration and the European electronic passport, Rattani et al., (2007) proposed a method for feature level fusion of these two characteristics. The common level of fusion in existing systems (for these two characteristics) is based on the confidence score.

The following is a brief description of the extraction operations for the two modalities used in their experiment:

1. Face Recognition. Facial features were extracted using the Scale Invariant Feature Transform (SIFT) technique. SIFT is an image descriptor that provides a compact representation of the local gray-scale structure (of facial features), which are invariant to image scaling, translation, and rotation[12]. A set of SIFT features $(s_1, s_2, \ldots, s_m)$ were extracted, where each feature consists of the $(x, y)$ spatial location, the local orientation $\theta$, and the key point descriptor $k$ of size $1x128$. That is, each $s_i = (x, y, \theta, k)$.

2. Fingerprint Recognition. The position and orientation of local fingerprint features (minutiae) were determined for each fingerprint image. The extraction process involved normalisation, Gabor filter pre-processing, binarisation, ridge map thinning, and minutiae extraction (these techniques are discussed in Chapter 4 sections 4.4.3 and 4.4.4). A set of local features $(m_1, m_2, \ldots, m_n)$ were extracted, where each feature consists of the $(x, y)$ spatial location, the local orientation $\theta$. That is, each $m_i = (x, y, \theta)$.

In order to align the feature sets corresponding to the two different modalities, a key point descriptor was defined for each minutiae extracted from a fingerprint image. The local region around each minutia was convolved using Gabor filters to produce a key point descriptor of size $1x128$. The key point descriptors—from each feature, from each modality—were then normalised using the min-max technique, thus reducing their ranges to the interval $[0, 1]$.

The feature sets were then concatenated to give the combined feature vector $(s_{1norm}, s_{2norm}, \ldots, s_{mnorm}, m_{1norm}, m_{2norm}, \ldots, m_{nnorm})$. Three feature reduc-

[12]Illumination changes are partially invariant.

tion strategies ('k-means clustering', 'neighbourhood elimination', and '3 points in a specific region') were applied. The aim of k-means clustering and neighbourhood elimination strategies was to remove redundant data (using Euclidean distance measures) that were very near specific features. The aim of the third strategy was to consider only those points in highly distinctive regions.

Matching involved the comparison of a query feature set and a template feature set (both constructed in the same manner). Two different fusion classifiers were used:

1. Point Pattern Matching. Here correspondences between points (in the feature sets of both modalities) were determined based on spatial distance, direction difference, and the Euclidean distance between the corresponding key point descriptors (each within a pre-determined threshold). A final matching probability was calculated as the ratio of the number of correspondences to the total number of feature points in both feature sets.

2. Delauney Triangulation. Here, a triplet of feature points were grouped into a new feature. A Voronoi diagram isolated regions around each feature point, and delauney triangulation connected the centres of every pair of neighbouring Voronoi regions; a feature vector, comprising information derived from the delauney triangulation, was thus obtained. The final score was based on the number of corresponding triangles that both query feature set and registered feature set had in common.

For the experiment, 5 facial images and 5 fingerprint scans were collected, from 50 individuals. The facial images were obtained from the Biometric Access Control for Networked and E-Commerce Applications (BANCA) database, and fingerprint scans were collected by the authors.

Only 1 of the 5 samples per participant—for a facial image/fingerprint pair—was used as a template for training the fusion classifier. The 4 remaining sample pairs per participant were used for testing purposes. This involved comparing a participants 4 query samples against the database of templates. Therefore, there were 200 (50x4) genuine test scores and 2450 (50x49) impostor test scores generated.

The best results, with a FAR of 0.0102 (1.02%) and a FRR of 0.0195 (1.95%), were achieved using the Delauney triangulation classifier, and the '3 points in a specific region' feature reduction strategy. The best results achieved using the point pattern matching classifier and the k-means clustering feature reduction strategy, were a FAR of 0.0318 (3.18%) and a FRR of 0.0198 (1.98%). Results presented by the authors also demonstrated that feature level fusion out-performed confidence score level fusion in all tests.

For their experiment, Yao et al., (2007) used two biometric characteristics; facial recognition and palmprint recognition.

Feature extraction involved the application of two different transform techniques:

1. Circular Gabor Filter Transform. The transform, which produces four possible scales and eight possible orientations, was calculated by Equation 2.4.

$$G(x, y, \theta, u, \sigma) = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2 + y^2}{\sigma^2}) \times exp(2\pi i(ux\ cos\theta + uy\ sin\theta)) \quad (2.4)$$

   where $\sigma \in \{2, 4, 8, 16\}$, $u = 1/\sigma$, and $\theta \in \{0, 1, 2, 3, 4, 5, 6, 7\} \times \pi/8$.

2. Principal Component Analysis (PCA) Transform. Also called the Karhunen-Loeve (KL) Transform. This technique de-correlates the components or features of a data source and redistributes the energy contained in the feature set so that most of the energy is contained in a smaller number of features. After PCA transform, the features are completely de-correlated, and the energy contained in the feature set is maximally concentrated in a small number of features.

Firstly, the Gabor transform was applied to a facial image and a palmprint image to produce $X_{face}$ and $X_{palm}$ respectively. Then the PCA transform was applied to the respective outputs, to extract the more discriminant features (and thus reduce dimensionality). That is, $Y_{face} = PCA(X_{face})$ and $Y_{palm} = PCA(X_{palm})$.

Normalisation was then applied according to Equation 2.5:

$$y_{norm-face} = \frac{y_{face} - \mu_{face}}{\sigma_{face}} \tag{2.5}$$

where $y_{face}$ was a single sample of $Y_{face}$, $\mu_{face}$ and $\sigma_{face}$ were the mean and variance of the training sample set of $Y_{face}$, and $y_{norm-face}$ was the resultant normalised facial feature vector. The same process was applied to each single sample $y_{palm}$ of $Y_{palm}$ to obtain the normalised palmprint feature vector $y_{norm-palm}$.

A distance-based separability value—based on class membership according to the Nearest Neighbour (NN) classifier—was calculated for each $y_{norm-face}$ and $y_{norm-palm}$. These were denoted $S_{face}$ and $S_{palm}$ respectively, and were used in the determination of a weight vector $[w_1, w_2, \ldots, w_M]$, where $M$ was the number of test samples. The average weighting value was then calculated as $\overline{w} = \Sigma_{j=1}^{M} w_j / M$.

Fusion was then simply performed according to Equation 2.6:

$$y_{fuse} = [y_{norm-face},\ \overline{w} \times y_{norm-palm}] \tag{2.6}$$

The Nearest Neighbour classifier was used to classify $Y_{fuse}$ (that is, each $y_{fuse}$), by the Euclidean distance between each feature in the registered sample and query samples.

The experiment used 20 facial images from 119 participants obtained from the AR Face Database[13], and 20 palmprint images from 119 participants obtained from the Hong Kong Polytechnic University 2D_3D_Palmprint Database[14].

Only one sample (per participant) was used for training purposes (providing 119 fused training vectors); there were 2,261 (119x19) samples (per participant) used for testing purposes.

Results are presented using a performance variable called the recognition rate, and the feature level fusion achieved a rate of 90.73%. Unfortunately, the calculation of the performance variable was not specified, and no FAR or FRR figures were provided.

[13]Information about this database is available from Ohio State University—http://www.ece.osu.edu/ãleix/ARdatabase.html.

[14]Information about this database is available from Hong Kong Polytechnic University—http://www4.comp.polyu.edu.hk/b̃iometrics/2D_3D_Palmprint.htm.

Though this paper did not provide values for the typical performance variables—so that results could be compared with those of the current study—it was included in the review because fused vectors of feature level data were used (for classification) when testing the fusion method. The method used by Ross and Govindarajan (2005), processed feature data into a single scalar score that was alluded to be representative of a fused feature vector. The approach taken by Yao et al., (2007), to perform feature level fusion based on actual feature data, seems a more appropriate method for feature level fusion than one based on a single score derived from feature data.

Nandakumar (2008) investigated a number of different data fusion strategies for verification purposes, as part of a PhD dissertation. The following discussion concentrates on those data fusion strategies most closely associated with the current study.

Firstly, data fusion was performed at the confidence score level. Like the first four studies reviewed (that is, those by Jain et al., (1999a), Chatzis et al., (1999), Ross et al., (2001), Wang et al., (2003)), the experiment by Nandakumar (2008) could be considered as adopting a complementary approach to data fusion (because all confidence scores were fused and all features were utilised to attain those confidence scores).

The work presented a statistical framework to achieve the fusion of confidence scores, which required explicit estimation of genuine and impostor score densities. The well known Gaussian Mixture Models (GMM) were employed to estimate score densities, because the estimates obtained from such models have been shown to converge to the true density. Other density estimate models were investigated, however the GMM returned best results. Fusion of confidence scores was achieved using the complete likelihood ratio test.

For the experiment, two multi-modal databases were utilised; the NIST Biometric Scores Set - Release I (NIST-BSSR1) and the XM2VTS-Benchmark database. The NIST-BSSR1 has three partitions. Partition 1 is a multi-modal database, which consists of 517 users with two fingerprint and two face scores. Partition 2 is a finger-

print database and consists of scores from left and right index fingerprint matches of 6,000 individuals. Partition 3 is a facial image database, which consists of scores from two face matchers applied on three frontal facial images from 3,000 individuals (National Institute of Standards and Technology, 2010b). The XM2VTS-Benchmark database consists of the scores from five face matchers and three speech matchers obtained from four recordings of 295 subjects (The XM2VTS Database, 2010).

Best results were achieved using the GMM and the complete likelihood ratio fusion, tested on the NIST-BSSR1 multi-modal database (Partition 1); a FAR of 0.0001 (0.01%) and a FRR of 0.009 (0.9%). These results are at the very least comparable to the other reviewed research efforts that have fused data at the confidence score level. Also, as the results were tested on benchmark databases of substantial sample sizes, the methodology can be considered to have demonstrated excellent accuracy on a much larger population than the other experiments.

The issues involved in relation to feature level data fusion, as discussed in this section (2.3), were taken into consideration for the experiment in the current study. Chapter 5 section 5.6 describes the implementation of data fusion in this study. The results achieved when data fusion was implemented are presented in Chapter 6 section 6.4.3. A discussion of these results, when compared to the research efforts covered in this section (2.3), is provided in Chapter 7 section 7.2.3.

The next section 2.4 discusses pattern recognition and artificial neural networks.

## 2.4 Pattern Recognition And Artificial Neural Networks

This section firstly provides an overview of pattern recognition in section 2.4.1, and then looks at a number of different classification schemes in section 2.4.1.1.

Section 2.4.2 provides a discussion on Artificial Neural Networks. In section 2.4.2.1, the concept of ANNs simulating the operations of the human brain is explored and how they can be modeled to be useful for computerised tasks. An explanation of some of the different ANN architectures is presented in section 2.4.2.2;

of particular relevance to the experiment in this dissertation are the single and multi-layer perceptrons discussed in sections 2.4.2.2 and 2.4.2.2 respectively.

Section 2.4.3 discusses the properties of the multi-layer perceptron, which has led to this neural network gaining wide acceptance as an accurate pattern classifier.

### 2.4.1 Pattern Recognition

From birth, human beings and other animals utilise their inherent object recognition abilities for survival in their environment (Marques de Sa, 2001). The objects of concern may be physical or conceptual. The recognition task may be simple or complex. Whatever the case, our ability to develop object recognition skills is crucial to learning and progressing, and thus surviving.

The skills to perform object recognition are developed from experience (Marques de Sa, 2001). For example, when we first encounter an object, we observe and identify certain information about that object and store it in our memory. When next we encounter that object, we again identify information about the object and compare that information to our memory of the object.

An example would be recognising another person. When we encounter someone for the first time, we identify certain characteristics about that person and store them in our memory. When next we encounter that person, we again identify the characteristics and compare them to those in our memory.

Pattern recognition is analogous to object recognition, however it refers more specifically to those tasks performed by automated systems to imitate or simulate the human ability for object recognition (Marques de Sa, 2001).

Pattern recognition had its origins in theoretical research—in the area of statistics—prior to the 1960's, but with the advent of the computer age has led to the development of practical automated methods for recognising patterns in many different applications areas (Theodoridis and Koutroumbas, 2006). Some of the application areas for this scientific endeavour are machine intelligence or learning, machine vision, character recognition, computer-aided diagnosis, and speech recognition.

In order to observe and subsequently recognise an object, that object needs to be accurately described (Friedman and Kandel, 1999). This description of an object is referred to as a pattern—that is, a pattern describes an object.

Pattern recognition is a discipline whose goal is the classification of patterns into different categories or classes (Theodoridis and Koutroumbas, 2006). That is, it is concerned with discriminating between different populations of patterns (Friedman and Kandel, 1999). In a practical sense, it involves assigning an query input pattern to one of a finite number of $M$ known or exemplar patterns (Lippmann, 1989).

For example, there may be four different types of road vehicles—cars, buses, trucks, motorcycles. In this case, there are four categories or classes to which a road vehicle can belong. Given an input or query pattern, allocation to a class is often referred to as classification. The decision to allocate a particular pattern to a particular class is based on the features that are used to describe the object (and thus obtain the pattern) and the features that specify or characterise each class (Friedman and Kandel, 1999).

Different classes may be described by some common features, though the values of these features should differ enough to cause an input pattern to be allocated membership to one specific class. For example, all human beings have weight and height features and the values for these features could be used to formulate different classes; tall and light, tall and heavy, short and light, short and heavy.

Feature selection involves determining the features to use when describing an object—to form a pattern—and also to use for class differentiation. More precisely, identifying the features that make patterns distinct, and the measurable attributes that make the distinction apparent for the defining of classes or categories to which a pattern may belong (Theodoridis and Koutroumbas, 2006). The process of obtaining the actual measurements or values for the selected features is called feature extraction.

A pattern recognition system employs the available information—that is, extracted features—to classify patterns or data based either on a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified

are usually groups of measurements or observations, defining points in an appropriate multi-dimensional space.

A typical pattern recognition system should consist of (Wang, 2002):

- A sensor that gathers the observations to be classified or described.
- A scheme that discerns the salient features, defines the appropriate classes, and describes the pattern.
- A feature extraction mechanism that computes numeric or symbolic information from the observed selected features.
- A classification or description scheme that performs the actual task of classifying or describing observations, based on the extracted features.

However, there are practical concerns, that can cause error in a pattern recognition system, that need to be taken into account when designing and implementing the system (Marques de Sa, 2001):

1. The selected features that describe a pattern should be sufficient to truly represent the object being observed.
2. The pattern samples used to define a class should be truly representative of that class.
3. The classification scheme should be effective in separating the classes.
4. The classification scheme should be able to differentiate between classes, given that some classes may contain naturally occurring feature overlap.

The next section discusses classification schemes in more detail.

#### 2.4.1.1 Classification Schemes

A pattern recognition system has as its ultimate goal the correct classification of an input pattern (Friedman and Kandel, 1999). Classification occurs after relevant features (that characterise the pattern) have been identified and extracted. Once

the features have been extracted, they are presented to the classifier to determine if the input pattern is a member of a given class.

A classifier is a 'machine' that is designed to allocate a given input pattern to the most appropriate of the available or known classes (Marques de Sa, 2001). To do this, the classifier applies its computational algorithm to the given features and *indicates* the likelihood of class membership. Note that *indicates* implies that the classifier is making a best estimate, and that estimate may or may not be correct.

The classification scheme is usually based on the availability of a set of patterns and known classes that have been described or categorised by human experts (Marques de Sa, 2001). The set of patterns is termed the training set, and the resultant learning strategy is characterised as supervised learning. Supervised learning provides an association between input data and decision making. Learning can also be unsupervised, where the system is not given an a priori labeling of patterns; instead the system itself establishes the classes based on the statistical regularities of the input patterns.

There are various methods for classification, appropriate to the data being classified and the application to which it is being applied. The classification schemes discussed below assume a priori knowledge of classes (Friedman and Kandel, 1999):

- Decision Functions. This scheme deals with a known number of training patterns that are geometrically separable. Input patterns are classified by decision functions. For example, a simple linear classifier could be defined for classes $C_1$ and $C_2$ by the decision function $d(x)$ according to Equation 2.7:

$$\begin{aligned} d(x) > 0 \quad &\Rightarrow \quad x \in C_1 \\ d(x) < 0 \quad &\Rightarrow \quad x \in C_2 \end{aligned} \tag{2.7}$$

  where $d(x) = 0$ is the 'decision boundary'. If decision boundaries can be defined such that they separate numerous classes in real space, then the classes are said to be linearly separable. Non-linear classifiers exist that use generalised decision functions to distinguish between classes that are not linearly separable.

- Minimum-Distance Classifiers. This scheme usilises distance functions (for classification) when patterns in a training set graduate toward a number of different clusters[15]. If each class consists of a single cluster, a minimum-distance classifier can be used to classify an input pattern. If each class consists of multiple clusters, the nearest-neighbour classifier can be used. Here, the distances from the input pattern to the patterns in the training set are measured, and the input pattern is classified as being a member of the same class as its nearest neighbour (in the training set).

- Statistical (or decision theoretic) Approach. When the patterns of multiple classes exhibit feature overlap, a statistical approach (for classification) can be adopted. Statistical pattern recognition is based on statistical characterisation of patterns. That is, the patterns that originate from statistical distributions. A statistical classifier incorporates the risk or probability of mis-classification.

- Fuzzy Classifiers. The result of classification may not always be certain. That is, there may be doubt about the result, or the input pattern could be classified as a member of more than one class. Feature selection may define an ambiguous metric. For example, a general class definition for height could be 'tall' or 'short'. However, the exact measurement, 173 cm, may be considered to belong to both classes or neither. To alleviate such problems, fuzzy classifiers assume an input pattern to be a member of every class to varying degrees; the grade (strength or weakness) of membership in each class is expressed as a value in the continuous interval $[0, 1]$. By allocating the appropriate grade of membership, classification can be determined.

- Syntactic (or structural) Approach. Syntactical pattern recognition is based on the structural inter-relationships between the features of patterns. Examples of research areas where structural components are used for classification are character recognition and fingerprint recognition. It is common to define a syntax language that describes the structural components of a pattern.

[15] A cluster consists of a number of similar patterns grouped together.

Syntax classifiers are then used to process the resultant symbolic strings representing the pattern.

- Artificial Neural Networks. This approach assumes a set of training patterns and their correct classifications. The correct classifications (for each training pattern) are used as 'targets' when training the neural network. The patterns are supplied to the neural network via the input layer, and the targets are used to guide the learning algorithm of the neural network toward correct classification. Artificial neural networks are discussed in more detail in the next section 2.4.2.

### 2.4.2 Artificial Neural Networks

Artificial Neural Network (ANN) models have numerous names such as: connectionist models, parallel distributed processing models, and neuro-morphic systems (Lippmann, 1987). Regardless of the name, they attempt to achieve high performance processing by utilising dense interconnection of simple computational elements.

ANNs are a technology that is well established in subject areas such as: neuroscience, mathematics and statistics, physics, computer science, and engineering (Haykin, 1999). They have been applied to research and development disciplines such as: mathematical modeling, time-series analysis, pattern recognition, signal processing, and process control.

The next section (2.4.2.1) discusses the biological model of the human brain, the operations which an ANN attempts to model, as well as how ANNs learn. Section 2.4.2.2 provides a discussion of a number of prominent ANN architectures.

Section 2.4.3 explains why the Multi-Layer Perceptron (MLP), as a pattern classifier, is ideally suited to pattern recognition tasks where the data is of a complex nature, such as the individual and combined feature data sets involved in the current study. Tried and tested freeware applications of the MLP are available for ease of implementation. For these reasons, the MLP was used in the current study.

#### 2.4.2.1 Imitating The Biological Model

An ANN attempts to imitate the computational functionality and memory capacity of the human brain (Beale and Jackson, 1990). However, an ANN is not biological. An ANN is a massively parallel distributed processor made up of simple processing units, which have a propensity for storing experiential knowledge and making it available for use (Aleksander and Morton, 1990).

Thus an ANN imitates the brain's capacity to (Aleksander and Morton, 1990):

1. Acquire knowledge from its environment through a learning process.

2. Store the acquired knowledge in synaptic weights, via inter-neuron connection strengths.

The human brain contains approximately $10^{10}$ (i.e. ten thousand million) basic analogue processing units, called neurons (Beale and Jackson, 1990). Each neuron is connected to about $10^4$ (i.e. ten thousand) other neurons. Figure 2.9 illustrates the components of a biological neuron.

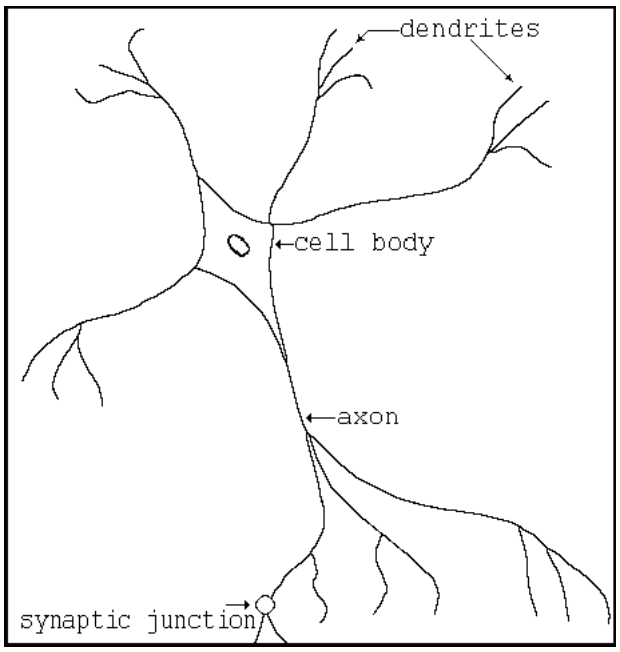


Figure 2.9: Components of a Biological Neuron
The image was sourced from Jian-Kang, 1994.

The basic operation of a neuron is to accept many inputs (in the form of nerve impulses) from dendrites, which are filaments attached to the cell body; these nerve impulses are received from other neurons (Beale and Jackson, 1990). The neuron accepts all inputs, and if the accumulated signal (called the 'resting potential') surpasses a critical threshold, the neuron activates; otherwise it remains inactive.

The axon—a component of the cell body that serves as the output channel—acts as a non-linear threshold device. That is, if a neuron activates (or fires), the axon produces a series of rapid pulses called the 'action potentials'. These pulses are propelled along the axon, which terminates at the junction of dendrites from other neurons; such a junction is called a synaptic junction.

There is no actual connection at this junction, rather a chemical reaction when the synapse's potential is raised sufficiently by the action potential received via the axon. Neurotransmitters released by the synapse allow for the flow of ions across the junction, which alter the dendritic potential and results in a nerve impulse which is conducted along the dendrite to its cell body. Each dendrite may have many synapses acting on it, which facilitates massive inter-connectivity.

Synaptic junctions alter the effectiveness of the transmitted signal. Thus a synapse may 'excite' or 'inhibit' a dendrite. If a dendrite is excited by a synapse, a large nerve impulse traverses the junction and is conveyed by the dendrite to the cell body; if a dendrite is inhibited, a small nerve impulse passes to the cell body. The release of more neurotransmitters increases the coupling at the synaptic junction, which increases the connection strength.

These modifications to the normal connection strength are thought to facilitate learning. Because the neuron receives all inputs, and the signal strength affects the accumulated resting potential, the firing of a neuron (i.e. exceeding the threshold value) is influenced by the strength of incoming signals.

The parallel nature of the brain's functionality means that the processing workload is distributed across many neurons (Lippmann, 1987). If one neuron malfunctions, its affect is unlikely to have a significant impact on the overall result. That is, the biological brain is generally fault tolerant.

ANNs attempt to model the operations of the brain, and consist of (Lippmann, 1987; Beale and Jackson, 1990; Haykin, 1999):

1. Simple processing units or elements called nodes (because of their correspondence to the biological neuron, they are often referred to as neurons).

2. An inter-connection topology in the form of weights.

3. A learning scheme.

Thus, ANNs do not attempt to represent each component of the biological neuron, only the functionality of the biological neuron. As the interaction between biological neurons occur at the synaptic junctions, it is the operations of the synaptic junction that ANNs attempt to model.

The model of a simple neuron consists of three basic components (Haykin, 1999):

1. A set of weights expressing the connection strength between the input signals to the neuron; this is analogous to the synapses of a biological neuron. For an input signal $x_i$ connected to a neuron $k$, $i$ is the connecting synapse and the weights are denoted $w_{ki}$[16].

2. A summation function which accumulates the input signals, by multiplying each input signal by the weight connecting it to the neuron.

3. An activation function for limiting the amplitude of the neuron's output. For two class problems, binary output is typically 0 or 1. For multiple class problems, the output range is typically in the closed continuous interval $[0, 1]$ or $[-1, 1]$.

Figure 2.10 illustrates the components of a simplified non-linear model of a neuron $k$. Here, the value for each input node is multiplied by its associated weight[17].

[16]Note that this common naming convention has the variable designating the neuron listed prior to the variable designating the connecting weights.

[17]Note that because weights are analogous to the signal strength received by dendrites from synapses in the biological model neuron, larger weight values correspond to excitatory synapses (which transmit larger pulse signals across the synaptic junction); smaller weight values correspond to inhibitory synapses.

The sum of this accumulated resting potential—plus a bias component[18] $b_k$—is then applied to an internal threshold in the activation function. When the resting potential exceeds the threshold, the neuron activates; otherwise, it remains dormant.

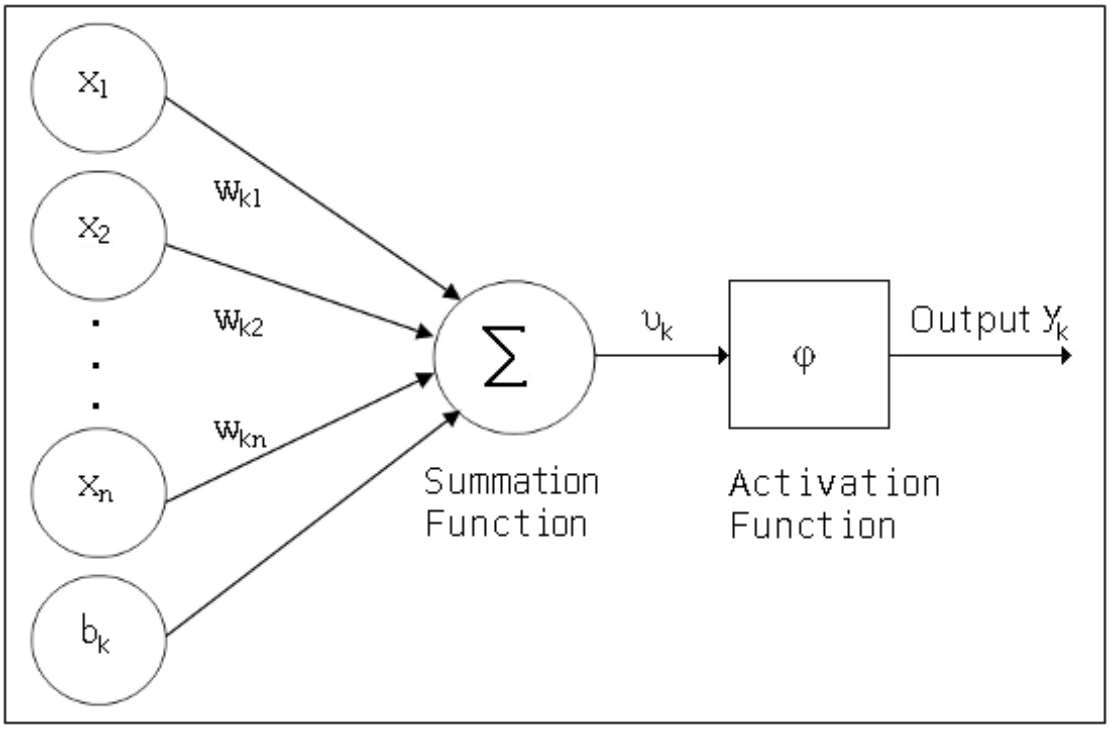


Figure 2.10: Simple Non-linear Model of a Neuron
The image was sourced from Haykin, 1999.

Equations 2.8 and 2.9 describe the mathematical operations of the simplified non-linear neuron model illustrated in Figure 2.10:

$$\upsilon_k = \sum_{i=1}^{n} w_{ki} x_i + b_k \tag{2.8}$$

$$y_k = \varphi(\upsilon_k) \tag{2.9}$$

where $x_1, x_2, \ldots, x_n$ are the input signals, $w_{k1}, w_{k2}, \ldots, w_{kn}$ are the synaptic weights of neuron $k$, $b_k$ is the bias, $\upsilon_k$ is the induced activation potential, $\varphi$ is the activation function, and $y_k$ is the output signal of neuron $k$.

Alternatively, the bias can be incorporated into Equation 2.8 by initialising $x_0 = 1$ (and to always retain this value), $w_{k0} = b_k$ and then defining the mathematically equivalent Equation 2.10 (Beale and Jackson, 1990; Haykin, 1999):

[18]The bias effectively adds an offset to the accumulated resting potential, and is intended to increase or decrease the net input into the activation function.

$$\upsilon_k = \sum_{i=0}^{n} w_{ki} x_i \tag{2.10}$$

Note that because the process flow of the model neuron is one way, presenting the inputs and producing the output, it is called a *feedforward* system.

The activation function $\varphi(\upsilon_k)$ determines the eventual output of neuron $k$, and is dependent on the summation function output $\upsilon_k$. Numerous activation or threshold functions may be applied in determining the eventual output. Thus the choice of activation function plays an important role.

Figure 2.11 provides three examples of typical threshold functions that are applied in the activation function (Lippmann, 1987). These are: the step-wise function (a), the piece-wise linear function (b), and the sigmoid function (c). More complex nodes may include temporal integration and other types of time dependencies[19].

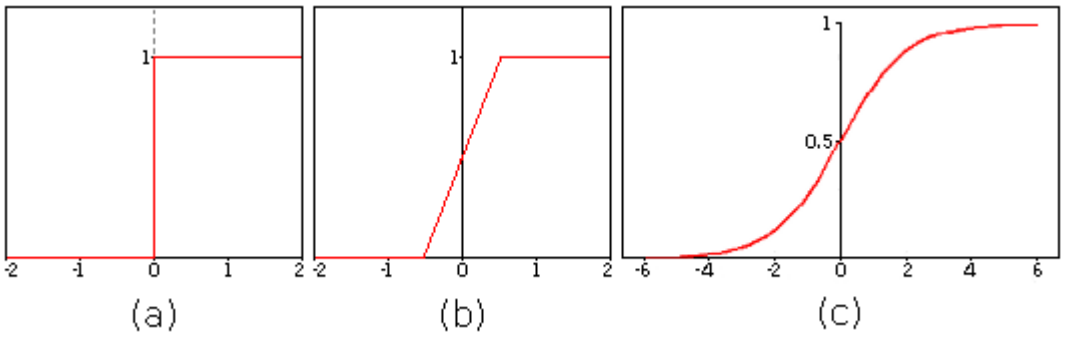


Figure 2.11: Illustrations of Step-wise (a), Piece-wise (b) And Sigmoid (c) Functions

### The Learning Process

Human beings (particularly when young) very often learn from positive or negative reinforcement (Beale and Jackson, 1990). That is, a positive outcome results from 'good' behaviour, while a negative outcome results from 'bad' behaviour. Of course, the interpretation of what is positive and good or negative and bad is subjective to the situation under observation, but in general this process of reinforcement is often helpful in the learning process.

[19]It should be noted, that the step-wise function (Figure 2.11a) is the usual threshold function applied to the simple model of a neuron discussed thus far because the model produces binary output (i.e. 0 or 1).

As with human beings, the simple model neuron can be taught to 'learn' from its mistakes. That is, to reduce the chance of an incorrect or unwanted outcome from occurring.

To demonstrate, assume two classes A and B. The learning process requires the following steps (Beale and Jackson, 1990):

1. Assign random values to all weights from the input nodes to the output node. This corresponds to the state of the neuron knowing nothing.

2. Present an instance of class A input.

3. Perform the actions assigned to the summation and activation functions. If the resting potential exceeds the internal threshold, output 1; otherwise output 0.

4. For inputs of class A, assuming an output of 1 (the correct answer), do nothing; assuming an output of 0 (the incorrect answer), increase the resting potential (by increasing the weight values) so that the threshold is exceeded and the correct output is produced.

5. For inputs of class B, the neuron should be expected to produce an output of 0. When an instance of class B is input, decrease the weight values to keep the resting potential below the threshold.

By adjusting the weight values according to the steps 4 and 5, the neuron learns to recognise instances of class A input, and that instances of class B input are not instances of class A input. So, the ability to learn is directly attributable to the storage and adjustment of weight values.

Thus the following rule[20] can be defined to facilitate learning, by adjusting weight values (Beale and Jackson, 1990):

1. Increase the weights (on active inputs), when active output (i.e. the value 1) is required. This can be achieved by adding the inputs to the existing weight values.

[20]Note that the rule reinforces active connections only.

2. Decrease the weights (on active inputs), when inactive output (i.e. the value 0) is required. This can be achieved by subtracting the inputs from the existing weight values.

This rule presupposes knowledge of the correct class. That is, knowing that the input is an instance of class A, and that class A is the intended class. As the rule guides learning using this knowledge, it is known as 'supervised learning'.

When adopting a supervised learning approach, instances of the correct class are presented to a neuron along with their expected output (i.e. the value 1) (Lippmann, 1987). Also, instances of the incorrect class are presented to the neuron along with their expected output (i.e. either 0 or -1). The expected output is commonly referred to as the 'target'.

By presenting a neuron with instances of both classes and their respective target outputs, the neuron learns to correctly classify the intended class. The presentation of incorrect class instances helps the neuron to learn—by adjusting weights accordingly—not to attribute correct classification to incorrect class instances. That is, the neuron learns what to classify correctly according to examples of what is correct, and what is not correct (positive and negative reinforcement).

In 1962 Frank Rosenblatt coined the term 'perceptron' to describe a feedforward ANN composed of one or more simple output neurons that function as discussed thus far (Beale and Jackson, 1990). This perceptron is the simplest kind of ANN, and can be considered a binary classifier, because each neuron outputs either a 1 or a 0; it is also often referred to as a linear classifier, because it classifies two classes according to the hyperplane that separates the two classes (called the decision boundary).

A perceptron may consist of a single layer; that is, one or more simple output neurons or nodes making up the output layer[21]. A perceptron may also consist of multiple layers; that is, one or more simple output nodes making up the output layer, with one or more intermediate layers (consisting of one or more nodes) between the input neurons and the output layer. In such a case, all nodes from all layers have full inter-connectivity.

[21]Note that this description does not describe the inputs as a layer.

### ANN Training

In the training process, inputs are presented to a processing element (node) which apply a summation function to inter-connections (weights) that act upon the inputs. As inter-connections can be excitatory or inhibitory, the weights can be of larger or smaller magnitude (which add to or subtract from the accumulated value). When a specific threshold is reached, the node 'fires' according to an activation function, which produces the resultant output.

The result of an activation function can be in the binary domain (that is, discrete values $(0, 1)$[22]) or the continuous domain (that is, over the continuous interval $[0, 1]$[23]). Discrete value results are typically classified using the sum-and-threshold model (for example, a step-wise function—refer Figure 2.11a—or a piece-wise linear function—refer Figure 2.11b) (Bishop, 1995). This seems a natural choice because the optimal discriminant function needs to distinguish two classes. Continuous value results are typically classified using an exponential or logarithmic model (for example, the logistic sigmoid function—refer Figure 2.11c).

The output of the activation function is compared with the expected or target output. The difference between the activation function output and the target is calculated; this difference is commonly referred to as the error. The error is added to each input and the corresponding weights are adjusted (refer Equation 2.12). This training process continues until error is minimised. The weights are then stored and used in the testing process.

The number of different outcome classes defines the problem space. If there are two possible outcome classes, there exists a linearly separable problem space (Masters, 1993). That is, the outcome can only be one class or the other. The boundary between outcome classes may be linear or non-linear. This boundary is called the decision boundary, and distinguishes between classes in the problem space. It also determines the number of output layer nodes the network requires[24].

[22] Note, the discrete values $(-1, 1)$ are sometimes used.
[23] Note, the continuous interval $[-1, 1]$ is also commonly used.
[24] That is, each output layer node represents a possible outcome class.

The behaviour of a neural network, as it attempts to arrive at a solution, can be thought of in terms of the error or energy function. The energy is a function of the inputs and the weights. For a given input pattern, the energy function can be plotted against the weights to determine the energy surface in a three dimensional space. This can be envisaged as a landscape of hills and valleys, with points of minimum energy (known as wells) and maximum energy (known as peaks).

If the problem space has more than two possible outcome classes, the error can be minimised by adjusting weights so they correspond to points of lowest energy (that is, minimised error). The wells in the error surface may be many, but there will be one that is deeper than any other. This is the global minimum, and corresponds to the lowest possible error that can be attained for that input pattern. The other wells are local minima.

The objective in training the network is for it to reach the global minimum error; this means that it has trained to most accurately classify that class of pattern. The weights of the network in this state (that is, when the training objective has been achieved) are stored for the testing procedure.

### ANN Testing

The testing procedure involves constructing an ANN of the same configuration as that used during training, and loading the stored weights (resulting from the training process) into this ANN. Test input patterns are then presented to the ANN. It is preferable (so that the trained ANN can generalise) that the input patterns presented for testing have not been used in the training process (that is, different samples of the same class of input patterns). For the testing procedure, no target outcomes are provided (Lippmann, 1987).

The summation and activation functions are applied (as previously explained) and the outputs are produced by the ANN. The type of output is dependent on the number of decision boundaries in the problem space, which determines the type of activation function used and therefore the nature of the output data (that is, binary or continuous).

It is important to remember that the ANN output resulting from the testing procedure may or may not be correct. That is, ANNs can, and do, attribute correct and incorrect class membership. Output in the continuous domain is typically subject to a final classification scheme, where a decision threshold is applied (refer Chapter 6 section 6.2).

If the nature of the training and testing data exhibits consistency *and* the ANN has been well trained, classification should be accurate. However, if the nature of the training and testing data exhibits variability *or* the ANN has not been well trained, the classification may not be as accurate.

For the current study, ANNs were used to classify query inputs as belonging to (or not) the training group members' classes, according to the training group members' registered templates. The registered template for a training group member consisted of the weights of the ANN that had been trained to recognise the pattern of their training data. During testing, query data sets were applied to the ANN (using the stored weights for registered templates) to determine correct classification.

ANNs are generally effective in solving classification and pattern recognition problems (Beale and Jackson, 1990), and as shown in the next section, there are a number of architectural models designed for these types of problems. The architectural model used in the current research was the Multi-Layer Perceptron (with error back propagation), because it was well suited to the complexity of the pattern recognition task of this experiment. An explanation of the operations of the Multi-Layer Perceptron (MLP) is provided in the next section.

#### 2.4.2.2 ANN Architectures

This section describes the architecture (or topology) and properties of some prominent Artificial Neural Networks. Firstly, the Single Layer Perceptron is presented, as it is the simplest ANN architecture and because it is beneficial to understanding the Multi-Layer Perceptron. The Multi-Layer Perceptron—which was used in the current research—is discussed after the Single Layer Perceptron. Following the discussion of these two architectures, some other architectures of interest are described.

### The Single Layer Perceptron

The Single Layer Perceptron (SLP) is a feed forward network, which has the ability to learn to recognise simple patterns (Lippmann, 1987; Beale and Jackson, 1990; Haykin, 1999). It classifies input data sets into one of two classes (such as class A or class B).

The architecture for the SLP consists of an input layer[25], an output layer, and the connecting weights (refer to Figure 2.12). This has a close analogy to the simplified nonlinear neuron model discussed in section 2.4.2.1. In fact, the SLP demonstrates a similar architecture and functionality to the simplified nonlinear neuron model, except that there may be multiple output layer nodes.

The simplified illustration of the SLP, in Figure 2.12, consists of only two output layer nodes; there may be many output layer neurons in a SLP, each of which will be connected to each of the input layer nodes by an associated connecting weight.

Inputs are supplied to the SLP (via input layer nodes); they and their associated weights are applied to the summation function in the output nodes. The summed value has a threshold value subtracted from it, and the result is applied to the activation function (eg: a step-wise function). An example outcome could be, designate class A if the output y was $+1$ or class B if it was $-1$.

The SLP forms two regions separated by a hyperplane (called the decision boundary), such that inputs which classify to class A are located on one side of the linear boundary; class B outputs are located on the opposite side. The equation of the boundary line is dependent on the weights and the threshold value.

To demonstrate the summation and activation functions of the SLP output node, the following description is presented.

Let $w_i$ be the weight corresponding to input $i$, at time $t$, for $(0 \leq i \leq n)$. Set $w_0 = -\theta$, and $x_0$ to always remain equal to 1. Provide inputs $x_1, x_2, \ldots, x_n$, and the desired output (or target) $d(t)$. Initialise all other $w_i(0)$ to small random values.

[25] By convention, the input nodes are not counted as a layer (even though they are presented as such in architectural illustrations). This is because only the output layer has operational nodes.

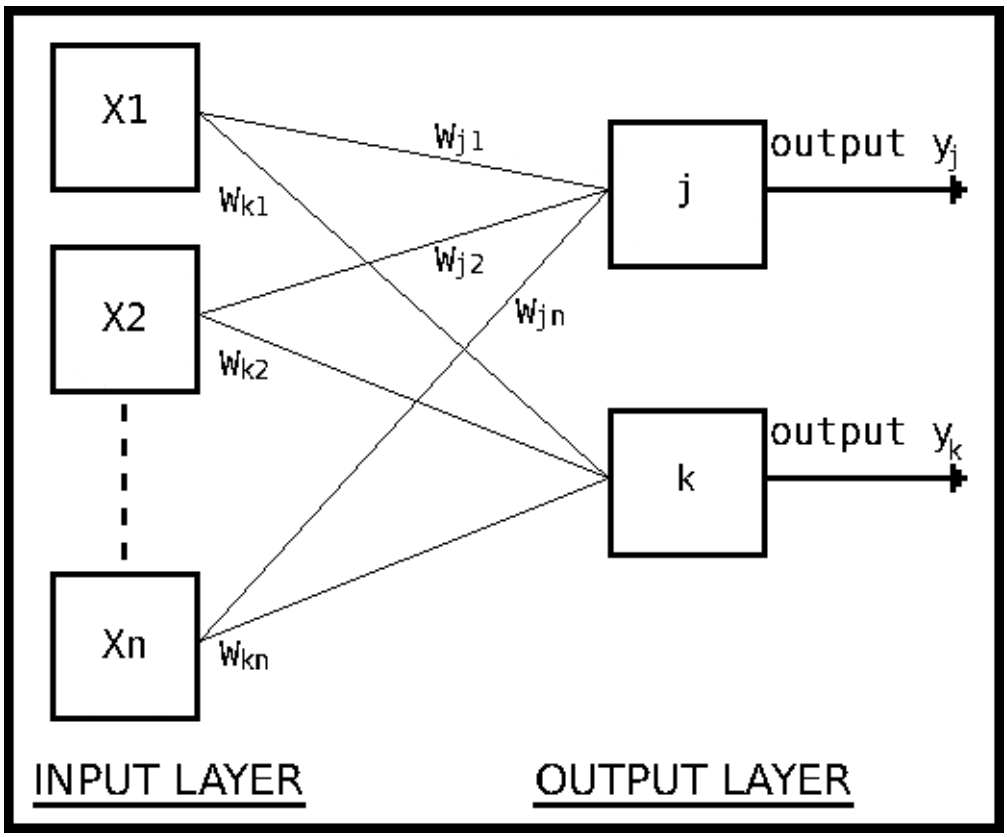


Figure 2.12: The Single Layer Perceptron

The output $y(t)$ is calculated according to Equation 2.11:

$$y(t) = f_h \left[ \sum_{i=0}^{n} x_i(t) w_i(t) \right] \tag{2.11}$$

where $n$ is the number of input layer nodes, $\theta$ is the internal threshold or bias, and $f_h$ is the activation function (eg: step-wise function) used to produce an output.

In order to facilitate learning the SLP repeats Equation 2.11, adjusting all weight values after each repetition. This process continues until the network converges to the minimum error.

Adjusting the weights is accomplished according to Equation 2.12:

$$w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t) \tag{2.12}$$

where $\eta$ is a gain function to control the adaption rate, for $(0 \leq \eta \leq 1)$. The gain term controls the rate of weight change, ensuring the network modifies the weights by a suitable magnitude[26].

[26] Adjusting the gain term by a smaller magnitude, rather than larger, helps to avoid any tendency

The target output $y$ is designated according to Equation 2.13:

$$d(t) = \begin{cases} +1 & \text{if input from class A} \\ 0, & \text{if input from class B} \end{cases} \tag{2.13}$$

A major benefit of this architecture is that the network will resolve to the best possible output (that is, it will always find the global minimum error). However, as demonstrated by Equation 2.13, this architecture is suitable only when there are two classes of possible outcomes (that is, it can only solve linearly separable problems of two outcome classes).

### The Multi-layer Perceptron

The Multi-Layer Perceptron (MLP) is able to classify along $n$ dimensional decision boundaries in the problem space, by utilising a non-linear threshold function and by incorporating extra layers of nodes in its configuration (Lippmann, 1987; Beale and Jackson, 1990; Haykin, 1999). These modifications solves the limitation of the SLP (i.e. of only being able to solve two class problems) and allow the MLP to classify complex data (i.e. data demonstrating multiple classes in multi-dimensional problem space).

Like the SLP, the MLP has a feed forward operation. However, as illustrated in Figure 2.13 the architecture is quite different[27]. There are at least two layers of nodes in a MLP[28]; the input layer, the output layer and one or more hidden layers in between. Just as the output nodes in the SLP function as individual perceptron units (i.e. simple neuron models), so the hidden and output layer nodes of the MLP function as perceptron units. That is, nodes in all layers (excluding input nodes) accept input, apply the input to the summation and activation functions, and produce an output.

---

to oscillate between extreme weight values as the network trains toward minimum error.

[27]Note that unlike in Figure 2.12, Figure 2.13 does not include labels for the weight connections between nodes. This was done to keep the illustration less noisy, and thus make the configuration differences clearer. The weight labeling convention used in Figure 2.12 would be the same for Figure 2.13.

[28]Input nodes perform no computational operations and are not counted as a layer.

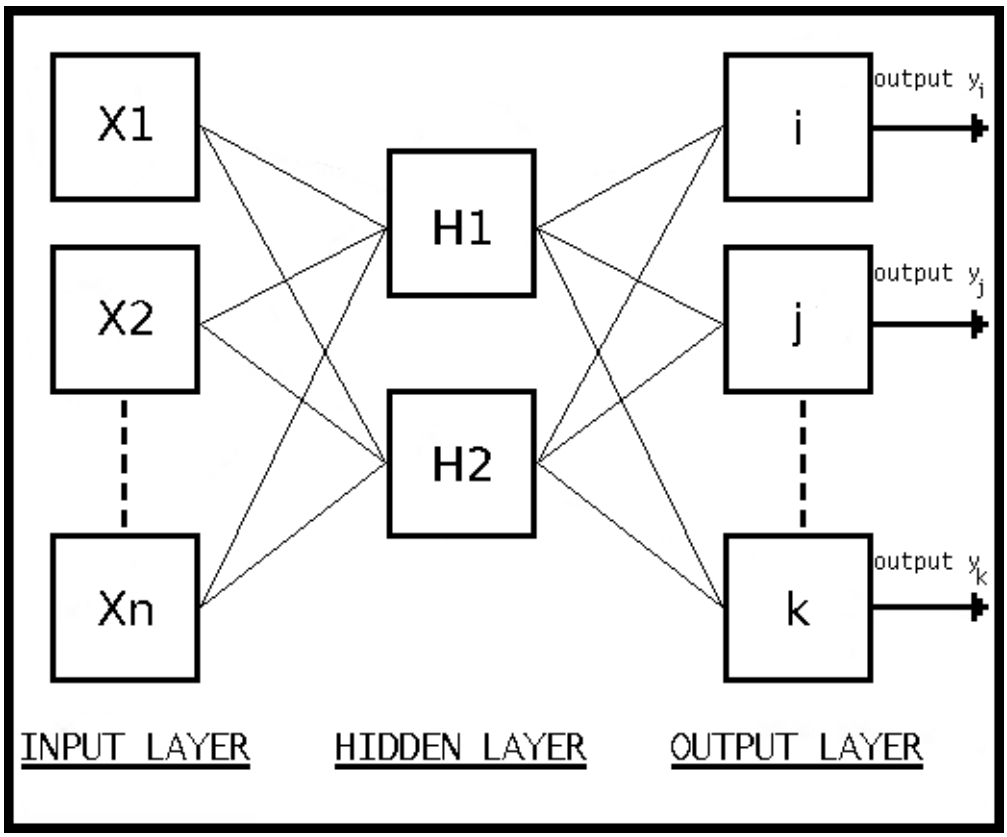


Figure 2.13: The Multi-Layer Perceptron

A notable difference between the MLP and the SLP is that the activation function applied in the MLP reflects the complex nature of the problem space. Because more than two classes are being classified, the step-wise function is no longer appropriate. As such, an exponential or logarithmic model is more appropriate; typically the most commonly applied function is the logistic sigmoid function. Therefore, node outputs are in the continuous domain rather than the binary domain.

As previously discussed, the number of input layer nodes is determined by the input pattern, and the number of output layer nodes must match the number of different classes in the problem space.

However, the number of middle layer nodes has no specific method of determination. A rule of thumb is to assign half the number of input layer nodes as a temporary value for the number of middle layer nodes. Then increment or decrement the number of middle layer nodes and test the error at each adjustment, until the lowest error rate is attained.

Because the MLP exhibits a modified configuration to the SLP, and utilises a different activation function (i.e. sigmoid compared to step-wise), the learning rule of the SLP requires modification for the MLP (Beale and Jackson, 1990; Haykin, 1999). Input is presented to the network; comparison is made between the network output and the desired target; the error is determined and can then be used to update the weights (and produce successively more accurate output).

The type of activation function used in MLPs allows for continual reduction of error values by small increments, and thus the network output gradually approaches the desired target. This is achieved by using the 'generalised delta rule' to calculate the error values for that input (at the output layer), and adjusting weights by back-propagating the error through the network to the previous layers. This functionality is responsible for the MLP being often termed the back-propagation neural network.

Nodes in the hidden layers are adjusted in proportion to the error in the nodes (of the output layer) to which they are connected. So if an output node has a larger error value, the connected hidden layer nodes use a value proportionate to the output layer node (rather than the same error value). This allows the network to learn, as the method of error reduction facilitates correct adjustment of weights between the layers.

To demonstrate the operations of the MLP, the following description is presented (Lippmann, 1987; Beale and Jackson, 1990). Provide input pattern $X_p = x_0, x_1, \ldots, x_{n-1}$; the desired output (or target) $T_p = t_0, t_1, \ldots, t_{m-1}$; and $w_i$ the weight corresponding to input $i$ for $(0 \leq i \leq n)$, where $n$ is the number of input layer nodes and $m$ is the number of output layer nodes.

Set $w_0 = -\theta$, and $x_0$ to always remain equal to 1. Initialise all other $w_i$ to small random values.

Let $y_{pj}$ be the actual output values for pattern $p$ on node $j$, calculated for each node for each layer according to Equation 2.14:

$$y_{pj} = f\left[\sum_{i=0}^{n-1} w_i x_i\right] \tag{2.14}$$

Note $o_{pj}$ denotes the output layer values for pattern $p$ on node $j$.

Adjusting the weights is accomplished according to Equation 2.15, starting from the output layer nodes (and progressively working backward through each layer of the network):

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_{pj} o_{pj} \tag{2.15}$$

where $w_{ij(t)}$ represents the weights from node $i$ to node $j$ at time t, $\eta$ is the gain term, $\delta_{pj}$ is the error term calculated for pattern $p$ on node $j$.

Equation 2.16 defines the error term for the output layer nodes:

$$\delta_{pj} = k o_{pj}(1 - o_{pj})(t_{pj} - o_{pj}) \tag{2.16}$$

Equation 2.17 defines the error term for the hidden layer nodes:

$$\delta_{pj} = k o_{pj}(1 - o_{pj}) \sum_{k} \delta_{pk} w_{jk} \tag{2.17}$$

Note that the first expression $k o_{pj}(1 - o_{pj})$, in equations 2.16 and 2.17, is the derivation of the sigmoid function[29]. Also, the latter expression $(t_{pj} - o_{pj})$, of equation 2.16, incorporates the desired output or target $t_{pj}$, whereas the latter expression $\sum_k \delta_{pk} w_{jk}$, of equation 2.17, incorporates the sum of error terms $\delta_{pk}$. Here $k$ designates all nodes in the layer preceding the layer where node $j$ is situated. Therefore, the sum of $\delta_{pk}$ is calculated from node $j$ to all nodes $k$. Thus the error is back-propagated through the network proportionately.

As discussed in the previous section, the error or energy surface can be determined as a function of the input and weights. As the MLP has more layers of nodes than the SLP, the energy surface becomes more complex and can be populated with numerous local minima; but still only one global minimum.

In order to assist training, so that the minimum error is attained, there are two parameters that may be used in calculations when updating weight values. One, the gain term—discussed previously—is used to speed or slow progress toward a

[29] For the mathematical proof of this derivation, refer to (Beale and Jackson, 1990) Chapter 4 section 4.4.1.

minimum (local or global) error rate. The other parameter is the 'momentum' term, which is used as stimulation of the network during training; to jump out of local minima and eventually (hopefully) settle in the global minimum.

The summation function (equation 2.14) is applied to the inputs and weights (between the input and the first hidden layer of nodes); the node outputs from each successive hidden layer are calculated, and these values become the inputs for the next layer of hidden nodes; this continues until the last layer of hidden nodes pass their outputs as inputs for the output layer nodes.

The error values (which are calculated to incorporate targets and the activation function) are then propagated backward through the network and are used to successively update all weights (according to equations 2.15, 2.16, and 2.17). Once the weights between the input layer and the first layer of hidden nodes have been updated, the feedforward process begins again with the network status altered by the newly updated weights (and possible adjustment to the gain and momentum terms). This process continues until the error rate is minimised.

One negative feature of the MLP is that during the training phase, the error rate will not always easily—if at all—resolve to the global minimum. Quite often the error rate gets ensnared in one of many local minima on the error surface, and requires stimulation by the momentum and gain terms to escape it. This means that training a MLP requires quite a bit of trial and error testing, by manually manipulating the number of middle layer nodes and the momentum and gain terms, in order to reach the optimum efficiency and accuracy.

The next section (2.4.2.2) discusses the architecture and properties of the Hopfield Neural Network.

#### Hopfield Neural Network (HNN)

The Hopfield neural network is an auto-associative network. The auto-association of patterns means that presentation of corrupt or incomplete input will result in the reproduction (as output) of the original pattern (Lippmann, 1987; Beale and Jackson, 1990). The network thus works as a content addressable memory (CAM).

As illustrated in Figure 2.14, the architecture of the HNN consists of a number of nodes (visualised as one layer), each of which are connected to each other node (but not itself). Therefore, the HNN is a fully connected network, with binary inputs and outputs (that is, values of $(0, 1)$ or $(-1, 1)$). Also, the network is symmetrically weighted (that is, any weight $w_{ij} = w_{ji}$).

The difference in architecture between the HNN and perceptrons means the HNN operates in a different way. The network is left to cycle through a succession of states until it converges on a stable solution; this occurs when node values no longer change. The final network output is taken to be the value of all nodes in this final stable state. Because of the fully inter-connective property, the value of one node affects the value of all nodes.

In their initial state each node represents different values (received as inputs), with each node trying to affect the others; thus the network is initially in an unstable state. During operation, some nodes may attempt to turn other nodes 'on', while some other nodes may attempt to turn other nodes 'off'.

As the network progresses through successive states, it works toward a state (by a system of 'compromise') where all nodes settle into a stable state (representing the 'best compromise'). At this point there are as many inputs attempting to turn a node on as there are inputs attempting to turn a node off.

Training involves one iteration per pattern presented to the network. Only the weights are adjusted by calculating the cross product of the input vector. Each successive input vector updates the weight matrix. The top-left to bottom-right diagonal values are set to 0.

Testing involves many iterations. Input is presented to all nodes simultaneously, and the network is left to stabilise. Updating of nodes occurs via weighted sum and a hard limiting step-wise function. Output of each node becomes the input fed back to the other nodes (but not itself). Outputs from the nodes in the stable state form the output of the network. So when presented with an input pattern, the network outputs a stored pattern nearest to that presented input pattern.

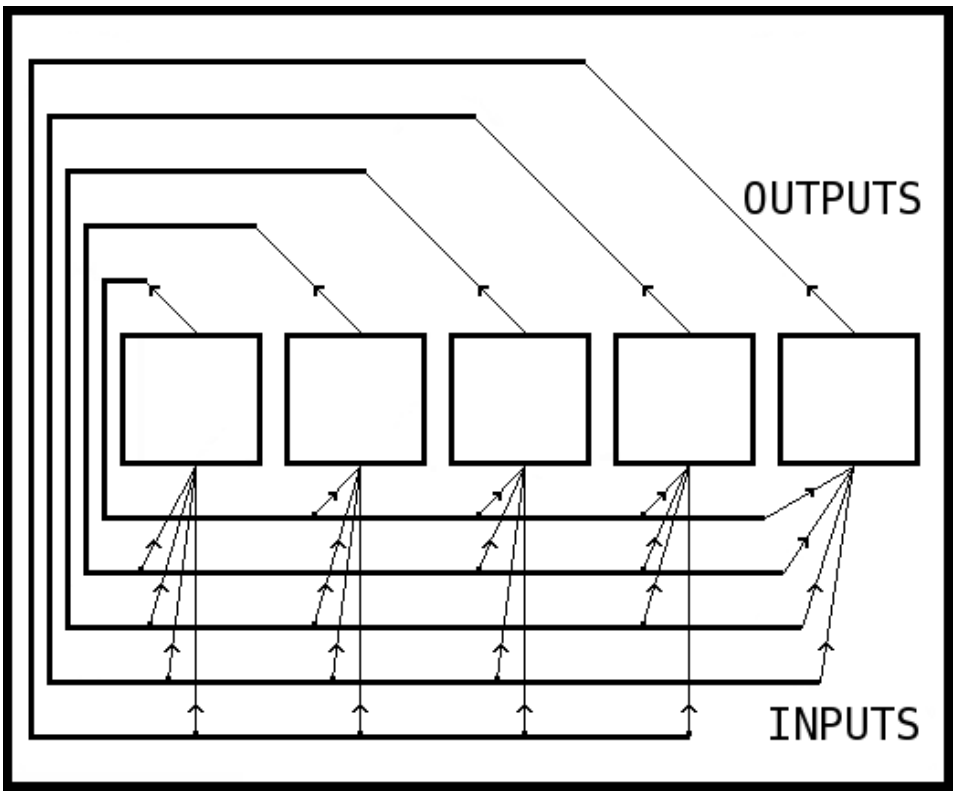


Figure 2.14: The Hopfield Neural Network

The Hopfield network has no learning algorithm as such. Patterns (or facts) are simply stored by setting weights to lower the network energy (or error).

The training stage occurs according to Equation 2.18:

$$w_{ij} = \begin{cases} \sum_{s=0}^{M-1} x_i^s x_j^s & i \neq j \\ 0 & i = j, 0 \leq i, j \leq M-1 \end{cases} \tag{2.18}$$

where $w_{ij}$ is the connection weight between node i and node j, $x_i^s$ is the $i^{th}$ element of the exemplar pattern for class $s$, $M$ is the number of pattern classes.

Note that for each weight, the product of the input $i$ and input $j$ is added to the existing weight. Therefore, the result of the training stage is the association of a pattern with itself.

Initialisation for the testing stage occurs according to Equation 2.19:

$$\mu_i(0) = x_i \quad 0 \leq i \leq N-1 \tag{2.19}$$

where $\mu_i(t)$ is the output of node $i$ at time $t$.

Nodes are updated according to Equation 2.20:

$$x_i = x_i w_{ij} x_j \tag{2.20}$$

where input $x_i$ represents the node being updated, input $x_j$ represents the input into that node, and $w_{ij}$ is the weight connection.

The network is allowed to iterate freely (in discrete time steps) until it converges. Note that the output of the network is forced to match that of the imposed unknown pattern.

Convergence during the testing stage occurs according to Equation 2.21:

$$\mu_i(t+1) = f_h \left[ \sum_{i=0}^{N-1} w_{ij} \mu_i(t) \right] \quad 0 \leq j \leq N-1 \tag{2.21}$$

where $f_h$ is the step-wise function.

If the input $i > 0$, its output is 1; if the input $i < 0$, its output is -1 (or 0); otherwise, the input value is left as it is. These values are then fed back into the network as input into the other network nodes.

The advantage of the HNN, as an auto-associative network, is that data can be retrieved (via its CAM functionality) even when incomplete or corrupt information is presented to it.

However, a limited number of patterns can be stored and recalled in a HNN; this has an impact on its applicability for many pattern recognition tasks (Beale and Jackson, 1990). Also as mentioned previously, the HNN has no learning algorithm as such. As the current experiment required an ANN to possess the ability to learn patterns in data, the HNN was not chosen.

The next section (2.4.2.2) discusses the architecture and properties of the Self-Organised Map.

#### Self-Organised Map (SOM)

The type of learning utilised in a multi-layer perceptron requires the correct response (target) to be provided during training (Lippmann, 1987; Beale and Jackson, 1990; Haykin, 1999). This approach is known as supervised learning. Though biological

systems display this type of learning, they are also capable of learning by themselves. Learning without the assistance of targets is known as unsupervised learning.

A system with the capability to learn unsupervised, requires self-organisation. During training, such a system learns appropriate associations without any targets (or prior knowledge) being provided.

An ANN model of this type is the Self-Organising Map (SOM), also known as the Kohonen Network (after it founder Dr. Teuvo Kohonen). The SOM is a competitive neural network; such networks represent a type of ANN model where nodes in the output layer compete with each other to determine a 'winner'. The winner indicates which prototype pattern is most representative of the input pattern.

As illustrated in Figure 2.15, the SOM has only one layer of nodes (the competitive layer or sometimes referred to as the Kohonen feature map). This layer is two dimensional, with lateral interconnections forming a grid like topology. Note that the architecture (with only one layer of nodes) is different to the hierarchical structure of layers in perceptrons.

All inputs are connected to every node in the competitive layer. There is no designated output layer; each node in the competitive layer is an output node. As there is only the one layer of nodes, error cannot be fed backwards through the network. Instead, feedback is facilitated via the lateral interconnections of neighbouring nodes in the competitive layer.

When presented with the training input data, the learning algorithm organises the competitive layer nodes into local neighbourhoods that act as feature classifiers. The topology of nodes is configured by the cyclic process of comparing input patterns to vectors stored at each node.

Where inputs match the node vectors, that area of the feature map is optimised to be representative of that class of training data. From its initial state, the network 'settles' into a feature map that has local representation and is self-organised. The following formulae define the operations of the SOM.

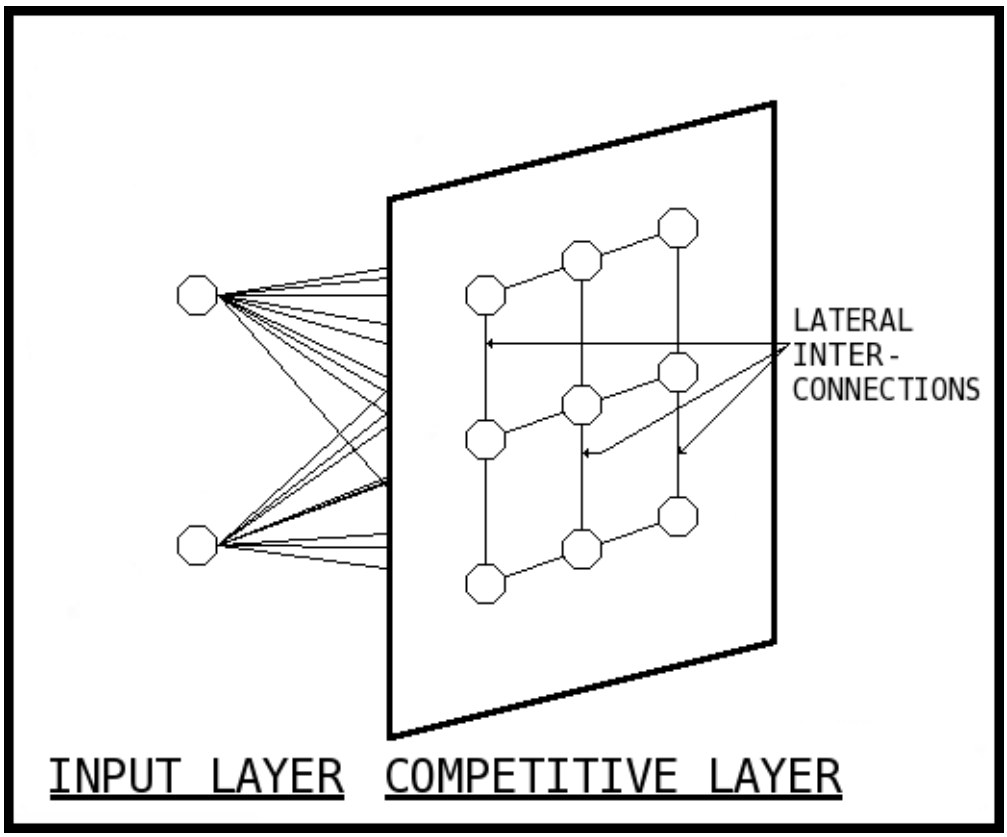


Figure 2.15: The Self-Organising Map (SOM)

Let $w_{ij}(t)$ be the weights from input $i$ to node $j$ at time $t$. Initialise (with small random values) the weights from $n$ inputs to the all nodes. Set to large, the initial neighbourhood around node $j$, $N_j(0)$. Present inputs $x_0(t), x_1(t), x_2(t) \ldots x_{n-1}(t)$, where $x_i(t)$ is the input to node $j$ at time $t$.

For each node $j$, calculate distance $d_j$ from input $x_i$ to node $j$, according to Equation 2.22:

$$d_j = \sum_{i=0}^{n-1} (x_i(t) - w_{ij}(t))^2 \tag{2.22}$$

Determine the node with the minimum distance and designate that node as $j*$.

With a neighbourhood size set to $N_{j*}(t)$ around node $j*$, update the weights between node $j*$ and its neighbours according to Equation 2.23:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\,(x_i(t) - w_{ij}(t)) \tag{2.23}$$

where $j$ is in the neighbourhood $N_{j*}(t)$, $(0 \leq i \leq n-1)$, $\eta(t)$ is the gain term $(0 < \eta(t) < 1)$ that decreases in time to slow weight adaption.

Note, all nodes in the neighbourhood of $j$ have their weights updated, by applying a proximity factor $\sigma(t)$ to the latter expression of Equation 2.23[30]. Importantly, the updating of weights does include any adjustment to the lateral interconnections of the competitive layer. It was previously stated that feedback is restricted to neighbours through the lateral interconnections. These interconnections do not have weight values themselves. They provide excitatory and inhibitory signals based on their proximity to their neighbours; that is, a node has excitatory connections to its immediate neighbours, and inhibitory connections to more distant nodes. All nodes in the competitive layer receive a mixture of excitatory and inhibitory signals from other competitive layer nodes (and of course from input layer nodes).

The weights that are updated (according to Equation 2.23) are those from the input layer nodes to the competitive layer nodes. The adjustment takes the form of a fractional factor applied to these weights. The proximity to the winning node determines the factor applied to the adjustment of weights.

As an input pattern is presented, some of the nodes are sufficiently activated to produce outputs which are fed back to other nodes in their neighbourhoods (via the weight adjustment process). The node with the weight vector closest to the input pattern vector (the winning node) produces the largest output.

During training, input weights of the winning node and its neighbours are adjusted to make them resemble the input pattern even more closely. At the completion of training, the winning node ends up with its weight vector aligned (identified) with the input pattern and produces the strongest output whenever that particular pattern is presented. Nodes in the winning node's neighbourhood also have their weights modified to settle to an average representation of that pattern class. As a result, unseen patterns belonging to that class are also classified correctly.

The reason for not utilising the SOM for the current experiment is because of its primary purpose. The SOM has the basic property of clustering similar objects close to each other on the feature map. Conceptually, a visual inspection of the feature map would indicate the clusters and thus the similarity relationships within the data set.

[30] i.e. $\eta(t)\sigma(t)\left(x_i(t) - w_{ij}(t)\right)$.

However, the current experiment required exact classification of data. Classification is an ordering of objects into predefined classes. By ordering of objects into n-classes the n-dimensional output layer must be defined. The SOM has no output layer as such, and therefore could not meet the requirement of the experiment.

The next section (2.4.2.2) discusses the architecture and properties of another ANN that utilises unsupervised learning: the Adaptive Resonance Theory model.

#### Adaptive Resonance Theory (ART)

The Adaptive Resonance Theory (ART) was developed to model a massively parallel architecture for self-organising neural pattern recognition networks, based on biological and behavioural data (Beale and Jackson, 1990).

The ANNs discussed in previous sections are implemented in two separate operational stages. Whilst during the training stage, these ANNs allow parameters and weights to be modified (until minimum error is achieved), the parameters and weights cannot be changed once training has ceased and testing begun (if the network is to be maintained in a stable state or condition). It means that the networks are not able to learn new information once training has ceased. This limitation is referred to as the stability-plasticity problem.

The ART has the ability to switch between a learning mode and a classification mode, without adversely affecting previous learning. As illustrated in Figure 2.16, the ART has a two layered architecture. As well as presenting input to the network, the input layer also provides a comparison functionality; the output layer produces network output and also provides a recognition functionality.

The nodes of each layer are connected to each other via weights, with the output layer nodes providing feedback to the input layer[31]. The ART incorporates feedforward weight vectors from the input layer to the output layer, and feedback weight vectors from the output layer to the input layer. The nodes of the output layer also provide feedback to each other; this feedback takes the form of lateral inhibition.

[31] For the sake of image clarity, weights in Figure 2.16 are shown between only four input layer nodes and two output layer nodes. Obviously in a real-world system, all input layer nodes would be connected to all output layer nodes.

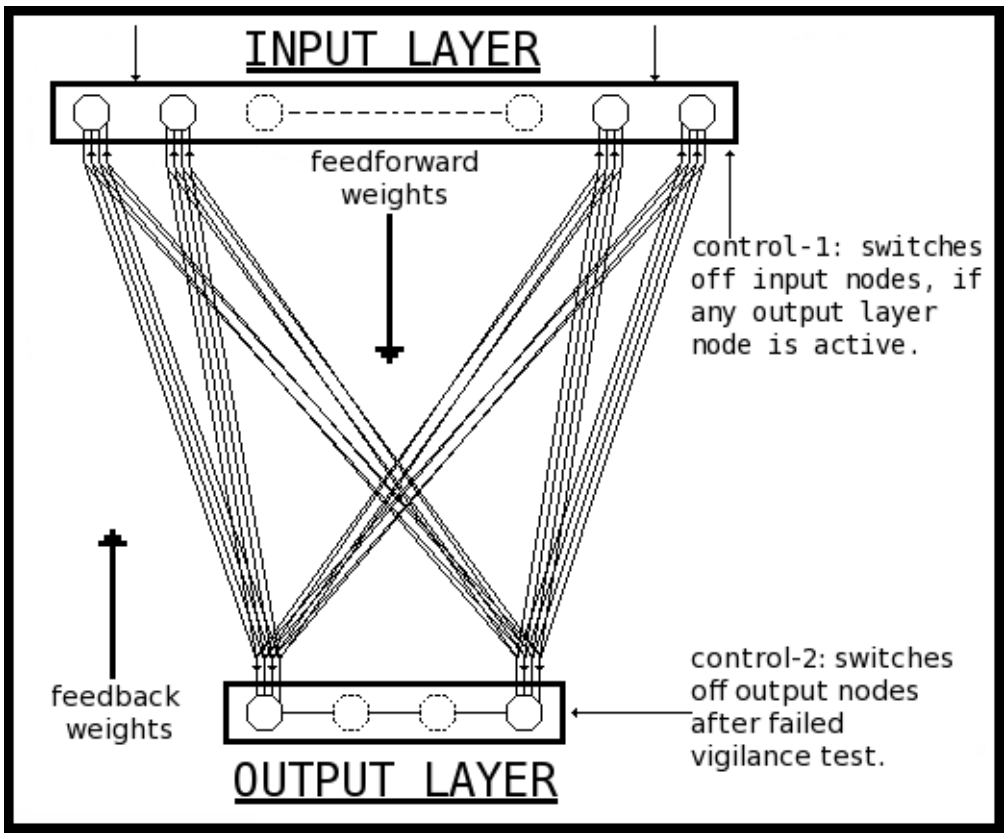


Figure 2.16: Adaptive Resonance Theory (ART)

The ART model also includes a control unit for each layer (control-1 for the input layer and control-2 for the output layer), which are responsible for data flow between layers during the operational cycle.

A reset unit provides functionality to reset output layer nodes (i.e. shut off all currently active nodes for a period of time, without affecting the ability of any inactive nodes to become active). The reset unit also performs a 'vigilance' test to determine whether to create new nodes (at the output layer) to accommodate a new class pattern; this would occur in the event of a new input pattern being presented to the network (Lippmann, 1987; Beale and Jackson, 1990).

Input layer nodes have three components; the input signal $x_i$, the feedback signal from output layer nodes, and the control-1 sentinel value. In the initialisation phase, control-1 determines which operational task the input layer should perform; accept input or perform comparison. If valid input is presented to the network, control-1 sets the sentinel value of each input node to 1. If any node at the output layer becomes active, control-1 sets the sentinel value of each input node to 0 (thus switching to comparison mode).

The output layer control unit (control-2) sets the sentinel value of each output node to 1 when valid input is presented to the network; these sentinel values get set to 0 if and when the vigilance test fails. The feedforward weight vectors are initialised to $w_i = \frac{1}{(1+n)}$, where $n$ is the number of input layer nodes; the feedback weight vectors are initialised to 1.

In the recognition phase, input presented to the network is matched against the classification represented at each output layer node. If any two components of an input layer node are active, the node output has the value 1 (otherwise it is 0).

Each weight vector at each output layer node can be thought of as a stored template or exemplar weight vector for the current input pattern. Based on the comparison between the dot product of the input vector and the exemplar weight vector at each output layer node, the 'winning node' is determined (i.e. the node with the largest dot product). Note that the lateral inhibitions between nodes in the output layer also influence the determination of the winning node.

The winning node then passes its stored template back to the input layer (via its exemplar feedback weight vector). Because the output layer now has an active node (i.e. the winning node), control-1 sets the sentinel value of each input node to 0. The two vectors (i.e. the input vector and the exemplar feedback weight vector) have an AND operation performed on them, which produces a new vector called the 'comparison vector'. This is passed to the reset circuit along with the input vector.

The reset circuit tests the similarity of the two vectors against the vigilance threshold according to Equation 2.24:

$$S = \frac{\sum t_{ij} x_i}{\sum x_i} \tag{2.24}$$

where $t_{ij}$ are the feedback weight vectors from node $j$ to nodes $i$.

If $S > \rho$ (where $\rho$ is the vigilance threshold), class membership (for the current input vector) is indicated for the winning node; otherwise the correct exemplar pattern has not been determined, and a search is made for another matching vector. The current winning node is disabled and prevented from further participation, and the procedure repeated to find another winning node.

The following equations provide the mathematical operations of the ART. Firstly, weights are initialised: $t_{ij}(0) = 1$ and $w_{ij}(0) = \frac{1}{(1+N)}$, where $t_{ij}(t)$ is the feedback weight vector between nodes $i$ and $j$ at time $t$, and $w_{ij}(t)$ is the feedforward weight vector between nodes $i$ and $j$ at time $t$. $M$ is the number of output nodes ($0 \leq j \leq M-1$) and $N$ the number of input nodes ($0 \leq i \leq N-1$).

After presenting new input, compute the matching score (to determine the winning node) according to Equation 2.25:

$$\mu_j = \sum_{i=0}^{N-1} w_{ij}(t) x_i \tag{2.25}$$

where $\mu_j$ is the output of node $j$ (for $0 \leq j \leq M-1$), $x_i$ is the $i^{th}$ element of the input vector (with possible values 0 or 1 only), and $w_{ij}(t)$ are the feedforward weight vectors from node $j$ to nodes $i$ at time $t$.

The recognition phase begins by determining the exemplar node $\mu_{j*}$ according to $\mu_{j*} = max_j(\mu_j)$. Preliminary calculations for the vigilance test are accomplished according to Equations 2.26 and 2.27:

$$||X|| = \sum_{i=0}^{N-1} x_i \tag{2.26}$$

$$||T \cdot X|| = \sum_{i=0}^{N-1} t_{ij*}(t) x_i \tag{2.27}$$

where $x_i$ is the $i^{th}$ element of the input vector, and $t_{ij*}(t)$ are the feedback weight vectors from exemplar node $j*$ to nodes $i$, at time $t$.

Next perform the following inequality test according to Equation 2.28:

$$\frac{||T \cdot X||}{||X||} > \rho \tag{2.28}$$

where $\rho$ (for $0 \leq \rho \leq 1$) is the vigilance threshold.

If the result of Equation 2.28 was FALSE, disable the current exemplar node $\mu_{j*}$, set the output of node $j*$ to 0, and search for a new $\mu_{j*}$.

If the result of Equation 2.28 was TRUE, adjust the exemplar weight vectors (both feedback and feedforward) according to Equations 2.29 and 2.30 respectively:

$$t_{ij*}(t+1) = t_{ij*}(t)x_i \tag{2.29}$$

where $t_{ij*}(t)$ are the feedback weight vectors from exemplar node $j*$ to nodes $i$, at time $t$.

$$w_{ij*}(t+1) = \frac{t_{ij*}(t)x_i}{0.5 + \sum_{i=0}^{N-1} t_{ij*}(t)x_i} \tag{2.30}$$

where $w_{ij*}(t)$ is the feedforward weight vector to exemplar node $j*$ from nodes $i$, at time $t$.

Finally after updating weights, enable any disabled nodes and repeat the process by presenting a new input pattern.

The above basic operations of the ART demonstrate the ability to deal with binary input only; this imposing limitations on its use for a wide variety of applications. Subsequent innovations to the original ART allow for continuous data input. The original ART is thus referred to as ART-1, and the subsequent versions as ART-2 and ART-3. These versions have not been discussed here, as this section was intended to provide background of the basic ART-1 operations.

The ART has the advantage of continuing to learn new information and continually refine existing knowledge (Beale and Jackson, 1990). This allows for the addition of new patterns without affecting already learned information or the speed of the classification process.

However, the ART was not used in the current experiment because of the following disadvantages (Beale and Jackson, 1990):

- It has demonstrated an inability to handle noisy data.
- Implementation and tuning of a network can be complex.
- The network is highly sensitive to parameter changes.

- Reliance on a single node (in the storage layer) that represents a particular pattern reduces reliability.

The next section highlights the properties of the Multi-Layer Perceptron which make this architecture ideally suited for the task of pattern recognition.

### 2.4.3 The Multi-Layer Perceptron As A Pattern Classifier

This section explains why the Multi-Layer Perceptron is suited to pattern recognition problems, and thus was chosen as the architecture utilised for the experiment in this study. That is, what properties the MLP exhibits which make it suitable for such tasks. Also presented are some measures that can be undertaken to assist MLP training.

As discussed in section 2.4.2.2, the MLP first presents an input pattern to the first layer of hidden nodes. This layer of nodes produces output which is then passed to the next layer (either a second hidden layer or the output layer). Because each node performs the functionality of a perceptron unit, each first layer node output defines a single plane of separation between two classes (Lippmann, 1989; Beale and Jackson, 1990).

A node in the second layer (hidden or output), receiving input from two nodes in the first hidden layer, performs a logical AND operation if its threshold is set to 'on' (with the prerequisite that both previous layer outputs are 'on'). Since each of the previous layer nodes define a linear classification in the pattern space, the receiving node produces classification based on a combination of these lines.

There may be many nodes in the first hidden layer; thus at the second layer (hidden or output) the combination of all linear classifications (corresponding to each node output from the first hidden layer) partitions the pattern space to form a convex region (called a convex hull). A convex hull is defined as a region where any point can be connected to any other point by a straight line that does not cross the boundary of the region. Thus the convex hull comprises a region where the intersection of all linear classifications occur.

If a third hidden layer were to be added, the nodes of this layer would receive convex hulls as input (from the second hidden layer nodes). Thus a third layer would provide the facility to define arbitrary shapes. Whether a third hidden layer is added or not, it is clear that the MLP is very well suited to classifying data of a complex nature (and determining patterns within that data).

In general, MLPs exhibit the following properties or characteristics which make them ideally suited for application to the task of pattern recognition (Lippmann, 1989):

- They can solve problems that are too complex for conventional technologies; that is, problems that do not have an algorithmic solution or where an algorithmic solution is too complex to be computationally viable.

- They can be effectively applied to intelligence applications where a real-time response, using complex real-world data, is necessary.

- They can accurately associate input patterns to their correlated output patterns. The storage and summing functionality of MLP nodes facilitates their 'learning' capabilities (thus simulating the biological neuron). When tested with unknown instances, they exhibit inference capabilities.

- They have the capability to generalise. That is, if provided with a subset of available samples, the MLP can be guided—during the training phase—to accurately detect patterns, and then to correctly classify them in unseen samples (of the same class pattern) during the testing phase.

- They are robust and fault tolerant systems, and frequently produce reduced error rates when compared to conventional approaches. This property allows them to recall full patterns when presented with incomplete or noisy patterns.

The following steps can be undertaken to assist MLP training, and thus improve the MLP capacity for discerning patterns in data (Wong et al., 2005):

1. The use of cross validation during the training process. This has the effect of 'smoothing' data to reduce the effect of noise and variability, and prevent

over fitting. Cross validation involves selecting a small subset of the available training samples for validation during the training process. That is, as the MLP is training or learning, additional samples are used by the algorithm to reinforce correct learning.

2. Prior statistical analysis of the data. This further enhances the learning capability of the MLP during the training process. Analysis involves the identification of features responsible for data noise. For instance, higher frequencies of values at the extremities of a normal distribution indicate noisy data. Thus, by selecting features less affected by noise (thereby removing those responsible for noise), the MLPs ability to discern patterns in the data is improved.

3. The use of a large number of hidden layer neurons to reduce the effect of bias, and to prevent under fitting. Selection of the number of hidden layer neurons is typically a trial and error process, because there is no definite method or rule for this determination. Kasabov (1996) suggested starting with half the number of input layer neurons, and adjusting that quantity (up and/or down) until a satisfactory result is achieved.

As indicated by point 2 above, pre-processing data (to remove features responsible for noise) may greatly improve the performance of the MLP. The following attributes further highlight the benefit of pre-processing data for a pattern recognition system (Bishop, 1995):

- The pre-processing tasks, of identifying appropriate features to represent patterns and the subsequent extraction of those features, result in prior knowledge being obtained from the raw data. Obtaining prior knowledge for any task involving supervised learning could be seen as beneficial.

- Reducing the number of inputs (by such pre-processing), often leads to improved performance by mitigating (at least to some degree) the curse of dimensionality.

So as well as being beneficial to MLP training by providing prior knowledge, pre-processing also improves efficiency in the training process by reducing the input pattern space.

For all of the above reasons, the MLP with back-propagation was selected for use in the current experiment (refer Chapter 5). In the experiment, one MLP was trained to recognise the input pattern of each training group member. As there were three phases of the experiment, there were three types of input patterns (one type for each phase) presented to the MLP for each participant. The three types of patterns corresponded to keystroke dynamics data (refer Chapter 5 section 5.4), fingerprint feature data (refer Chapter 5 section 5.5), and data resulting from the fusion of the other two types (refer Chapter 5 section 5.6).

The next section concludes this chapter by summarising the relevance of this chapter to the overall discussion.

## 2.5 Conclusion

This chapter provided a discussion on three areas of study associated with the experiment conducted for this dissertation. The material presented provided background to help understand these associated areas, and why certain choices may have been made during the experimental stage of the study.

The first area of study associated with this dissertation, was that of biometrics (section 2.2). An overview of biometrics was presented in section 2.2.1, which included its definition as the personal characteristics that make individuals unique. Also discussed were reasons why biometrics provide an alternative to traditional authentication procedures, the difference between physical and behavioural biometric characteristics (their advantages and disadvantages), and the biometric technologies that have evolved in this area.

The components of a biometric authentication system were discussed in section 2.2.2. These include a capture module, a feature extraction module, a matching module, and a decision module. The requirements of a biometric characteristic, for use in a biometric authentication system, were presented as: universality, unique-

ness, permanence, and collectability. The biometric authentication system also needs to consider performance, acceptability, and circumvention. Also, the two phases of biometric authentication system (enrolment and validation) were described.

As with any system, possible errors require identification and performance requires measurement. Two system error rates (the failure to capture rate and the failure to enroll rate) were described in section 2.2.3; this section also discussed the performance variables used to present experimental results (the false acceptance rate, the false rejection rate, and the equal error rate).

A description of a number of well known biometric characteristics—for possible use in a biometric authentication system—was provided in section 2.2.4. These included facial recognition, iris and retinal pattern recognition, speaker recognition, fingerprint and palmprint recognition, hand geometry, keystroke dynamics, signature recognition, gait recognition, and body odor recognition.

The material presented in section 2.2 was researched in an effort to understand the requirements of a biometric authentication system, and to identify two biometric characteristics that could be utilised for the current investigation. The choice of which two biometric characteristics to utilise was based on the ability to achieve accurate verification, and operational considerations in terms of cost effectiveness and ease of use in a biometric authentication system. A detailed discussion of the two biometric characteristics chosen for the current experiment, and reasons for their selection, is provided in Chapters 3 and 4.

Section 2.3 discussed the areas of data fusion and multi-modal biometrics. An overview of data fusion was presented in section 2.3.1, where the data fusion paradigms (refer section 2.3.1.1), the data fusion levels (refer section 2.3.1.2), and data alignment (refer section 2.3.1.3) issues were described. Section 2.3 also provided an overview of multi-modal biometrics in section 2.3.2. Data fusion levels (as applied to this area of research) were discussed in section 2.3.2.1, and a review of related literature was presented in section 2.3.2.2.

An overview of pattern recognition and Artificial Neural Networks was presented in section 2.4. Section 2.4.1 provided an overview of pattern recognition, and some

classification schemes were described in section 2.4.1.1. Artificial Neural Networks were discussed in section 2.4.2. Section 2.4.2.1 explained how ANNs attempt to model the functionality of the biological neurons in the human brain, and section 2.4.2.2 described a number of different ANNs architectures and their operations. Section 2.4.3 explained why the Multi-Layer Perceptron is particularly suited to pattern recognition problems, and the reason why this ANN architecture (in preference to other ANN architectures) was used for the experiment in this study.

In the next chapter an in-depth discussion of the biometric characteristics keystroke dynamics is presented.

# Chapter 3

# Keystroke Dynamics

## 3.1 Introduction

This chapter provides a discussion of the biometric characteristic known as keystroke dynamics. The overview of the subject area provides the conceptual basis of keystroke dynamics (refer section 3.2).

Section 3.3 describes the possible metrics that may be used for experimentation in this area of research, including those used in the current study. The calculation of metrics for the current study is described in Chapter 5 section 5.4.3.

Keystroke dynamics metrics exhibit higher degrees of variability than some other biometric characteristics; thus it is not recognised to be as accurate as some of the other characteristics. Section 3.4 provides a review of research efforts in this area to demonstrate that keystroke dynamics can confidently be used, provided the appropriate issues are carefully considered.

Finally, section 3.5 summarises a number of the issues associated with the appropriate use of keystroke dynamics, and section 3.6 concludes the chapter.

## 3.2 Overview of Keystroke Dynamics

Keystroke dynamics is a behavioural biometric characteristic that involves analysing a computer user's habitual typing pattern when interacting with a computer keyboard (Monrose and Rubin, 2000).

Typing involves predominantly subconscious control of finger movement (at least for those who type regularly); it incorporates movement characteristics that are different between individuals and consistent over time (Gaines et al., 1980). The habitual nature of typing facilitates the development of a 'typing signature' that is distinguishable enough between people to be used for authentication purposes (Gaines et al., 1980).

When analysing typed samples from different people, examination of the times between keystroke events reveals a definite pattern for each person. Digraph is a term used to describe a pair of typed characters. Digraph time is the time taken from the depression of the first key to the depression of the following key. It is also termed keystroke time or keystroke latency[1]. For example, when typing the digraph "th", different people will achieve a different keystroke latency for this character pair. By combining the keystroke latencies for all pairs of characters in a typed text, a definite pattern for each person becomes evident.

Digraphs are analysed because they are the most elemental typing unit (as opposed to analysing the time taken to type a complete word or a sentence or a paragraph) (Gaines et al., 1980). Keystroke latency—based on digraphs—allows for a fine granularity in the calculation of metrics that may be used to discern a pattern; it is a basic metric, which can be calculated according to time differences between successive pairs of digraphs or keystroke combinations. As such, a vector of metrics is determined for each person, which describes a unique pattern for that person. There are other possible metrics which allow for an even finer granularity than keystroke latency. A full discussion of the possible metrics and how they are calculated is given in section 3.3.

When analysing a pattern for the purpose of initial authentication (i.e. logging on), verification is performed at a discrete time (the time of attempted authentication). This is termed 'static verification' (Maher et al., 1995). Once a claimant has been authenticated, no further attempt is made to verify the claimant's identity. That is, the authentication procedure has accepted that the claimant is who they profess to be.

[1]Keystroke latency will be used to describe this metric for the remainder of this document.

However, if further verification is deemed necessary, analysis of a claimant's typing pattern can be performed on a periodic basis, during the period they are logged on. That is, after initial authentication, the claimant's ongoing interaction with the computer can be periodically monitored. This is termed 'dynamic verification' (Maher et al., 1995), and can be useful as a subsequent check of identity. For instance, it can be used to determine if another user has gained unauthorised access to a terminal, in the temporary absence of the legitimate user.

Important considerations when attempting authentication based on keystroke dynamics are:

- Is the objective to determine the identity of a user from among a population (identification) or to verify a claimed identity (verification)?
- Is the intent to perform static or dynamic verification?
- What text length should be used for analysis of the typing pattern (i.e. no specific length or a pre-determined length)?
- Should standard text or a prefabricated character string be used?
- What metrics should be used in the analysis of keystroke characteristics? A full discussion of keystroke metrics is given below in section 3.3.
- What method of analysis should be used when performing pattern recognition? A number of early studies used deterministic statistical methods. However, in the last two decades, a number of studies have used Artificial Neural Networks (ANNs) and other machine learning techniques.

These issues and experimental validity will be considered in the review of relevant literature (section 3.4). The next section however, describes in detail the possible metrics that can be used in the analysis of keystroke characteristics.

## 3.3 Metrics

Keystroke latency was the first metric identified in the study of keystroke analysis. It was the initial basic metric, calculated according to the time difference between

the key press events of any digraph (that is, two character combination). Some research efforts have introduced the concept of a 'trigraph', which is a term used to describe a triplet of typed characters (Bergadano et al., 2002; Hu et al., 2008).

The general term for $n$ number of successively typed characters is an $n$-graph. Whatever the size of $n$, the keystroke latency still remains a single metric which measures the time difference between the depression of the first character in the sequence and the depression of the $n^{th}$ character in the sequence.

Since the work of Brown and Rogers (1993), the most common metrics used in keystroke dynamics research are the 'keystroke duration' and 'digraph latency'; these metrics are calculable from any digraph. Although other metrics are possible (as illustrated by the blue lines in Figure 3.1), keystroke duration and digraph latency are the most widely used and accepted (refer section 3.4), and thus were used in the current experiment.

With any digraph, there are four events corresponding to the depression and release of each character key in the digraph. The four events denote six possible individual measurements (Peacock, 2000). As illustrated in Figure 3.1:

1. T1 represents the time the 1st key is depressed.

2. T2 represents the time the 1st key is released.

3. T3 represents the time the 2nd key is depressed.

4. T4 represents the time the 2nd key is released.

The red lines in Figure 3.1 highlight the keystroke duration and digraph latency. Thus, keystroke duration is the total time a key is depressed. Digraph latency is the time distance between two successive keystrokes (i.e. the time between the release of a key and the depression of the next key).

By definition then (as demonstrated by the red lines in Figure 3.1), the two metrics are the time differences between states. That is, T1 $\Longrightarrow$ T2 represents the keystroke duration and is calculated by subtracting T1 from T2. Similarly, T2 $\Longrightarrow$ T3 represents the digraph latency, and is calculated by subtracting T2 from T3.

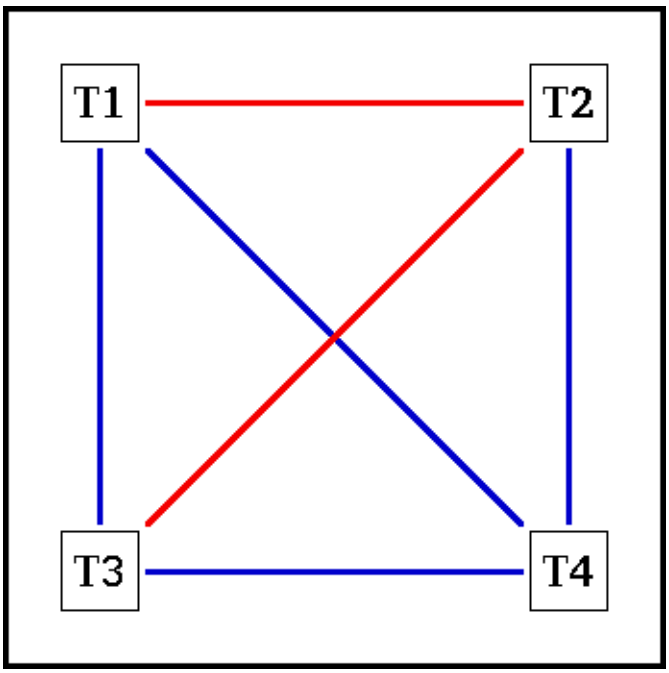


Figure 3.1: States of a Two-Key Combination

As an example, Figure 3.2 illustrates the keystroke durations and digraph latencies for the two character sequence "th", followed by the depression of the Enter key. The depression of the Enter key is necessary for the calculation of the last metric of the last character (in this case, the digraph latency for the letter 'h').

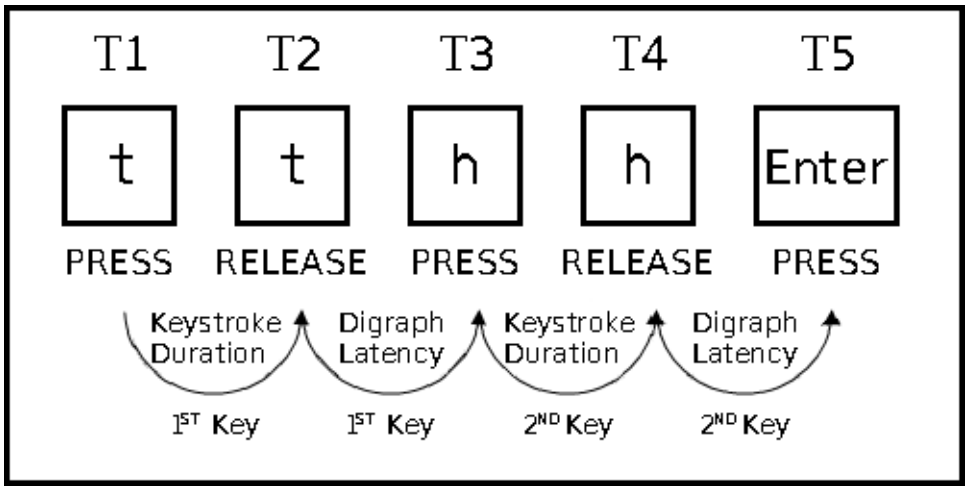


Figure 3.2: Keystroke Durations and Digraph Latencies for the digraph "th"

Table 3.1 demonstrates the calculation of metrics for the illustration in Figure 3.2. Note that it is possible for the digraph latency metric to be of negative magnitude. This may occur if the second key of the digraph is depressed before the first key is released. In the above example, T3 (the depression of the second key) could occur

before T2 (the release of the first key) and thus the calculation of the metric could result in a negative magnitude. Similarly, T5 could occur before T4.

| METRIC | $1^{ST}$ KEY | $2^{ND}$ KEY |
|---|---|---|
| Keystroke Duration | T2 - T1 | T4 - T3 |
| Digraph Latency | T3 - T2 | T5 - T4 |

Table 3.1: Metric Calculation for a Two-Key Combination

However for the keystroke duration, a negative magnitude is not possible because the measurement of the metric is not initiated until a depressed key is released. So in the above example, T2 must occur after T1 and thus the calculation of the metric can only result in a positive magnitude. Similarly, T4 must occur after T3.

The next section presents a review of literature relating to keystroke dynamics research.

## 3.4 Keystroke Dynamics Related Research

Card et al. (1980) proposed the keystroke-level model to evaluate the cognitive processes in the completion of a typing task. The model was originally intended as a design evaluation method for highly interactive programs (Umphress and Williams, 1985). Inadvertently, it formed the basis for research into the keystroke characteristics that determine an individual's typing pattern.

The model for measuring a typing task is represented as the sum of an acquisition time and the execution time. The acquisition time is the time required to assess the overall task; this includes building a mental representation of the functions to be performed, and choosing appropriate methods to perform them (Card et al., 1980). The acquisition time involves strategic planning, and varies depending on the extent of the task. It is not easily quantifiable and cannot reliably characterise an individual (Umphress and Williams, 1985).

The execution time is the time required to call on the system resources to accomplish the tasks (Card et al., 1980). Execution time describes physiological actions that are quantifiable. It is defined as the sum of the time required for mental prepa-

ration (i.e. tactical planning, as opposed to strategic planning in the acquisition time) and the time to key in information (keystroke time) (Umphress and Williams, 1985). Here, the sum of the keystroke time and mental time is not considered a macroscopic operation, but the sum of sub-tasks of keystroke and mental times. Each sub-task is referred to as a cognitive unit, and the keystroke times within each cognitive unit characterise an individual's typing pattern.

A number of research efforts further investigated the concepts proposed by Card et al., (1980), specifically concentrating on analysis of keystroke times. These studies have demonstrated the effectiveness of keystroke dynamics as a method of identity verification. However, though the first papers were published as long ago as 1980, by Gaines et al., it has not been a widely researched field.

With the emphasis of improving the initial authentication procedure (logging on), the majority of the research efforts have concentrated on static verification (refer Table 3.2 in section 3.4.1), with only a small number investigating dynamic verification (refer Table 3.3 in section 3.4.2). Thus this review first considers experiments where static verification has been investigated, including recent research utilising ANNs. For completeness, a very brief review of some experimental work where dynamic verification has been investigated, will follow[2].

### 3.4.1 Static Verification

The research findings and methodological issues relating to keystroke dynamics experiments involving static verification—as conducted by the authors of the papers reviewed—are summarised in Table 3.2[3].

As well a being useful as a quick reference for the following discussion, the information in Table 3.2 will be used in Chapter 7 to compare results achieved in the current study with those of previous research.

[2]It should be noted that the review is by no means a comprehensive coverage of all work done in this field. Rather the research efforts reviewed here were chosen to provide an overview of the current status in this field, and to provide figures with which to compare the results from the current experiment.

[3]Although some authors expressed the performance variables (FAR and FRR) as a percentage, columns 9 and 10 denote the actual rates (i.e. the percentage divided by 100). During the discussion, the performance variables are presented as both the actual rates and their corresponding percentages (in parenthesis).

| Reviewed Paper | Number of Participants | Samples Per Participant | Collection Sessions | Text Length | Resolution in Milliseconds | Metrics | Analysis | FAR | FRR |
|---|---|---|---|---|---|---|---|---|---|
| Gaines et al., 1980 | 6 | 3 | 2 | 818 words | 1 | 1 | DSC | 0.0 | 0.04 |
| Umphress and Williams, 1985 | 17 | 2 | 2 | 1700 chars | 10 | 1 | DSC | 0.0588 | 0.1176 |
| Leggett and Williams, 1988 | 36 | 2 | 2 | 537 chars | 10 | 1 | DSC | 0.05 | 0.055 |
| Joyce and Gupta, 1990 | 6+27 | 13+30 | 2 | 32 chars | na | 1 | DSC | 0.0025 | 0.1636 |
| Brown and Rodgers, 1993 | 46+15 | 41+30 | 2 | 8 chars | 1 | 2 | ANN | 0.0 | 0.115 |
| Obaidat and Sadoun, 1997 | 15 | 9210 | 41 | 7 chars | 1 | 2 | ANN+SPR | 0.0 | 0.0005 |
| Cho et al., 2000 | 21+15 | 275+75 | na | 8 chars | 1 | 2 | ANN | 0.0 | 0.01 |
| Peacock, 2000 | 11 | 20 | na | 24 chars | 1 | 1 | KNN | 0.042 | 0.08 |
| Bergadano et al., 2002 | 44+110 | 5+1 | na | 683 chars | 10 | 1 | DSC | 0.0001 | 0.04 |
| Yu and Cho, 2004 | 21+15 | 350+75 | na | 6-10 chars | 1 | 2 | GA-SVM | 0.0 | 0.0369 |
| Abernethy et al, 2004 | 50 | 50 | 5 | 32 chars | 1 | 2 | ANN | 0.0119 | 0.108 |
| Jiang et al., 2007 | 58+257 | 20+28 | na | 9 chars | 1 | 1 | HMM-GM | 0.0254 | 0.0254 |
| Revett et al., 2007 | 20+30 | 520+80 | 14x3 | 6-15 chars | 1 | 8 | PNN | 0.0195 | 0.0195 |
| Hu et al., 2008 | 19+17 | 5+27 | na | 100 chars | 1 | 1 | KNN | 0.00045 | 0.0 |

| LEGEND | ANALYSIS METHOD DESCRIPTION |
|---|---|
| na | Not Available |
| DSC | Deterministic Statistical Calculations |
| SPR | Statistical Pattern Recognition Techniques |
| ANN | Artificial Neural Networks |
| PNN | Probabilistic Neural Networks |
| KNN | K-Nearest Neighbour Algorithm |
| GA-SVM | Genetic Algorithm using Support Vector Machine |
| HMM-GM | Hidden Markov Models and Gaussian Modeling |

Table 3.2: Summary of Reviewed Literature Involving Static Verification

The first reviewed investigation into static verification was by Gaines et al. (1980). They had 6 participants enter three paragraphs of text in two collection sessions (consisting of 818 words of prose in total). The space character was included as a valid character in a digraph. Keystroke events were captured at a resolution of 1 millisecond, and keystroke latency was the metric used.

The aim was to determine if a 'typing signature'—that could be used for accurate verification—was apparent; and if so, which digraphs (character combinations) indicated more significance than others.

Statistical analysis was performed using T-tests on 87 out of a possible 729 digraphs. Only those digraphs that occurred 10 times or more for all participants were used. Also, digraphs that included capital letters, backspace, and a keystroke latency greater than 500 milliseconds were removed from the analysis[4].

The results were presented as a false acceptance rate (FAR) of 0.0 and a false rejection rate (FRR) of 0.004 (4.0%). The conclusion was that a 'typing signature' was apparent for each participant. Furthermore, the digraphs 'in', 'io', 'on', 'no', 'ul', 'il', 'ly' were more distinguishable than other digraphs. Note that these digraphs are all typed by the right hand (in the standard typing method). The researchers believed that the corresponding combinations on the left hand would also provide significant distinctiveness, though there was not enough data to demonstrate this.

Though the results were impressive, it was a preliminary study with only 6 participants. Also, each participant contributed only 3 samples, and only 87 digraphs (from the typed text) were tested. Therefore, the researchers were cautious about the same degree of accuracy being achievable in a larger population. They also believed that further research into the significant digraphs was warranted. Even with these validity concerns, the investigation provided a valuable framework for successive studies.

Umphress and Williams (1985), recruited 17 participants who provided one sample each over 2 collection sessions. The first session required participants to enter 1,400 characters of prose which was used to determine the reference template, while

[4]Keystroke latencies over 500 milliseconds in duration were discarded because it was considered an indication that the typist may have become distracted or lost concentration.

the second session required entry of 300 characters of prose to be used as test data. No reason was given for the chosen text length. Keystroke events were captured at a resolution of 10 milliseconds, and keystroke latency was the metric used.

Because data collection involved typing prose, the following filtering was applied to the collected data:

- Only digraphs within the first six characters of words were considered. This was based on work by Cooper (1983), who determined that typists typically pause between words as well as within words that are longer than six characters.
- Digraphs within and word that contained a typing error were eliminated.
- Keystroke latencies over 750 milliseconds in duration were discarded because they were not considered good candidates for inclusion in the typists reference template. Reasons for durations over this limit could be that the typist became distracted, lost concentration, or was unfamiliar with the keyboard.
- Digraphs which included capital letters had the latencies of the appropriate key and the 'Shift' key summed and halved.
- The space character was not included as a valid character in a digraph.

For each participant, two measures were generated from the filtered data for their reference template:

1. The mean keystroke latency time for each digraph.
2. The overall mean and standard deviation of the keystroke latency times.

A matrix of 26 rows and 26 columns was used to store the metrics for all digraphs. The rows represented the first character of a digraph and the columns represented the second character. The mean keystroke latency for each digraph was calculated and recorded at the corresponding intersection of row and column.

The analysis involved comparison of the mean keystroke latencies from the reference template matrix with the corresponding test data matrix. The overall mean

and variance were used as a measure of tolerance between two corresponding values. If a test keystroke latency was within 0.5 standard deviations of the corresponding reference template mean value, the test keystroke latency was considered valid. The experiment achieved FAR and FRR results of 0.0588 (5.88%) and 0.1176 (11.76%) respectively.

The issues of concern with this research are: the use of a course metric (the keystroke latency; a timing resolution of 10 milliseconds; the small number of participants (17); the small number of samples (2 per participant); and the use of the mean and variance as a course classifier.

Leggett and Williams (1988) replicated the above experiment with some modifications. They recruited 36 participants, who provided one sample each over 2 collection sessions. In both sessions participants were required to enter 537 characters of prose; the first set was used in determining the reference template, while the second set was used as test data. Again, no reason was given for the chosen text length. Keystroke events were captured at a resolution of 10 milliseconds, and keystroke latency was the metric used.

The authors evaluated a number of different digraph latency combinations in an attempt to determine which would prove the most accurate, but found that the method previously used by Umphress and Williams (1985) returned the best results. They used the same analysis method as Umphress and Williams (1985), except that the overall mean latency time was discarded because the standard deviation alone was more determinant.

The results achieved were 0.05 (5.0%) for the FAR and 0.055 (5.5%) for the FRR. This shows a slight improvement in the FAR and a marked improvement in the FRR. The improved results could be attributed to having more participants and a larger test data set than the earlier experiment by Umphress and Williams (1985). However, the same concerns identified in the Umphress and Williams (1985) study, were also applicable to the Leggett and Williams (1988) study.

Joyce and Gupta (1990) conducted an experiment with 33 participants. Each provided 13 samples, in an initial enrollment session, from which 8 were used to

derive the reference template and 5 were used to test for false rejection. Six participants were then randomly chosen as targets for the remaining 27 participants to attempt to impersonate. Each of the 27 'impostors' provided 5 samples per target to test for false acceptance. Therefore, the number of samples obtained from each of the 27 non-target participants was 30 (6x5). Each sample consisted of four components (username, password, first name, last name). It was estimated that samples comprised an approximate average of 32 characters. The metric used to characterise the digraph times was the keystroke latency.

A reference template for each target participant was determined as follows: for each keystroke latency in the participant's 8 samples, the mean and standard deviation were calculated. Latencies outside 3 standard deviations of the mean were considered outliers and discarded. The means were re-calculated from the remaining latencies. A resultant vector of keystroke latency means was referred to as the mean reference template.

Testing against the reference templates was achieved as follows: the magnitude of difference between the mean reference template and a 'test signature' was computed. This involved comparing each mean latency in the reference template vector with the corresponding value in the 'test signature' vector, and returning a single difference value between the two. A threshold value was used to decide if the 'test signature' was valid. This threshold was determined as the mean plus 1.5 standard deviations (the standard deviation obtained when determining the reference template). If a 'test signature' difference value was within the threshold variability of the mean reference template, it was accepted as valid.

The results achieved were a FAR of 0.0025 (0.25%), and a FRR of 0.1636 (16.36%). The FAR seems quite acceptable, thought the FRR is very high. However, there are a few areas of concern. With only 8 samples used in determining the reference templates, the means and standard deviations may not contain enough meaningful information to be distinguishable in a larger population.

Also, using only 5 samples to test for false rejection means that the FRR has a very course granularity. The FAR was tested by comparing 810 samples against each

of the 6 target reference templates, but this is a very low number of participants to target. It is questionable that these results would be representative should a larger population be tested. Therefore, it is difficult to have confidence in the low FAR of 0.0025 (0.25%).

Brown and Rogers (1993) conducted the earliest known investigation—utilising ANNs for pattern recognition—for identity verification of keystroke dynamics data. They also introduced the concept of using two metrics to characterise the digraph times (rather than just the keystroke latency). The two metrics were the keystroke duration and digraph latency (refer section 3.3). Two metrics give a finer granularity and provide a more determinable pattern. Keystroke events were measured at a 1 millisecond resolution.

Forty six participants were recruited; each provided an average of 40 samples of their own username (approximately 8 characters) and one sample of each of the other usernames (45). For each participant, 30 samples were used for positive case training of the ANN; there were 45 negative case samples available per participant[5]. As well as the 30 samples per participant used for training the ANN, there were 10 samples available for testing for false rejection. Also, there were 30 samples collected from 15 different volunteers to test for false acceptance.

The experiment first used the Kohonen network (self-organizing map or SOM) to eliminate outliers from participants typing samples. A Kohonen network is a neural network that organses dimensionally complex data into clusters. This capability clusters a set of inputs according to intrinsic relationships. Any authentic input not mapping to the appropriate cluster was considered an outlier and discarded.

For testing verification, the single layer perceptron (SLP), the multi-layer perceptron (MLP), and a distance measure were used for classification. The experiment achieved a FAR of 0.0 (for all classifiers), with a FRR of 0.149 (14.9%) for the distance measure, 0.174 (17.4%) for the SLP, and 0.115 (11.5%) for the MLP.

The authors reported that the FAR (for all three classifiers) was forced to achieve a rate of 0.0 (by adjusting a decision threshold). The reason for forcing the FAR to

[5]Positive case samples are those that the ANN is being trained to recognise. Negative case samples are those that the ANN should not recognise, thus aiding the ANN in distinguishing between correct and incorrect patterns during training.

0.0 was because authentication in mission critical situations typically require a FAR of 0.0 (or at least very close to 0.0). In an experimental setting however, this could be seen as influencing the outcome, and thus was avoided in the current study.

The FRR results show that the MLP (with a FRR of 0.115 (11.5%)) performed much better than the SLP (with a FRR of 0.174 (17.4%)); even the distance measure (with a FRR of 0.149 (14.9%)) performed better than the SLP. The probable cause for the SLP having a higher FRR than the MLP was probably due to the fact that error back propagation is not employed in SLP neural networks. Whilst weights are adjusted when training a SLP, errors are not used to update weight values via back-propagation through the network as is done in the MLP.

The unimpressive FRR values could also have been impacted by course granularity, with only 10 samples per participant available for testing false rejection. The FAR of 0.0 (although 'forced') demonstrated that using ANNs for recognising typing patterns returned better results than the deterministic statistical methods used by previous researchers. It also demonstrated the better performance attainable by using two metrics instead of one.

Obaidat and Sadoun (1997) conducted an experiment that compared the performance of various statistical pattern recognition techniques with ANN architectures for pattern recognition. There were 15 participants who provided 225 samples per day for eight weeks. Assuming this means working days, this gives a total of 9,000 samples. These samples were partitioned into 4,500 training samples and 4,500 testing samples. In addition, to test for false acceptance, each participant typed 15 samples of the other participants' usernames (that is, 14x15 = 210).

Although a previous experiment determined that strings of less than 10 characters in length steeply reduced accuracy (Bleha and Obaidat, 1991), usernames had an average length of just 7 characters. The metrics used were the keystroke duration and digraph latency. Keystroke events were measured at a resolution of 1 millisecond.

The statistical pattern recognition techniques tested were the K-Means Algorithm (KMA), the Cosine Measure Algorithm (CMA), the Minimal Distance Algo-

rithm (MDA), Bayes' Decision Rule (BDR), and Potential Functions (PF). The ANN architectures tested were the Multi-Layer Perceptron with Back-propagation (MLP-BP), the Counter-propagation Neural Network (CPNN), the Fuzzy ARTMAP (FA), the Radial Based Function Network (RBFN), the Learning Vector Quantisation Network (LVQ), the Reinforcement Neural Network (RNN), the Sum-Of-Product Network (SOP), and the Hybrid-Sum-Of-Product Network (HSOP).

The best statistical pattern recognition technique was the PF with a FAR of 0.007 (0.7%) and a FRR of 0.019 (1.9%). The BDR also achieved results very close to these results. However, the study found that ANNs performed much better overall than the statistical pattern recognition techniques. Of these classifiers, the MLP-BP (using the sigmoid transfer function), the FA, the RBFN, and the LVQ performed best with FAR and FRR results of (0.0, 0.001), (0.0, 0.0), (0.0, 0.0), and (0.0, 0.0) respectively.

These results were excellent, although there were some concerns. Firstly, there were only 15 participants, and these may not be representative enough of a larger population. Secondly, by the authors own admission the text length should have been longer than 7 characters.

When training ANNs it is not typical to use as many training samples as the authors used (4,500 per participant). This tends to over train the networks, which can distort the true classification accuracy (particularly if other validity issues are present). A 'rule of thumb' for positive case training data sets is to use approximately 30% of the total number of samples acquired (Kasabov, 1996). However with 9,000 samples available, this would eventuate in 3,000 samples, which still seems an inordinately high number of samples.

This was an important study, as it was the first to compare statistical pattern recognition techniques with ANN architectures for pattern recognition. It demonstrated the effectiveness of ANNs for recognising typing patterns, as opposed to using deterministic statistical methods and other pattern recognition techniques. Also, the accuracy achieved lends credibility to the use of keystroke dynamics as an accurate person identifier. These results had a key influence on the design of the author's study.

Another study, by Cho et al. (2000), also used ANNs for static verification. The data collection was conducted, unsupervised, over the World Wide Web (WWW). They recruited 21 participants, each providing between 150 and 400 samples of their password (an average of 275 samples). Passwords were typically less than 8 characters in length. The last 75 samples were set aside for testing (for positive recognition), the remainder were used for training the ANN after outlier samples were discarded. A different group of 15 participants were recruited to provide 75 samples each of the 21 known passwords.

The author's approach was to use the Multi-Layer Perceptron as an auto-associator (MLP-AA) for identity verification. The MLP-AA is a neural network where the input vector is also used as the target. During the training process, the MLP learns by encoding properties from the data (i.e. typing pattern) in the input vector into the network (Cho et al., 2000). During the testing process, when presented with an unseen pattern—supplied by the same identity who provided the input vector for training the network—the network outputs a vector very close to the input vector. When presented with an unseen pattern—supplied by a different identity—the network outputs a vector very different to the input vector.

The experiment achieved FAR results of 0.0, with FRR results varying from 0.0 to 0.01 (1.0%). Two thirds of the participants tested had FAR results of 0.0 and FRR results of 0.0. Although this seemed impressive, the number of participants was small and the data collection was unsupervised. There was nothing to prevent participants using the copy and paste facility when entering their samples (thus potentially corrupting the experiment).

Also, there were only 1,125 (15x75) samples available for testing false acceptance, and 75 for testing false rejection (for each of the 21 participants). These concerns raise internal validity issues, and caste some doubt on the ability to replicate the results.

An experiment conducted by Peacock (2000) introduced the concept of using the arithmetic sum of all 6 possible measures between a two key combination (refer section 3.3), resulting in only one value to characterise each digraph. Keystroke events were measured at a resolution of 1 millisecond.

There were 11 participants who provided 20 samples each. A sample comprised a username, password, and a 9 character string. For each participant, 15 samples were used as training data for the learning algorithm, and 5 as validation data.

The analysis was performed by the k-Nearest Neighbour (KNN) method. KNN is a popular technique for forming classification decisions. The technique involves computing the distance of an input vector $I_{\mathrm{i}}$ from a set of stored training examples $W_{\mathrm{ij}}$. The classification decision is performed on the basis of majority voting of classification of the K nearest pattern in N-dimensional space, where N is the number of features in the pattern. This method forms the basis for memory-based learning, and is referred to as a 'lazy learning technique', where all classification is done during the validation phase (when input data is compared, distance wise, with the stored training data).

The results achieved were a FAR of 0.042 (4.2%) and a FRR of 0.08 (8.0%), which were not very encouraging. The low number of participants and the lack of supervision during the data collection raises internal validity concerns. The results also indicate that summing the 6 possible metrics may be nullifying some of the identifiable characteristics normally obtained by the individual metrics.

Comparing the results of this experiment with previous studies that used ANNs for recognising typing patterns (Brown and Rogers, 1993; Obaidat and Sadoun, 1997), it is again clear that ANNs out perform other pattern recognition methods.

Like the previous study, Bergadano et al. (2002), used one metric to characterise typing samples. However, the authors used trigraphs instead of digraphs. A trigraph is a term used to describe a triplet of typed characters. The duration of the trigraph is measured from the press of the 1st key to the press of the 3rd key. Note that there is only one metric per trigraph (this is essentially the keystroke latency for a three key combination).

There were 44 participants recruited to represent legitimate users of a system. Each provided 5 samples of the same prose, resulting in a total of 220 samples. There were also 110 participants recruited to represent impostors, who attempted to impersonate a legitimate user. Each provided 1 sample of the same prose, resulting

in a total of 110 samples. This gave a total of 330 samples. Each sample comprised 683 characters of prose.

Data collection was conducted over 28.8-Kbaud phone line, with the data capture program residing on a Sun Workstation (which was a server machine for a local area network–LAN). An obvious concern here is that the capture of keystroke event times would be subject to interruptions—due to high traffic loads—and therefore the data captured could not be guaranteed to truly reflect the correct time intervals. Another point of concern is that keystroke event times were measured at a 10 millisecond resolution. A previously mentioned, this is a rather course resolution and is an odd choice for an experiment conducted in 2002.

For each legitimate user, 4 samples were used to form a profile; 1 sample was used to test for false rejection[6], and 325 samples to test for false acceptance[7]. The 325 was made up of the impostor samples and the 5 samples from each of the other legitimate users (i.e. 110 + 215).

The authors claimed to have 220 samples available to test for false rejection (for each legitimate user), and 71,500 samples to test for false acceptance. In order to arrive at this quantity, the preceding process was repeated for each individual sample (for each legitimate user). This meant that the evaluation was performed on the same sample sets five times over, with just a different ordering of the samples each time. The authors accepted that this raised some experimental validity concerns.

The method of analysis involved the calculation of the distance between two samples, based on the duration of the trigraphs that comprise the text. Firstly, all trigraphs were ordered according to their durations, such that trigraphs and their associated duration were stored in ascending order of time in a two column matrix. This was done for any two samples, so let $V$ and $V'$ be the matrices containing the associated trigraphs and durations of the two samples, with dimension $N$x2 (where $N$ is the number of trigraphs). The distance between the position of a trigraph in

[6]Though the authors used the term false alarm rate, their explanation described what most researchers refer as the false rejection rate–FRR.

[7]Though the authors used the term impostor pass rate, their explanation described what most researchers refer as the false acceptance rate–FAR.

$V$ and the position of the same trigraph in $V'$ determined the distance $d$ for that trigraph[8].

A normalised total distance for all trigraphs between both samples—based on the sum of the $d$ distances—was calculated according to Equation 3.1:

$$nd(S1, S2) = \sum_{i=1}^{N} d_i/(N^2/2) \tag{3.1}$$

where $nd(S1, S2)$ is the normalised distance between two samples $S1$ and $S2$, and $N$ is the number of trigraphs[9]. The result of this calculation was termed the 'degree of disorder'.

Prior to testing, the following measures were required:

- The mean of the normalised distances of all samples for a legitimate user, denoted $m(A)$ for a user $A$.

- The mean distance between a query sample and the normalised distances of all other samples for the claimed legitimate user, according to Equation 3.2:

$$md(S, Q) = \sum_{i=1}^{N} nd(S_i, Q)/N \tag{3.2}$$

  where $md(S, Q)$ is the mean distance of all normalised distances (between a query sample $Q$ and all other samples), and $N$ is the number of trigraphs.

When verifying a query sample, the following criteria needed to be met for that sample to be considered to belong to a legitimate user:

1. $md(A, Q)$ was the smallest mean distance.

2. $md(A, Q)$ was closer to $m(A)$ than any other $md(B, Q)$ computed. This can be calculated as shown by Equation 3.3:

$$md(A, Q) < m(A) + |k(md(B, Q) - m(A))| \tag{3.3}$$

[8]So theoretically, $d$ could be determined as a distance value in the scope of $0, 1, \ldots, N-1$.
[9]If $N$ was odd, $(N^2 - 1)/2$ was used as the divisor.

where $B$ is another legitimate user of the system such that $md(B, Q)$ is the second closest mean to $m(A)$ after $md(A, Q)$, and $k$ is a constant[10].

Two other constants $a$ and $b$ were introduced as filter mechanisms, to provide and acceptable balance between the FAR and FRR. The best results achieved were a FAR of 0.0001 (0.001%) and a FRR of 0.04 (4.0%).

Though these results appear better than those achieved by other studies using statistical methods (and even some that used ANNs), there are concerns about the experiment. In addition to the concerns already stated above, the use of only 1 sample per legitimate user (even though it was used 5 times over) to test for false rejection, results in course granularity for that performance variable.

Given these issues, caution could be applied when attributing confidence to the experimental findings. On the positive side, the analysis method was quite innovative and could provide valuable information; perhaps if incorporated with two metrics calculated from digraphs (i.e. keystroke duration and digraph latency) the measurements could be useful.

An empirical study conducted by Monrose et al. (2002), proposed a method of 'password hardening'. Password hardening is a concept where knowledge of the password is only half of the requirement for authentication. The other criteria may be based on keystroke dynamics. Thus, authentication is based on knowledge of the password and how the password is typed by the legitimate user.

Statistics attained from the legitimate user's typing pattern are used to encrypt the password; only the ability to type the password in the same manner as the legitimate user will be successful in decrypting the encrypted password. Whilst this approach is very different to that of the current investigation, valuable information can be gained from this study.

The authors demonstrated the entropy or randomness of keystroke features among 20 participants typing samples. They also determined the average number of distinguishable features obtained from a password of 8 characters. All partici-

[10] A factor $k = 0.5$ simply requires that $md(A, Q)$ is closer to $m(A)$ than any other $md(B, Q)$, whereas $k = 0.66$ enforces less stringency and $k = 0.33$ requires stronger evidence of $Q$ belonging to $A$.

pants entered the same password at least five times each. For different values of a constant $k$, the entropy of the metrics and the corresponding average number of distinguishable features were identified.

For example, for $k = 0.1$, the entropy achieved was 10.5, which is the maximum attainable for 20 users. The average number of distinguishable features for that constant value was 14 out of a possible 15.

While these facts were based on a study with only 20 participants, and the application to which they were applied was different to the current study, it did re-affirm that keystroke patterns are quantifiable and distinguishable between different people.

Yu and Cho (2004), identified three factors which they believe may affect the capability of ANNs to recognise typing patterns in a keystroke dynamics authentication system (and thus possibly produce erroneous results):

1. In a real-world situation only legitimate users' data are available in advance (for training purposes), as it is impossible to gather data from all prospective impostors. For ANNs to be trained effectively, they require both positive and negative case input data. In order to overcome this perceived liability, the authors adopted the use of support vector machines (SVMs)[11], which require only positive case input data for training purposes.

2. Obtaining enough training data is problematic. In research, and in a real-world situation, it is unlikely that users would appreciate entering hundreds of samples of the required text. The authors adopted the ensemble method to generate extra data from existing data.

3. Raw data contains much noise and variability (Cho et al., 2000). Pre-processing attempts to eliminate outliers by selecting relevant subsets of features (ignoring redundant or erroneous features). In many cases this is done manually. However, an authentication system requires automation of such a process.

[11]SVMs use supervised machine learning techniques based on the concept of decision planes that define decision boundaries. A decision plane separates a set of objects having different class memberships. For problems that can not be linearly separated in the input space, SVMs attempt to find a solution by making a non-linear transformation of the original input space into a high dimensional feature space, where an optimal separating hyperplane can be found (Rychetsky, 2001).

The authors made use of genetic algorithms (GA) to select relevant subsets of features[12].

The authors concern stated in point 1 above, is a matter of perspective. As stated by the authors, it is impossible in any real-world situation to collect samples from all prospective impostors. However, depending on the situation in which the authentication system is to be employed, this may or may not be a real problem. For a home computer situation (where there are possibly only 3 or 4 legitimate users), the point is quite relevant; it would be very difficult to obtain enough input data to effectively represent an entire population of impostors. In contrast, for a system with multiple users (that is, small to large companies or corporations), this is less likely to be an issue.

In relation to point 2 above, it is unlikely—in research or in a real-world situation—that collecting enough data for training an ANN would be a real problem (for a multiple user system). In either situation the number of samples required from participants should not be unreasonably high. Introducing methods to generate synthetic data sets seems unnecessary.

In relation to point 3 above, there are numerous methods—statistical and machine learning—that could be employed for feature selection (to reduce data noise). Statistical packages (such as SPSS and R) are available that accept internal commands for achieving the same outcomes that most users achieve via a graphical user interface. These packages could be integrated into an automated authentication system, as easily as any machine learning technique.

For their experiment, the authors recruited 21 legitimate users who provided between 150 and 400 samples (on average 275) of their password (for training purposes), and then provided 75 samples for testing purposes.

Passwords were between 6 and 10 characters in length. So on average, legitimate users could be said to have provided 350 (275+75) samples. Fifteen impostors prac-

[12]GAs are a search technique used to find (exact or approximate) solutions to search problems by mimicking the process of natural evolution. They are a class of search meta-heuristics, that use techniques such as inheritance, mutation, selection, and re-combination (crossover), which are known in the field of evolutionary biology (Reeves, 2003).

ticed typing the passwords, and then provided 5 attack attempts for each password (that is, 75 impostor samples per password).

Analysis was performed by a genetic algorithm and support vector machine (GA-SVM) combination. Using a wrapper approach, the GA-SVM iterated until the most relevant subset of features—determined by the genetic algorithm—achieved the best classification outcome (utilising the support vector machine).

The experiment returned best results of FRR of 0.0369 (3.69%) when the FAR was forced to 0.0. These result are comparable with other research efforts already discussed, that employed machine learning techniques. However, the steps the authors took to overcome what they saw as deficiencies with ANNs, entailed additional processing steps to achieve essentially similar accuracy. Any unnecessary processing opens up the possibility for introducing error.

In the preceding experiments it is clear that participants were required to type text of vastly different lengths for each sample they provided. Depending on the focus of the researchers, some used prose of many words, some used a phrase of only a few words, and some used a username or a password or both. This means that the data sets for analysis also varied in length.

Little research had been done to investigate the optimal length text that would return the best verification accuracy under the same conditions, although Bleha and Obaidat (1991) stated that character strings with length less than 10 were prone to great variability and loss of accuracy. In an attempt to bring consistency to experimentation in this field, Abernethy et al. (2004), conducted an experiment to determine the optimal text length (for best verification accuracy), when using ANNs for classification. This was a preliminary study to the one reported in this dissertation, carried out by the author and his collaborators.

The experiment had 50 participants provide 50 samples each of a 32 character string (in 5 collection sessions), though only 40 samples were used in the experiment; the first ten allowing participants to become familiar with the phrase.

The string was a derivative of the familiar typist training sentence “the quick brown fox jumps over the lazy dog”. For the convenience of participants, this was

shortened to "brown foxes jump over lazy dogs". Note that these are all lowercase characters. Timing resolution was 1 millisecond, and the metrics used were the keystroke duration and digraph latency.

Twenty five participants were randomly selected for the training group, and the remaining 25 were assigned to the non-training group. Of the 40 samples per training group member, 30 were randomly chosen for the positive training case, and 2 samples from each of the other training group members were randomly chosen as the negative training case. Thus, each training group member had an input training file of 78 samples. These training files were used to derive the input training sets for each character length (from 2 to 31).

Classification was performed by the Multi-Layer Perceptron back-propagation neural network. An ANN was used to classify each character length, for each training group member. The number of middle layer neurons per ANN was varied, so that the configuration returning the best accuracy could be determined.

When testing for false rejection, the mean of the ANN outputs—for the positive acceptance tests—was determined. Analysis was based on this mean less a confidence level of 0.05. For the tests for both performance variables (FAR and FRR), if a score was above the mean minus the confidence level, it was considered a match. Otherwise, it was considered a non-match.

The results indicated that the optimum length was fifteen characters. At that length a FAR of 0.0119 (1.19%) was achieved, with a FRR of 0.108 (10.8%). Though the results were not as accurate as some previous experiments involving ANNs—for example, (Brown and Rogers, 1993; Obaidat and Sadoun, 1997; Cho et al., 2000)—they did indicate that a string of approximately 15 characters could be recommended to attain an acceptable accuracy.

A limitation with this study was that only 10 samples, for each training group member, were available to test false rejection (after 30 samples were used for training). This resulted in a course granularity in the FRR scores[13].

Also the mean (calculated from the outcomes of the 10 false rejection tests) which was used in determining classification, may have impacted on the FAR; that is, the

[13] 1 rejection resulting in an FRR increase of 0.1 (10%).

FAR may not have been truly representative of the classification accuracy achieved by the ANNs.

Jiang et al., (2007) proposed the use of Gaussian Modeling and Hidden Markov Models (HMMs) to analyse users typing patterns[14]. Gaussian modeling and the maximum likelihood estimation were used to determine the sample mean and variance for the distribution of keystroke times. These values were used as parameters by the HMM. Using the forward algorithm, the HMM predicted the probability that a query typing sequence matched that of the reference sequence.

For their experiment, the authors recruited 58 participants to type 20 samples each of a username and password. All 58 made 15 attempts to authenticate themselves (thus 870 legitimate user tests were performed). Another 257 volunteers provided 28 attempts each to authenticate each of the 58 legitimate participants (that is, a total of 3,528 impostor tests were performed).

The authors presented their results graphically (refer Figure 2 of their publication) to demonstrate the equal error rate (EER), false acceptance rate (FAR), and false rejection rate (FRR). The FAR and FRR could be independently reduced by adjusting the threshold (though this would impact on the other performance variable). The graph illustrated results at threshold values between 0.2 and 3.6, and the following approximate information can be extracted.

The best FAR of 0.0 was first achieved at threshold 3.0, however the FRR at that threshold was approximately 0.2 (20.0%). The best FRR of 0.0 was first achieved at threshold 1.0, however the FAR at that threshold was approximately 0.22 (22.0%). At the threshold 1.5, an equal error rate (EER) of 0.0254 (2.54%) was achieved; this means that both the FAR and FRR achieved this rate. Consequently, these figures were recorded in Table 3.2. A more acceptable FAR of approximately 0.01 (1.0%) and FRR of approximately 0.05 (5.0%) was attained at the threshold value of 1.8.

Revett et al., (2007) experimented with, what they called, primary and secondary metrics obtained from user input of a username and password (from 6 to 15 characters). Primary metrics are those directly obtainable, and included the time taken

[14]Hidden Markov Models are statistical models that can be used to characterise sequential data such as keystroke times.

to enter the username, the time taken to enter the password, and the total time for the entry of both username and password. The secondary metrics are derived values that are calculable from raw keystroke event times, and included digraph and trigraph latencies, typing speed, and edit distance.

For their experiment, they recruited 50 participants (20 legitimate users and 30 impostors). There were 10 samples maintained for each of the legitimate users; samples were collected 3 times daily for 14 days, and the metrics from the latest 10 samples were used to update the metrics in the 10 maintained samples. The 30 impostors attempted to login to the legitimate user accounts, with each impostor attempting 4 logins for every legitimate user (i.e. 80 login attempts by 30 impostors provided 2,400 login attempts). However, only 2,000 attempts (by impostors) were randomly selected for use in the evaluation. Legitimate users attempted to login to their own accounts 100 times. Therefore, there were 2,000 attempts by legitimate users used for evaluation (20 x 100).

Classification was performed using the Probabilistic Neural Network (PNN) and Multi-Layer Neural Network with back propagation (MLNN-BP). The PNN is fundamentally a neural network implementation of a Bayes-Parzen classifier (Revett et al., 2007); it determines class membership of multivariate sample data based on a set of measurements (in this case, the primary and secondary metrics).

Experimental results of concern for this review are those that include the use of all metrics—primary and secondary—using both the PNN and the MLNN-BP for classification[15]. The authors reported results as an average of the FAR/FRR performance variables. No attempt was made to differentiate between the two, even though most researchers typically do so in order to compare these variables with other research results. Also, considering the variables separately facilitates adjusting the error rates—via a decision threshold—usually to reduce the FAR error.

Results were an average FAR/FRR of 0.057 (5.7%) for the MLNN-BP and an average FAR/FRR of 0.039 (3.9%) for the PNN. In order to register the FAR and FRR results separately in Table 3.2, the rate reported for the PNN (being the better result) has been halved, because it was the average of the two performance variables.

[15]Results were also reported for the individual metrics, but were not considered for this review.

Thus for comparison with other research, it is estimated that the FAR and the FRR achieved 0.0195 (1.95%) respectively[16].

The metrics used in the experiment seems to be the main significant difference between this research and others that have used ANNs, as most other aspects of the experiment were comparable with accepted practices. Given the approximated rates above, it seems that the use of primary and secondary metrics did not demonstrate any improvement over the keystroke duration and digraph latency.

A study performed by Hu et al., (2008) utilised the normalised degree of disorder of tri-graphs, but extended the concept to a generalised degree of disorder for $n$-graphs. This was an extension to the original method proposed by Bergadano et al., (2002). The authors adopted the KNN approach for classification, instead of the statistical approach taken by Bergadano et al., (2002).

To improve performance, the calculation of a measure $A$ was determined. $A$ was a similarity measure of the number of $n$-graphs that have similar durations between two samples. Similarity of durations between $n$-graphs from two samples was measured according to Equation 3.4:

$$1 < \frac{max(d1, d2)}{min(d1, d2)} \leq 1.25 \tag{3.4}$$

where $d1$ and $d2$ were the durations associated with the same $n$-graph from both samples.

The normalised $A$ measure between two samples was calculated according to Equation 3.5:

$$A_n(S1, S2) < 1 - \frac{number\ of\ similar\ n-graphs\ shared}{total\ number\ of\ n-graphs\ shared} \tag{3.5}$$

The similarity measure $A$ was used along with the generalised degree of disorder metrics mentioned above.

A profile or template for legitimate users was formulated by determining the average metric for $n$-graphs (from all samples for that user)[17]. The KNN method

[16]Following the same rationale, the MLNN-BP could be attributed with achieving a FAR and a FRR of 0.0285 (2.85%).

[17]Analogous to the measure $m(A)$ defined by Bergadano et al., (2002).

was used to cluster the templates, of average metrics, for all legitimate users. Cluster sizes were controlled by an experimentally determined threshold.

Using measures defined by Bergadano et al. (2002), a query sample was confirmed as belonging to a legitimate user if:

1. The $A$ measure was within the same cluster as the legitimate user $X$.

2. $md(A, X)$ was the closest value to $m(A)$ within the same cluster.

For the experiment, 19 participants (designated legitimate users) provided 5 typing samples each. Another 17 (designated impostors) provided 27 typing samples. The text that participants were required to be typed was stated as being complex and not easily typed. Data collection was unsupervised.

There were 2,223 samples used to test for false acceptance, whilst each of the legitimate users' 5 samples were used to test for false rejection. The results were a FAR of 0.00045 (0.045%) and a FRR of 0.0. These are excellent results, though the small number of samples available for testing false rejection means that the granularity of the corresponding rate (FRR) was very course. Also, the small number of legitimate users (whose data was used for training purposes) could affect confidence in the ability to generalise based on these results.

The next section presents a very brief review of some experimental work where dynamic verification was investigated.

### 3.4.2 Dynamic Verification

The research findings and methodological issues relating to keystroke dynamics experiments involving dynamic verification, as conducted by the authors of the papers reviewed, are summarised in Table 3.3.

Leggett et al. (1991) based their investigation on previous research (Umphress and Williams, 1985; Leggett and Williams, 1988), but extended the concepts to explore the idea of dynamic verification. They attempted to improve the accuracy achieved in the previous studies, whilst developing their continuous monitoring application.

| Reviewed Paper | Number of Participants | Samples Per Participant | Collection Sessions | Text Length | Resolution in Milliseconds | Metrics | Analysis | FAR | FRR |
|---|---|---|---|---|---|---|---|---|---|
| Leggett et al., 1991 | 36 | 2 | 2 | 537 chars | 10 | 1 | DSC | 0.1111 | 0.1281 |
| Maher et al., 1995 | 67 | 2 | 1 | 168 chars | 1 | 1 | DSC | n/app | n/app |
| Monrose and Rubin, 1997 | 42 | n/avl | n/avl | 3 sentences | n/avl | 2 | SPR | n/app | n/app |
| Monrose and Rubin, 2000 | 63 | n/avl | n/avl | 32 chars | n/avl | 2 | SPR | n/app | n/app |

| LEGEND | DESCRIPTION |
|---|---|
| n/avl | Not Available |
| n/app | Not Applicable |
| DSC | Deterministic Statistical Calculations |
| SPR | Statistical Pattern Recognition Techniques |

Table 3.3: Summary of Reviewed Literature Involving Dynamic Verification

Their experiment used the very same data as that collected by Leggett and Williams (1988). They had 36 participants provide one sample each over 2 collection sessions. In both sessions participants entered 537 characters of prose; the first set was used in determining the reference template, while the second set was used as test data. Keystroke events were captured at a resolution of 10 milliseconds, and keystroke latency was the metric used. They applied the same filters to the data as did Leggett and Williams (1988).

The difference was the analysis method applied to the data. A reference template was obtained by storing the frequency, mean and standard deviation of each digraph in a 26 x 26 matrix (letters that make up the digraph were used as indices of the matrix). Templates obtained from test samples were then compared with the reference template by considering the next keystroke and time values and applying sequential statistic theory to compare digraph times. The results were less than encouraging, with a FAR of 0.1111 (11.11%) and a FRR of 0.1281 (12.81%).

Mahar et al. (1995) critically reviewed the experimental methods, particularly the statistical analysis, used in the studies by (Umphress and Williams, 1985; Leggett and Williams, 1988; Leggett et al., 1991) and prepared an improved experimental design. They had 67 participants provide samples in 1 collection session. Each participant was required to twice type 30 high frequency words and 6 other words (their name or a word of similar length). There were 10 words comprising two characters, 10 words comprising four characters, and 10 words comprising six characters. The length of the name (or equivalent) was not stated, but if an 8 character name is assumed, the total number of characters per sample would be approximately 168.

The objectives of the study were to validate the aforementioned deterministic statistical analysis methods by testing three key factors in the analysis that may have affected the verification accuracy achieved. Firstly, it was determined that there was a marked heterogeneity of variance in the latency with which participants type different digraphs, and this variance exerted an effect on the accuracy of identity verification.

Mahar et al. (1995) showed that the overall standard deviation, used in the previous experiments as a measure of tolerance, was inappropriate for making the final verification decision. Instead, they demonstrated that a digraph-specific standard deviation estimate improved accuracy.

Secondly, in the previous research, the maximum allowable separation between test and reference digraphs was set at 0.5 standard deviations, and the minimum proportion of test digraphs required to pass the maximum allowable separation test, before verification was assumed, was 60%. Mahar et al. (1995) indicated that a minimum proportion of 70% of test digraphs, with a maximum allowable separation of 1.5 standard deviations gave improved accuracy, when compared with the previous experiments.

After establishing the above results, the correctness of applying the upper time limit filter of 500 milliseconds per digraph (as used by Leggett and Williams (1988)), was tested. The results confirmed that this upper limit would *on average* yield optimal verification rates. However, considering the emphasis on the term "on average", and the fact that the Leggett and Williams (1988) experiment improved only a little on the results achieved by Umphress and Williams (1985) (who applied on upper limit of 750 milliseconds), the upper limit recommended here could not be considered definitive (but a recommendation only).

Studies by Monrose and Rubin (1997) and (2000) continued the investigation of dynamic verification using statistical techniques. The two metrics used were the keystroke duration and digraph latency, rather than just the one metric—keystroke latency (refer section 3.3).

The first study had 42 participants type text of approximately 3 sentences. Information about the character length of the sentences, the number of samples provided per participant, and the number of collection sessions was not made available in their paper.

In the second study, 63 participants provided typing samples, but no information on the character length of samples was provided. We may assume that as the experiment followed the methodology of that used by Joyce and Gupta (1990),

that samples were approximately 32 characters in length. As with their previous experiment, no information was provided on the number of samples per participant or the number of data collection sessions. It was stated that data collection was unsupervised. This fact and the lack of information about data collection, raises some validity concerns.

The two investigations were for the purpose of identification rather than verification. Although the authors believed that the classifiers used could achieve similar accuracy for both purposes, this was not demonstrated.

A similar statistical method used by Joyce and Gupta (1990) for attaining the reference templates was used in these studies, although a different classification approach was adopted. In the first study, three classifiers were used; in the second study, a fourth classifiers was used in addition to the three from the first study.

The four classifiers were: the Euclidean distance measure; the Non-Weighted probability; the Weighted probability; and the Bayesian classifier. In the first study, the best results were achieved with the Weighted probability classifier, where correct identification occurred with 90.7% accuracy. In the second study, the best results were achieved with the Bayesian classifier, where correct identification occurred with 92.14% accuracy.

The results of the experiments just discussed did not achieve desirable accuracy. Also, there were concerns about various aspects of the experiments that did not promote confidence in the results. This may explain why investigations into dynamic verification are not common.

The next section summarises the issues related to keystroke dynamics research, and discusses why this biometric characteristic was used for experiment in this study.

## 3.5 Summary of Keystroke Dynamics

The preceding review provides the following information about keystroke dynamics:

- Keystroke dynamics has been empirically investigated since approximately 1980 (a period of 30 years at time of this dissertation).

- Though early results in this field of investigation were not as accurate as other biometrics characteristics (when used for authentication purposes), much improvement has been demonstrated since 1993 (with more promising results achieved).

- The purpose of an experiment, in relation to data collection, is crucial. For example with static verification, the text (prose, password, or phrase) upon which analysis is to be performed, affects data collection decisions. The length and composition of the text (including any filtering) are also important. The number of samples to be collected from each participant needs to be determined. Note that the number of samples collected may be influenced by the classification method chosen.

- There have been a number of methods utilised for analysis or classification of typing patterns. The approach by early researchers, using deterministic statistical calculations, has been surpassed by more modern pattern recognition techniques. In particular, the use of ANNs has been shown to be the most effective, though there are operational considerations that require studious attention.

- No common standards in conducting experiments involving keystroke dynamics have been articulated. However, this is also true of other biometric characteristics.

Therefore, certain preliminary decisions need to be made, as early as possible, when implementing a biometric system that incorporates keystroke dynamics:

- Will the text be composed of prose, a password, or a phrase? If a password is used, the length is usually minimal (between 6 and 8 characters) and may affect accuracy (Obaidat and Sadoun, 1997). If a phrase is used, between 10 and 15 characters should provide the required accuracy (Obaidat and Sadoun, 1997; Abernethy et al., 2004).

- Will the text be chosen by the participants (as in a password), or will it be composed of particular character combinations as suggested by Gaines et al. (1980)?
- Will filtering be applied to the collection of data? If so, what upper time limit between digraphs will be imposed? Are correctly typed samples required (i.e. erroneous samples disregarded), or does the method allow for erroneous input? Are uppercase and lowercase characters allowed in the text, or is the text restricted to lowercase only.
- Once data has been collected, will filtering be applied to the raw data prior to classification. As demonstrated in the review of literature, the more recent research efforts apply filtering at this stage in acknowledgement of the accepted variability of raw keystroke dynamics data.
- What method of analysis or classification will be used?

Some researchers maintain that keystroke dynamics remains a difficult task when attempting to attain the accuracy comparable to that achieved by physiological biometric characteristics, such as fingerprints and retinal scans, because keystroke dynamics is a behavioural characteristic with high variability and is unstable (Bergadano et al., 2002). It is undeniable that keystroke dynamics exhibits more variability and cannot attain accuracy comparable to that achieved by fingerprints and retinal scans.

In many of the empirical investigations discussed in this chapter, researchers designed experiments where (a small number of) participants were required to enter a small number of samples comprising hundreds of characters of prose. This practice disregards the definition of keystroke dynamics, where the emphasis is on *habitual* typing pattern. This means that the effectiveness of keystroke dynamics is achieved when users type many samples of text that is familiar to them, and by that repeated entry, the users develop a habitual style of typing that text (which is highly distinguishable from other users who type the same text). As demonstrated in the review, the text only needs to be between 10 and 15 characters in length (Obaidat and Sadoun, 1997; Abernethy et al., 2004).

Though not as accurate as some other biometric characteristics (such as fingerprints and retinal scans), if treated carefully and correctly, keystroke dynamics can be a valuable contributor to an authentication system (particularly if used in a multi-modal authentication system). It offers the following advantages that the more accurate characteristics can not, whilst still maintaining an acceptable level of accuracy:

1. Current computer authentication systems, using traditional methods, require users to enter their username and password on a keyboard. So, computer users are already accustomed to authentication based on keyboard interaction.

2. The more accurate biometric characteristics may not be accepted by users. For example, retinal scans are highly intrusive (refer Chapter 2 section 2.2.4.4); people may be reluctant to go through the required procedure on a regular daily basis. Fingerprints are commonly associated with criminal investigation and legal proof of identity, so people may be wary of providing their fingerprints with the knowledge that they will be stored in a database and could be stolen or mishandled. Keystroke dynamics does not suffer from these problems of acceptability; computer users type regularly as part of their interaction with the computer, and so would presumably find this more acceptable as a basis for regular authentication. Keystroke dynamics does not have the same perceived association with the criminal element that fingerprints do.

3. Keystroke dynamics is cost effective; it requires no additional expensive hardware and/or software that other characteristics require, and can be implemented and maintained for minimal cost. For authentication systems associated with Web based applications, this is particularly pertinent. It would place an undue burden on consumers, to expect them to supply the hardware/software required for authentication by many of the physical biometric characteristics.

4. Even if one doubts the capability of keystroke dynamics to provide accurate authentication for a uni-modal biometric system, the characteristic may still be

advantageous in a multi-modal biometric system. Because of its ease of implementation and cost effectiveness (compared to other characteristics), keystroke dynamics could be given serious consideration as a viable option for such a system, even though it may demonstrate more variability than other biometric characteristics. System parameters could be employed to counterbalance the variability associated with keystroke dynamics; thus reliability and robustness may be attained.

Keystroke dynamics has been chosen for this experiment for the above reasons, and the issues mentioned in this summary were given careful consideration. The details are discussed in Chapter 5 (Research Method).

## 3.6 Conclusion

In this chapter, the conceptual basis for keystroke dynamics was provided in section 3.2. Section 3.3 described the metrics used in keystroke dynamics research, including the method of calculating the metrics.

A large portion of the chapter was dedicated to a review of keystroke dynamics related research in section 3.4, which covered both static and dynamic verification approaches (sections 3.4.1 and 3.4.2 respectively). This demonstrated that keystroke dynamics can be used with confidence, and can be a valuable tool in the system security armoury.

Finally, section 3.5 summarised the advantages and concerns associated with keystroke dynamics. The summary included reasons for choosing keystroke dynamics for this experiment, and highlighted the issues of concern that need careful consideration when using this biometric characteristic.

# Chapter 4

# Fingerprint Recognition

## 4.1 Introduction

This chapter provides a discussion of the biometric characteristic known as fingerprint recognition. The overview (section 4.2) provides some background and history of the subject area.

Section 4.3 provides a description of the inherent features (distinguishing characteristics) associated with fingerprints, both on a global level (section 4.3.1) and a local level (section 4.3.2).

With today's computer technology, fingerprint recognition is generally implemented as an Automated Fingerprint Identification System (AFIS). Section 4.4 lists and explains the stages of an AFIS, and includes an overview of fingerprint classification (section 4.4.5) and verification (section 4.4.6).

Section 4.5 reviews research efforts that have employed minutiae-based matching methods of verification; the method most related to the current study. This provides some detail of the techniques used, and also serves as a basis for comparison of the experimental results from the current study with those of previous studies.

Finally, 4.6 summarises the main tasks involved in a minutiae-based matching approach, and highlights the different approach adopted in the current experiment (and the reasons for this different approach). Section 4.7 then provides a conclusion to the chapter.

## 4.2 Overview of Fingerprint Recognition

A fingerprint is produced when the bulbous region of the distal phalanx (of any finger or thumb) makes contact with another surface, thus creating a duplicate impression of the existent characteristics of that finger tip (Faulds, 1880; Galton, 1892). Better quality fingerprints are obtained when the surface is smooth and flat. However, fingerprints are obtainable from course and uneven surfaces; though typically quality is compromised.

The most prominent characteristics of a fingerprint impression are caused by the papillary ridges (and the consequent valleys or furrows) of the epidermal layer of the finger. Figure 4.1 provides an example fingerprint illustrating the ridges and furrows, which form a pattern (known as the ridge pattern) that is distinguishable to the naked eye. There are also minute characteristics of the individual ridges (known as minutiae) that are not as easily distinguishable by the naked eye.


Figure 4.1: Fingerprint Impression Illustrating Ridges And Furrows

Ridge characteristics are discussed in more detail in section 4.3. Their significance for identification and verification are discussed in sections 4.4.5 and 4.4.6 respectively.

Historically, there is archaeological evidence that ancient civilisations used fingerprint impressions in clay or on transcripts to seal deeds, contracts of loan and other transactions. Cummins (1941) has suggested that the primary intent of these impressions was to bear witness to the terms of agreement.

The use of fingerprints as a form of identification is a contrasting concept, having its origin in the late 19th century, for the purpose of maintaining a correct record of criminal identities. Law enforcement seeks to establish the true identity of criminals, and thus maintain the integrity of criminal record systems. This is as true today as it was in the late 19th century. However in the 21st century, verifying identity has also become an integral part of authentication systems on computers and computer networks.

The earliest system of identification adopted by law enforcement agencies in the 'western world', was the anthropometric system introduced by Alphonse Bertillon in 1883 (Fosdick, 1915). This system, based on measurement of various parts of the anatomy, was used by countries such as France, England, Germany, Austria, Russia, Switzerland and parts of the United States of America until the second decade of the 20th century (Fosdick, 1915), as either the primary or secondary system of identification.

As an official identification system it met with some success, but suffered from the following limitations (Fosdick, 1915):

- For permanence reasons, measurement required the criminal to be of full physical maturity.
- The measurement instruments were intricate in design and their precision deteriorated in a short period of time with any inappropriate handling.
- The collection of accurate measurements required specialised training, which was not always available because of the small number of qualified trainers.
- The system involved collection of many body measurements, and was therefore more onerous than the simpler collection of fingerprints.

Because of these limitations, and the growing evidence that fingerprints provided a more accurate and manageable alternative, fingerprints became the preferred primary method of identification by the first decade of the 20th century (Fosdick, 1915).

The first reporting of the basic components of fingerprints (ridges, furrows, and pores) was in 1684 by Nehemiah Grew (Grew, 1684). However, it was not until 1823 that J.E. Purkinje formally specified nine categories or classes of basic ridge patterns (Cummins and Kennedy, 1940). Purkinje presented hand drawings and descriptions of these nine classes as part of his dissertation *"Commentatio de examine physiologico organi visus et systematis cutanei"*. These classes will be discussed (in section 4.4.5) in relation to the classes used for ridge pattern classification today.

It should be noted that neither Grew nor Purkinje made any supposition as to the individuality of the configuration of ridge patterns nor did they allude to the permanence of these patterns (Cummins and Kennedy, 1940). In fact, their existence was only recognised in relation to their biological characteristics; there was no speculation that they could be utilised as a form of identification.

The earliest known publication proposing the possible individuality of fingerprints and their use for identification was published by Henry Faulds in 1880 based on his empirical observations (Faulds, 1880). His collection method was achieved by spreading printers ink on a smooth flat surface, pressing the desired finger evenly onto the inked surface, and then onto slightly dampened paper.

Faulds (1880) proposed the possibility of the permanence of ridge patterns, though the experimental methodology—in relation to the quantity of fingerprint samples collected and the basis upon which comparison of fingerprint impressions was founded—was limited. Further information about his methodology became available in a later publication (Faulds, 1905), after other researchers had published their findings in this field (Galton, 1892; Henry, 1900).

Sir William Herschel maintained that he began examining the characteristics of fingerprints in 1858, whilst posted as a magistrate to Indian, continuing until his retirement in 1879 (Herschel, 1916). In 1862, he strongly recommended that the

Government of Bengal employ the use of fingerprints to validate claims before the courts. In 1877, whilst magistrate and controller of criminal courts, jails, and registration of deeds, Herschel implemented the use of fingerprints for non-repudiation purposes.

It should be noted that this use of fingerprints was implemented as a form of identification. Criminals before the court had their identity formally confirmed (based on fingerprints held on record), so that no other person could be substituted for the real criminal (as apparently was common practice at the time in India). Like Faulds, Herschel did not publish a full account of his methodology (Herschel, 1916) until after other researchers had published their findings in this field (Galton, 1892; Henry, 1900).

Sir Francis Galton (who accredited Sir William Herschel for his inspiration and early fingerprint samples) specified three major classes of ridge patterns, with each class containing possible variations (Galton, 1892). The major classes were the arch, the loop, and the whorl (refer section 4.4.5). He developed the first systematic approach to fingerprint classification by specifying a method of indexing fingerprints, based on the class divisions, to facilitate searching a collection of fingerprints for a particular print.

More importantly, Galton established the existence of the minute ridge characteristics known as minutia points or minutiae (section 4.3.2). He proposed that the configuration of these features is individual to each digit for every person, and that they remained persistent throughout ones lifetime. It is this property of fingerprints that provides the means to recognise discernible configurations or patterns and thus verify identity based on their distinctiveness.

Galton also noted the existence of the pores along the papillary ridges, and like Faulds before him, recognised that the bodily secretions through these pores left a residual fingerprint (on any surface that the finger came into contact with) of good enough quality for identification. The secretions of the pores on papillary ridges facilitates the collection of latent fingerprint used extensively in law enforcement (latent fingerprints are discussed in more detail in section 4.4.1.2).

In 1891, Sir Edward Henry was appointed Inspector General of Police in Bengal, India. In 1897, he introduced fingerprints as an auxiliary form of identification to the then utilised Bertillon system (Polson, 1951). By 1899, fingerprints were given preference as the primary system of identification and the Bertillon system was phased out of use. In 1901, Sir Edward Henry, then the Assistant Commissioner of the Metropolitan Police in London (in charge of criminal investigations), introduced what is now commonly referred to as the Henry system of identification, based entirely on fingerprints (Polson, 1951).

Henry extended the work of Galton and established a classification system which was the first to incorporate the use of core and delta points for fingerprint classification (Henry, 1900). He also established an indexing system that was practically implementable. Henry's system of identification forms the basis, with refinements and/or modifications, for the methods that are used by most law enforcement agencies in the world today (Polson, 1951).

Since the time of the first introduction of fingerprints for identification by Sir Edward Henry, much research and advancement has occurred. These developments will be discussed in the following sections. However, as the aforementioned proposals rely on the uniqueness of fingerprints (between different people), it would advantageous to explore this assumption. The next section discusses this issue.

### 4.2.1 The Uniqueness of Fingerprint

Since the time of Galton (1892), it has been widely accepted that every human finger has unique local features, different from that person's other fingers, and those of every other person.

Law enforcement agencies in particular are keen advocates of the acceptance of this 'fact', as it forms a basis for placing criminals at crime scenes. However, this so called 'fact' has been based on empirical observation only. In recent years, some researchers and law enforcement experts have expressed doubt as to the veracity of this accepted 'fact', as it has never been scientifically proved (Specter, 2002).

Whilst it is considered extremely improbably that two matching fingerprints could belong to two different persons, there have been cases in recent years that caste sufficient doubt on the assumption that it is not possible (Specter, 2002). A contributing factor in these cases is that convictions have been achieved based on the collection, from the crime scene, of only one fingerprint. As there has been no scientific proof that two matching fingerprints from different persons could not occur, basing a conviction on only one fingerprint is considered by some as lawfully dangerous. Also, as mentioned in section 4.4.1.2, conditions at a crime scene do not always favour the collection of good quality or complete latent fingerprints that can then be used to accurately identify a perpetrator.

The doubt over this issue has come up against strong opposition from the majority of those in law enforcement, because it would caste doubt on a well established presentation of evidence in criminal court cases, and bring into question convictions achieved based solely on fingerprint evidence.

Both parties agree on this point:

> **It is an acceptable practice to use latent fingerprints to help identify criminals.**

However, they differ on this question:

> **Is it acceptable to suggest that the latent fingerprint/s could *only* have come from a particular suspect, and to use this as undeniable proof to gain a conviction?**

It does seem unlikely, with the size of the human population, that there could ever be undeniable scientific proof of the uniqueness of each individual fingerprint. However, with research specifically targeted to investigate the hypothesis, there could be very strong inferred evidence gained. For now the impasse continues, however, with perhaps less attention or urgency since the development and wide adoption of DNA testing.

The next section discusses in detail the characteristics or features that allow fingerprint recognition to be utilised for identification and verification.

## 4.3 Fingerprint Features

The epidermal layer of the distal phalanx of a finger is covered with concentric raised friction ridges (Digital Persona, 2004). As well as helping us to grip objects, these ridges provide distinct characteristics or features. The ridges are formally known as papillary ridges because minuscule perspiration pores are prolific along them (Inbau, 1934). For the purpose of recognition, fingerprint features can be broadly classified into two categories: global and local.

### 4.3.1 Global Features

Global features are those fingerprint characteristics that are visible to the naked eye (Digital Persona, 2004). The following is a description of global fingerprint features, some of which are illustrated in Figure 4.2:

- Pattern area: the region of the fingerprint where ridge lines form highly distinguishable shapes or patterns and are clearly apparent. Ridge lines in these regions tend to exhibit high curvature.

- Basic ridge pattern: the discernible patterns made by ridge lines that have been defined into categories or classes. They are located within the pattern area, and are broadly classified as: arch, loop, and whorl. These basic ridge patterns are described in section 4.4.5.

- Core point: the upper most point (in relation to the tip of the finger) of the inner most ridge line (Henry, 1900). Core points are typically (though not always) located near the centre of the pattern area. Figure 4.2 provides an example of a core point. Core points (if present and determinable) can be used as a reference point for determining class (refer section 4.4.5), to assist in fingerprint image alignment, and also to facilitate ridge counting.

- A delta point may be formed in two ways (Henry, 1900):

    1. When a single ridge abruptly bifurcates into two, and the two diverging ridges depart in opposite directions (refer Figure 4.2 for an example).

2. When two ridges that had previously been running side by side abruptly diverge into opposite directions.

Along with the core point, a delta point (if present and determinable) can be used as a reference point for determining class (refer section 4.4.5), to assist in fingerprint image alignment, and also to facilitate ridge counting.

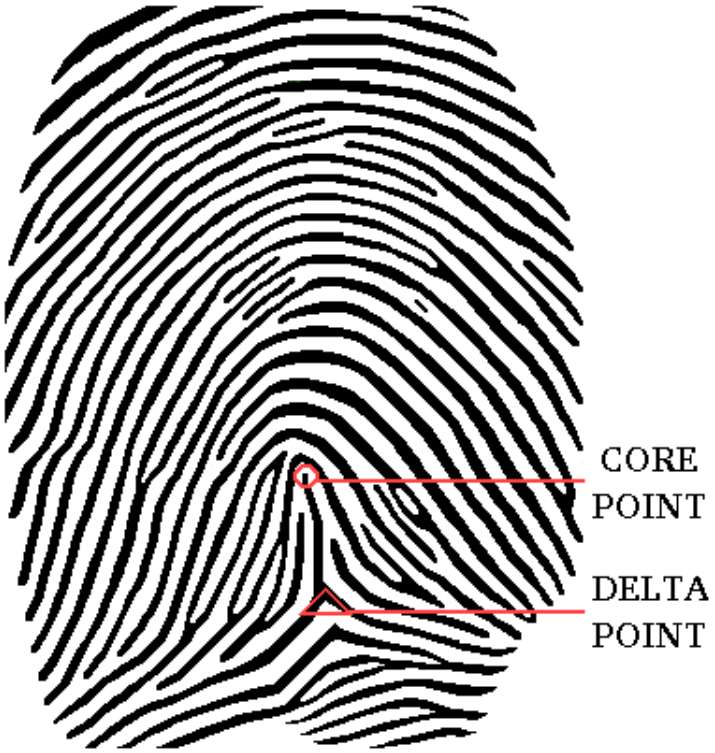


Figure 4.2: Fingerprint Impression Illustrating Core And Delta Points

- Ridge count: the number of ridges crossing the imaginary line segment between a core point and a delta point.

- Minutiae count: the total number of minutia points (refer section 4.3.2).

Because global features are more easily detectable than local features, they are commonly used to classify fingerprints into general categories or classes (Wayman et al., 2005). These categories are based on the basic ridge patterns (section 4.4.5). Categorising fingerprints allows for efficient identification, within large record systems, during the validation phase. Once the identification process narrows the search space to one category, verification becomes a more manageable task. Importantly, global features are insufficiently distinctive enough for the purpose of verification.

### 4.3.2 Local Features

Local features differ from global features in that they are not visible to the naked eye. Fingerprint ridges are not continuous straight lines; they may break, fork, change direction, or terminate (Digital Persona, 2004). The point of discontinuity is called a minutia point (the more common plural term being minutiae) (Galton, 1892).

There are five characteristics of a minutia point (Digital Persona, 2004):

1. Type: there are really only two primary types (Yager and Amin, 2004b). However, variations of the two primary types occur, as described below and illustrated in Figure 4.3:

   - Ridge termination is when a ridge ends abruptly. Two variations are:
     - Independent ridge: a short ridge terminating at both ends.
     - Dot or Island: a very small ridge that appears to be a dot.
   - Ridge bifurcation (branching or forking) is when a ridge divides into two or more individual ridges. A variation is:
     - Enclosure: a ridge that divides into two and then reunites to create an enclosed area. The length of the enclosure is typically quite small, with the ridges reuniting shortly after diverging.

2. Position: the location of the minutia point, determined as x, y coordinates in a two dimensional coordinate system. Figure 4.4 provides an example of minutiae whose positions are registered in a coordinate system; grid lines on the x axis are spaced 40 units apart and grid lines on the y axis are spaced 50 units apart. Each minutia point is identified by a red circle.

3. Spatial frequency: the average distance between ridges in the neighbourhood of a minutia.

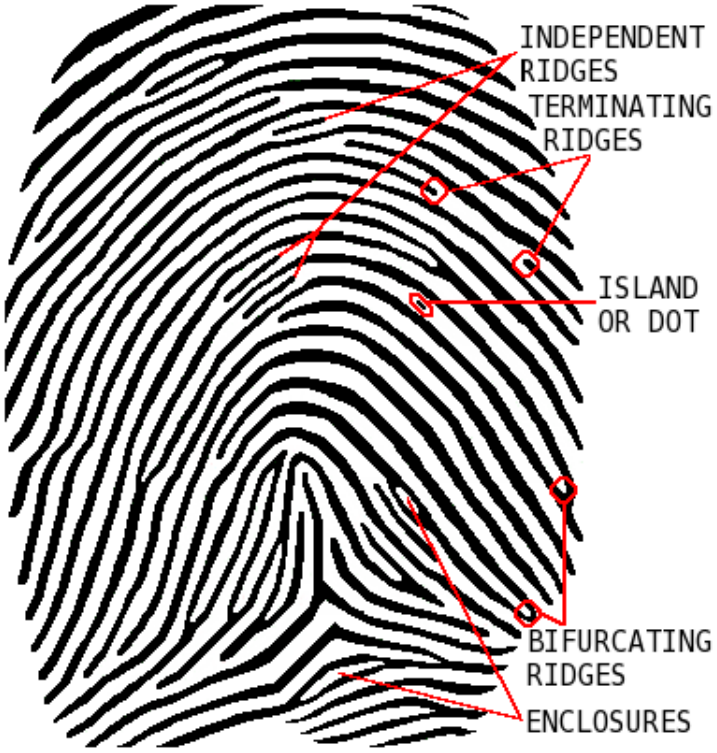


Figure 4.3: Local Fingerprint Features Types

4. Orientation: the angle between the tangent to the ridge at a minutia position and the horizontal axis (i.e. the axis at right angles to the vertical axis of the finger). Note that the "ridge" used for calculating the orientation is determined according to the minutia type—termination or bifurcation (Maltoni et al., 2003):

   - For a terminating minutia, the orientation is determined by the ridge that approaches the point.
   - For a bifurcating minutia, the orientation is determined by the centre line of the furrow that approaches the point.

   In Figure 4.4, the red tail extending from a red circle (which highlights a minutia point), indicates the direction of the tangent which provides the orientation.

5. Curvature: the rate of change of the ridge orientation as the ridge approaches a minutia. As just described in point 4 above, the ridge used for calculation is determined according to the minutia type.

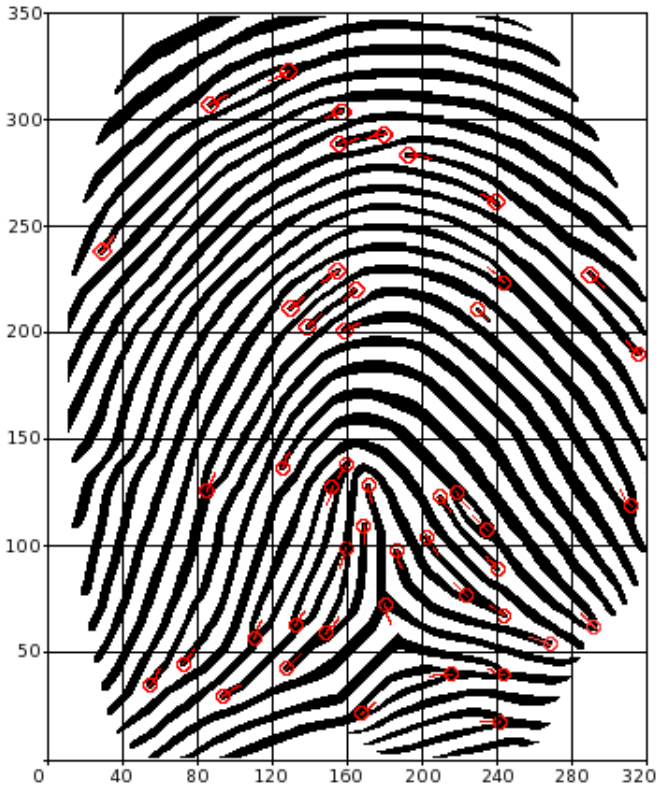


Figure 4.4: Local Features Illustrating Minutiae Positions

In automated systems, considerable difficulties have been encountered when distinguishing between the minutia types listed above (Yager and Amin, 2004b). Usually, to limit uncertainty, only the primary types are differentiated (i.e. ridge termination and ridge bifurcation).

These local features of fingerprint ridges are the unique characteristics used for verification during the validation phase. It is possible for two or more individuals to have almost identical global features but still be differentiated by their local features. Therefore, during the validation phase, global features are better suited to identification, whereas local features are necessary for verification.

Typically, all fingers have a different number of minutiae. Even if two fingers have the same number of minutiae, they will be in different relative positions. This relative positioning of minutiae forms a unique configuration or pattern. It is this pattern and the other minutiae characteristics that are used in the verification process (Maltoni et al., 2003).

Previous studies have shown that successive fingerprint scans of the same finger produce images that almost never match perfectly (Digital Persona, 2004).

That is, two different prints of the same finger will rarely be identical. Reasons for this situation could be sensor inaccuracy (resulting in missing data or introduced artifacts), positional variation due to instrument noise, imperfect imaging conditions, changes in physiological characteristics, ambient conditions, and the elasticity of the epidermal layer of the finger (Maltoni et al., 2003). However, the distinctive feature patterns will still be evident.

## 4.4 Automated Fingerprint Identification Systems

The early fingerprint identification process involved the detailed comparison of latent fingerprints (refer section 4.4.1.2) with images stored in a record system; the comparison was performed manually by fingerprint experts. Because of advances in technology and the huge number of fingerprint images stored in modern databases, automated computer systems are now employed for this task (Maltoni et al., 2003). However, physical (human) confirmation may still be required after computerised systems have narrowed the search, especially if the identification is to be used in some type of legal proceedings.

The first paper to consider automated fingerprint comparison was published by Trauring (1963). His proposal included allowance for affine transformation of minutiae locations, and provided a tolerance level allowing for the variability issues mentioned in section 4.3.2; although his work did attempt to limit positional and orientation variability by the use of a mechanical finger placement guide.

The method was based on the determination of three non-collinear minutiae, designated as 'reference minutiae', and an arbitrary number of other minutiae, designated as 'test minutiae'. Data stored about the reference minutiae were the type, position, orientation, and position relative to their closest neighbours. Data stored about the test minutiae were the type, orientation, and position relative to the three reference minutiae.

When authenticating at a later time, the same data about the respective minutiae were calculated and the results compared to the stored data (with allowance made for variability). Results were calculated utilising simultaneous mathematical equations.

After Trauring, research efforts explored the use of digital fingerprint processing techniques which lead to the development of the first Automated Fingerprint Identification System (AFIS) in 1991 (Kristensen et al., 2007). Since then, a substantial body of work has been undertaken (by law enforcement agencies, private corporations, and academic institutions) to develop and improve accurate and robust algorithms utilised in AFISs (Yager and Amin, 2004b).

The two main application areas for AFIS technologies are in law enforcement and the private sector. The differences between these application areas are:

- The size of the databases maintained. Typically, the criminal databases maintained by law enforcement agencies are much larger than those maintained by small corporations for biometric identification systems.
- The quality of fingerprint images. Law enforcement agencies must deal with inconsistency of quality due to the two types of fingerprints captured; those captured from criminals in custody and latent fingerprints collected from crime scenes. Biometric identification systems typically deal with images of consistent quality, as they are usually captured by the same device under well regulated conditions.

However, the AFISs researched and developed by law enforcement and private sector interests follow the same basic stages (Yager and Amin, 2004b):

- Fingerprint acquisition (discussed in section 4.4.1).
- Fingerprint representation (discussed in section 4.4.2).
- Pre-processing (discussed in section 4.4.3).
- Feature extraction (discussed in section 4.4.4).
- Fingerprint matching (discussed in sections 4.4.5 and 4.4.6).

Obviously, the emphasis placed on each of these stages will depend on their relative importance to the requirements of the particular AFIS being developed. The five stages are discussed in more detail in the following sections.

### 4.4.1 Fingerprint Acquisition

The purpose of fingerprint acquisition is to collect and maintain an accurate record system containing the images (and/or derived data) of all those fingerprinted. Historically, this meant the storage of the original card with the fingerprints imprinted on it or a photographic image of the card (see section 4.4.1.1). As time progressed, hardcopy record systems became very large and cumbersome to maintain and search. More recently, electronic databases have been utilised to take advantage of the fast processing and searching capabilities, and memory capacity, available on modern computer systems.

There are essentially two broad methods for the acquisition of fingerprints:

1. The historical method is called the 'ink technique'. This process is also termed off-line fingerprint acquisition (discussed in section 4.4.1.1). Additionally, law enforcement has evolved a special case of off-line fingerprint acquisition which involves the collection of 'latent' fingerprints from crime scenes (discussed in section 4.4.1.2).

2. The more modern method uses scanning devices to capture fingerprints. This process is termed live-scan fingerprint acquisition (discussed in section 4.4.1.3), and is the method used in biometric systems.

#### 4.4.1.1 Off-Line Fingerprint Acquisition

Off-line fingerprint acquisition is the historical way of obtaining fingerprints. The process involved the application of a thin layer of ink evenly over a subject's finger. This usually meant pressing the subject's finger evenly and firmly against an even flat surface (such as metal plate), that had been previously covered with ink (Galton, 1892). Once the ink was applied to the finger, the finger was pressed evenly and firmly against a paper card. All eight fingers and two thumbs were thus treated.

Historically, a photographic image for each fingerprint card was then obtained. As time progressed, a digital image for off-line fingerprints was obtained by scanning the card, or photographic image, with a digital scanning device.

It should be noted that the collection of fingerprints utilising this method was performed under favourable conditions, and consequently in most cases they were of superior quality (though this was not always the case due to inept or careless workmanship) to latent fingerprints gathered from crime scenes.

#### 4.4.1.2 Latent Fingerprints

The perspiration pores along the papillary ridges secrete a moist substance, consisting of salts, water and oil (Inbau, 1934). The residue from these secretions leaves an impression of the ridges of the finger on any surface that it comes into contact with. These impressions are known as 'latent' fingerprints. In law enforcement, latent fingerprints have become a crucial tool for the identification, and conviction, of criminals.

The use of latent fingerprints requires (Maltoni et al., 2003):

- The collection of fingerprints from criminals who have been apprehended because of some transgression of the law and/or the collection of fingerprints from suspects or a population which may include the perpetrator (with their permission).
- The maintenance of an accurate record system consisting of the true identity of the criminals and their fingerprints.
- The discovery and collection of impressions of fingerprints (i.e. latent fingerprints) left by the perpetrator at a crime scene.
- The comparison of latent fingerprints with those maintained in the record system.

There are numerous methods developed by forensic scientists to detect and collect or 'lift' latent fingerprints from the many different surfaces on which they may occur. To expose latent fingerprints, technicians use fingerprint powder, fuming and other techniques.

Because of the nature or properties of the various surfaces on which latent fingerprints may be detected, it is typical to group them into two basic categories:

1. Porous surfaces: these are normally conducive to the preservation of latent fingerprints because the residue can soak into the surface. Some examples of porous surfaces would be paper, unfinished wood, and cardboard.

2. Non-porous surfaces: these are less conducive to the preservation of latent fingerprints; because the residue may just be lying on the surface, the prints are in a much more fragile situation. Even the slightest handling can disturb a latent fingerprint on such surfaces, thus compromising the accuracy and usefulness of the lifted print. Some examples of non-porous surfaces would be plastic, glass, and metal.

The Latent Print Unit (LPU) of the United States of America's Federal Bureau of Investigation (FBI) suggest the sequential procedures presented in Table 4.1, for detecting and lifting latent fingerprints (Trozzi et al., 2000). Note that the procedures listed are a general approach which may need modification due to particular crime scene circumstances. For example, certain surfaces may require application of specific chemical agents. Also, a latent fingerprint may be embedded in blood or some other substance.

| Step | Porous Surfaces | Non-porous Surfaces |
|---|---|---|
| 1 | Visual | Visual |
| 2 | Fluorescence by laser of alternate light | Fluorescence by laser of alternate light |
| 3 | Iodine Fuming | Cyanoacrylate Fuming |
| 4 | DFO (1,8-Diazafluoren-9-one) | Laser or alternate light source |
| 5 | Laser or alternate light source | Cyanoacrylate Dye |
| 6 | Ninhydrin | Laser or alternate light source |
| 7 | Physical Developer | Vacuum Metal Deposition |
| 8 | | Powder |

Table 4.1: FBI Latent Fingerprint Collection Procedures

Explanation of the general steps listed in Table 4.1:

- Visual (Step 1) - Examine all evidence visually before using any latent fingerprint development technique. The evidence should be well illuminated, and any visible latent fingerprints should be photographed prior to further processing.

- Fluorescence (Step 2) - In a darkened room or enclosure, aim the light source at the object. View the object through an appropriately coloured filter, and photograph the latent fingerprints exposed by the light source. No pre-treatment is required, therefore no alteration of the exposed latent fingerprints occur.

- Utilise latent fingerprint development techniques (Steps 3 onward). As indicated in Table 4.1, this may entail application of various chemicals, dyes, powders, or vapours (fumes). These should be applied sequentially as instructed, and re-examination of exposed latent fingerprints (using a light source) should occur at the appropriate time. To lift a latent fingerprint from an object (after it has been exposed and photographed), apply black, grey, or white powder to the surface with a long hair brush. Use a short hair brush to remove excess powder. Use caution and avoid over brushing as loss of clarity of the latent fingerprint may eventuate. Use transparent tape applied to the exposed latent fingerprint to lift the impression from the object surface onto the tape. Place tape onto a backing card, ensuring that the color of the backing card contrasts with the color of the powder.

Though the methods of detection and collection of latent fingerprints have become efficient, it has been noted that latent fingerprints are sometimes useless for comparison (Inbau, 1934). Their collection is dependent on the quality of the impression, the conditions of the crime scene, and the completeness of the latent fingerprint. That is, collection is seldom performed under pristine conditions, and it is common for latent fingerprints to be of poor quality or incomplete (with only a portion of the full fingerprint being retrievable from a surface).

Therefore, establishing identity from latent fingerprints is often very difficult and sometimes not possible. Identification is made even more difficult when the latent fingerprints of only one or two fingers are able to be collected. In fact, it is unusual to discover (and subsequently collect) a complete set of latent fingerprints belonging to a perpetrator.

#### 4.4.1.3 Live-Scan Fingerprint Acquisition

Live-scan fingerprint acquisition necessarily involves the use of a scanning device. The finger is scanned and data from the scanning process are extracted and stored (usually represented as a two dimensional image). This method of acquisition is more convenient than the off-line method, and it is less expensive and less time consuming (producing digital images in real time).

Fingerprint scanning devices typically comprise the following components (Maltoni et al., 2003):

- A sensor for scanning the finger surface.
- An analogue to digital converter.
- A module for exchanging instructions and data with an external device (such as a computer).

Fingerprint scanning devices utilise sensors that are generally grouped into three common types (Xia and O'Gorman, 2003):

1. Optical sensors utilise a light source and lens to depict the characteristics of the fingerprint. When a finger contacts the platen of the scanner, the light source is triggered. The light that passes through the platen is totally reflected when it strikes the furrows between ridges, but the ridges themselves cause the light to scatter. So the light reflected from furrows demonstrates full light intensity, whereas the light scattered by ridges has significantly reduced intensity. The reflected light is focused by the lens onto either a Charge Coupled Device (CCD) or a Complementary Metal-Oxide Semiconductor (CMOS):
    - A CCD is an analog device that stores a small electrical charge in each of the photo sensors on the chip when light is focused on them by the lens . The charges are converted to voltage one pixel at a time as they are read from the chip. The conversion of voltage to digital data is accomplished by a digital converter that is usually incorporated into the scanning device.

- A CMOS chip is a type of active pixel sensor. Circuitry next to each photo sensor converts the light energy to a voltage. The CMOS chip incorporates an onboard digital converter to convert the voltage to digital data.

2. Solid-state sensors comprise an array of sensing elements that image the fingerprint:

   - Capacitive sensors determine the distance from the sensing surface to fingerprint ridges and valleys by measuring the electrical field strength. The ridges and furrows of the fingerprint can be differentiated by their capacitive measurement because ridges are closer to the sensing elements that the furrows.
   - Temperature sensors differentiate ridges from furrows by temperature difference; ridges touch the sensing surface while furrows do not, thus producing the temperature differential.

   In both cases, the measurement differentials allow for depiction of fingerprint characteristics that can be represented as a grey scale image.

3. Ultrasonic sensors rely on the acoustic qualities of sound waves (Bicz et al., 1999). They provide the ability to obtain an image of the fingerprint based on the acoustical impedance difference between ridges and furrows (Schneider and Wobschall, 1991):

   - Acoustic impedance (or sound impedance) is a ratio of the sound pressure divided by the particle velocity and the surface area, through which an acoustic wave propagates.
   - The difference in the acoustic impedance ratios between ridges and furrows is sufficient to affect the amount of acoustic energy returned to the receiver, thereby allowing a grey scale image to be produced that depicts the characteristics of the fingerprint.

A digital image determined by a fingerprint scanner has three main attributes (Maltoni et al., 2003):

1. Resolution: indicates the number of pixels (picture elements) per square inch (dpi). In general, an image with high resolution (greater than 500 dpi) exhibits clear definition. As resolution decreases the image becomes less defined. This usually results in greater difficulty when attempting to determine ridges and identifying their features, and often leads to inaccuracy during the feature extraction process.

2. Capture area: the dimensions of the rectangular area (height x width) read by the sensing element. The larger the area (that is, the higher the percentage of the full fingerprint area), the more ridges and furrows are captured. With more information available, there is a better chance of determining the uniqueness of the fingerprint.

3. Bit depth: refers to the format for expressing the intensity value of each pixel. Specifically, this is given as the number of bits allocated. For example, an 8-bit grey-scale image allows for 256 different scales of grey to be used in expressing the intensity value of each pixel.

Larger values for the above attributes results in better image quality and accuracy. However, this inevitably requires greater memory capacity for storing the image information.

Fingerprint scanners have the problem of occasional and unpredictable poor image quality (Xia and O'Gorman, 2003). In addition, other factors discussed in section 4.4.3 may impact on the scanning devices ability to accurately depict the true characteristics of a fingerprint.

### 4.4.2 Fingerprint Representation

After acquisition of a fingerprint in digital form, a convenient and applicable storage representation must be adopted. The chosen representation will be dependent on the number of fingerprints to be stored in a database, and the quality of information required during successive processing stages.

For example, law enforcement agencies (such as the FBI) maintain large databases of fingerprints and require efficient storage and processing of fingerprint images when querying the database. This could suggest reducing the amount of data in the digital images. However, the retention of high quality discriminatory data is a high priority for accurate matching of fingerprints. So, at the expense of storage and processing, quality must be maintained.

There are two representation schemes commonly used:

- The image-based method involves storing most of the actual image data, usually in compressed format, because high quality discriminatory information is required for accurate matching. The method entails extraction of features from an image every time a match is required; because of this extra processing efficiency will be affected. Also, because much of the image data is stored, substantial storage capacity is required.

  This representation method is used by law enforcement agencies because accuracy in the identification process is critical.

- The feature extraction method involves the storage of extracted features only. This scheme reduces the amount of storage space required, because only information about the image (i.e. the fingerprint features) is stored. It also means that processing is reduced when a match is required, because the features have already been extracted. This is the usual method employed in biometric systems because of the real-time processing advantages. The disadvantage of this method is that the actual image cannot be reconstructed, if it becomes necessary. For example, if the pre-processing or feature extraction procedures are upgraded or modified, all fingerprint images for personal in the database would need to be re-captured. Of course, this would not be an acceptable situation in the law enforcement arena, but would most likely be tolerated in biometric systems.

### 4.4.3 Pre-processing

Pre-processing refers to the processing of fingerprint images obtained during the acquisition stage—represented in the appropriate format—prior to feature extraction (Yager and Amin, 2004b)[1]. Pre-processing becomes necessary because of the variations that may occur during acquisition. The variations referred to are those between images of the same finger acquired at different times. These can occur for numerous reasons, including different positional finger placement on the scanning device for successive scans, and variability in the image quality resulting from the scanning process.

According to Maltoni et al., (2003) fingerprint images of the same finger, taken at different times, demonstrate a degree of variability and rarely match perfectly because of the following factors:

- displacement: translational deviation along the x and/or y axes, resulting from the differences in positional finger placement on the scanner.
- rotation: angular deviation in relation to the vertical axis, resulting from the differences in finger orientation during placement on the scanner.
- partial overlap: sometimes only part of a fingerprint is captured during the scanning process, because of the differences in finger placement on the device. This can result in different parts of a fingerprint being captured (where there is less than full overlap between prints), or parts of the fingerprint being outside the sensor capture area resulting in missing content.
- non-linear distortion: due to the elasticity of the skin and the pressure applied as the finger comes into contact with the scanner surface, distortion may occur as the sensed 3-D object (the finger) is scanned and rendered into a 2-D representation (because the scanner has a flat surface). This distortion could manifest as a global difference in scale, or localised distortion affecting the accurate determination of local feature positions in relation to each other.

[1]Note that the pre-processing discussed in this section does not involve the adjustment of fingerprint features in relation to their position, orientation or scale. That process (explained in section 4.4.6.2) is part of the feature matching technique for verification. The pre-processing referred to in this section is specifically aimed at image enhancement prior to feature extraction.

- non-uniform contact: dryness, sweat, injury, grease, dirt, and humidity all affect the contact between finger and scanner. As a result, the possibility of missing data and/or introduced artifacts from a poor quality image is ever present, and will vary depending on the extent to which these conditions exist.

- noise: applies to sensor noise as well as residue left on the scanner surface from a previous scan. Variability in sensor noise is a result of electromagnetic interference. Whilst scanners (and extraction software) attempt to compensate for this, they are successful to varying degrees. In relation to residue, the variability will depend on how often (and well) the scanner surface is cleaned between successive scans.

- feature extraction errors: depending on the quality of the fingerprint image, the feature extraction process may introduce non-genuine features or may be unable to detect some genuine features. The other points mentioned above may also affect the accuracy of the extraction process.

Because of the above factors, and because the successive stages of an AFIS (feature extraction and matching) are reliant on quality data being obtained from the image, pre-processing is an important stage. Low quality images may produce erroneous output from the feature extraction stage, which will subsequently affect accuracy at the matching stage. So, the major task of pre-processing is the improvement of the quality of the image, so that the fingerprint features may be more accurately determined.

Some of the pre-processing techniques developed to improve the quality of an image are:

- Local Ridge Orientation Field Estimation (sometimes referred to as directional field estimation). This process refers to a representation that uses a line segment to indicate the average direction of fingerprint ridges in a localised area. The calculation is based on the orientation of ridges around each pixel, and can be further refined according to the direction of other ridges in the immediate vicinity. The size allocated for the local area varies, but is usually greater

than 1 pixel. Computed for the entire image, the effect of the line segments for all local areas provides a representation that is indicative of the class that the fingerprint may be allocated to (refer Figure 4.5).

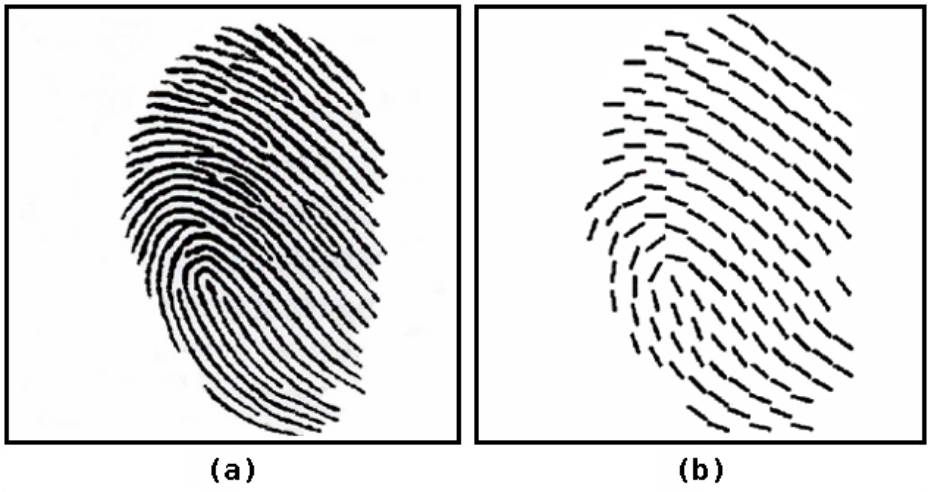


Figure 4.5: Captured Fingerprint (a) And Its Orientation Field (b)
The image was sourced from Yager and Amin (2004b).

- Local Ridge Frequency Estimation. The inverse of the number of ridges (per unit length) along an imaginary line segment, at right angles to the local ridge orientation at coordinate x/y.

- Segmentation. The separation of foreground fingerprint area from the image background. This is useful because it helps avoid extraction of features in noisy areas of the fingerprint and background.

- Enhancement. Contextual filtering techniques aimed at improving poor quality areas in fingerprint images. As fingerprints consist of ridges and furrows with determinate frequency and orientation, known frequency analysis techniques are often used to enhance ridge information. Some of the techniques used are fourier transforms, gabor filters, wavelets, histogram equalisation, and laplacian filtering (Yager and Amin, 2004b).

### 4.4.4 Feature Extraction

Feature extraction involves the collection of information from a grey-scale fingerprint image that can be used by classification and verification techniques. As discussed in

section 4.3, classification and verification have a different purpose based on global and local features respectively. Consequently, the feature extraction processes may differ for classification and verification. However, as both processes revolve primarily around ridge characteristics, some common techniques may be utilised for both.

As the focus of this study is on fingerprint verification, specifically minutiae-based matching techniques (refer section 4.4.6), this discussion will concentrate primarily on feature extraction processes for minutiae-based matching. That is, the feature extraction techniques specifically associated with determining minutiae (and their attributes) from a grey-scale fingerprint image.

There are two main approaches adopted for determining minutiae locations and their attributes:

- Minutiae extraction from a skeletal representation of the fingerprint.

  To achieve identification of genuine minutiae (and extract their attributes), image noise needs to be nullified or at the very least reduced. This approach typically consists of the following four processing steps:

  1. Local Ridge Orientation Field Estimation. A description of this technique was provided in the previous section 4.4.3.
  2. Ridge Detection (often referred to as binarisation). This task is achieved by utilising the intensity values of grey-scale levels in the captured image, with the purpose of returning a black and white image consisting of only binary values. Notably, the maximum intensity value in a grey-scale fingerprint image is achieved along the direction of a ridge; the values gradually decrease to the lowest intensity value, indicating a furrow. As grey-scale representation typically uses 8-bits for storage of the captured pixel values, these pixel values have a possible range of 0 (black) to 255 (white). Those pixels having the highest intensity values should theoretically indicate the line or direction of the top of a ridge; those pixels having the lowest intensity values should theoretically indicate the line or direction of a furrow. For the ridge detection process, thresholding is used to convert the grey-scale representation to a binary representation.

If a grey-scale value is below the threshold, that pixel is allocated the binary value 0; if a grey-scale value is above the threshold, that pixel is allocated the binary value 1—(refer Figure 4.6, image (b)).

3. Ridge Map Thinning. This process reduces the width of each ridge to one pixel. The output from the previous process (binarisation), has the ridges mapped according to the grey-scale captured image, but represented by binary values of 0 and 1. As ridges with values of 0 are very likely to be more than one pixel wide, the ridge representation needs to be 'thinned' to facilitate identifying and extracting the minutia points. This process should ensure that connectivity of ridge bifurcations is maintained, whilst reducing the width of the ridges.

   An example of a 'thinned' or 'skeletal' representation of a binarised image is provided in Figure 4.6, image (c). Very often, post-processing is required to remove spurious artifacts resulting from the binarisation and thinning processes.

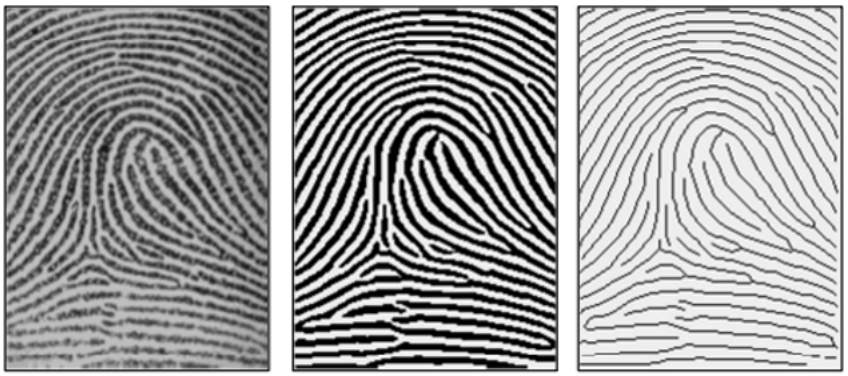


Figure 4.6: Captured Fingerprint (a), Binary (b) and Thinned Representations (c) The image was sourced from Maio and Maltoni (1997).

4. Minutiae Template. This process involves the detection and extraction of minutiae, and storage of their attributes. From the skeletal mapping, the process detects discontinuities in the pixel wide ridge lines. Ridge termination occurs when a skeletal ridge abruptly stops; this occurrence is detectable because the pixels surrounding the last pixel of the ridge line are white on all sides but one; the pixel on the ridge preceding the last

one. Ridge bifurcation occurs at the junction of two skeletal ridge lines; where one ridge line joins another or when one ridge line separates into two. The junction point is detectable because the pixel at that location is not surrounded by white pixels. In fact, there will be black pixels on three sides; the pixel on the ridge preceding the junction, and two other pixels in the directions that the bifurcated ridge lines are heading. Extraneous minutiae will always be present after the detection process, because of noise in the original image and because of spurious artifacts resulting from the binarisation and thinning processes. Smoothing algorithms are used to reduce the erroneous minutiae.

The output from this process is a minutiae template, typically consisting of the x/y coordinates of detected minutiae, and their orientations. Some extraction algorithms return other information, such as minutia type, although a number of matching algorithms do not use this extra information.

- Minutiae extraction from a grey-scale image. In the previous approach, processing the fingerprint image into a skeletal representation occurs before minutiae were identified and extracted. With his approach, ridges (and possibly furrows) are traced from the original grey-scale image. Minutiae are detected and recorded where ridges terminate (or furrows join) or separate (furrows depart). Most techniques use the orientation field estimation to help trace the ridges, and thereafter locate termination and bifurcation points (Maio and Maltoni, 1997; Jiang et al., 2001).

Matching techniques rely heavily on accurate information obtained from the fingerprint image. Thus, accurately identifying and quantifying fingerprint features during the feature extraction process is crucial for the matching process. Although minutiae-based matching is not the only method used for fingerprint matching, it is the most prominent. This and other methods are discussed in section 4.4.6.2.

### 4.4.5 Fingerprint Classification

Fingerprint classification refers to the assignment of a fingerprint's ridge pattern into a defined category, based on accepted pre-determined classes. Classification is necessary because identification requires comparison of a person's fingerprint with those stored in a database (or record system), and this made more efficient by classification-based searching.

Prior to the electronic age, the FBI held photographic images of fingerprints in their record system, but now maintain an electronic database of digital images for over 200 million persons (Yager and Amin, 2004a). This quantity imposes prohibitive time constraints on processing, when searching for a particular fingerprint.

By separating fingerprints into classes, only one subsection (i.e. a single class) needs to be searched. As long as the classes can be consistently differentiated, this is a more efficient method than having to search the entire record.

In 1823, Johannes Evangelista Purkinje (1787-1869) published his thesis entitled *"Commentatio de examine physiologico organi visus et systematis cutanei"*. He proposed nine fingerprint classifications based on the following basic ridge patterns: the traverse curve; the central longitudinal stria; the oblique stripe (left); the oblique stripe (right); the almond; the spiral; the ellipse; the circle; the double whorl (Cummins and Kennedy, 1940).

Galton (1892) proposed only three major classes—the arch; the loop; the whorl—but recognised Purkinje's other classifications as belonging to, or being variations of, his three major classes. Sir Edward Henry (1900) carried on the work of Galton, but extended Galton's three major classes into eight classes (these included an accidental class for those that could not be categorised into the other classes). Variants of Henry's system of classification are used by law enforcement in most countries today.

Table 4.2 shows the correlation between the classes specified by Purkinje, Galton, and Henry.

| Purkinje's Classes (1823) | Galton Classes (1892) | Henry Classes (1900) |
|---|---|---|
| Traverse Curve | Arch | Plain Arch |
| Central Longitudinal Stria | Arch | Tented Arch |
| Oblique Stripe (Left) | Loop | Left Loop |
| Oblique Stripe (Right) | Loop | Right Loop |
| Almond, Spiral, Ellipse, Circle | Whorl | Whorl |
| Double Whorl | Whorl | Whorl (Twin Loop) |
| – | – | Central Pocket |
| – | – | Accidental |

Table 4.2: Correlation of Early Fingerprint Classes by Purkinje, Galton, and Henry

Figure 4.7 provides a pictorial example of six of the classes specified by Henry[2], where core points are indicated by a red encircled white dot, and delta points are indicated by a green outlined white triangle. Figure 4.7 was sourced from an image provided by the Biometric Systems Laboratory (2010)[3].

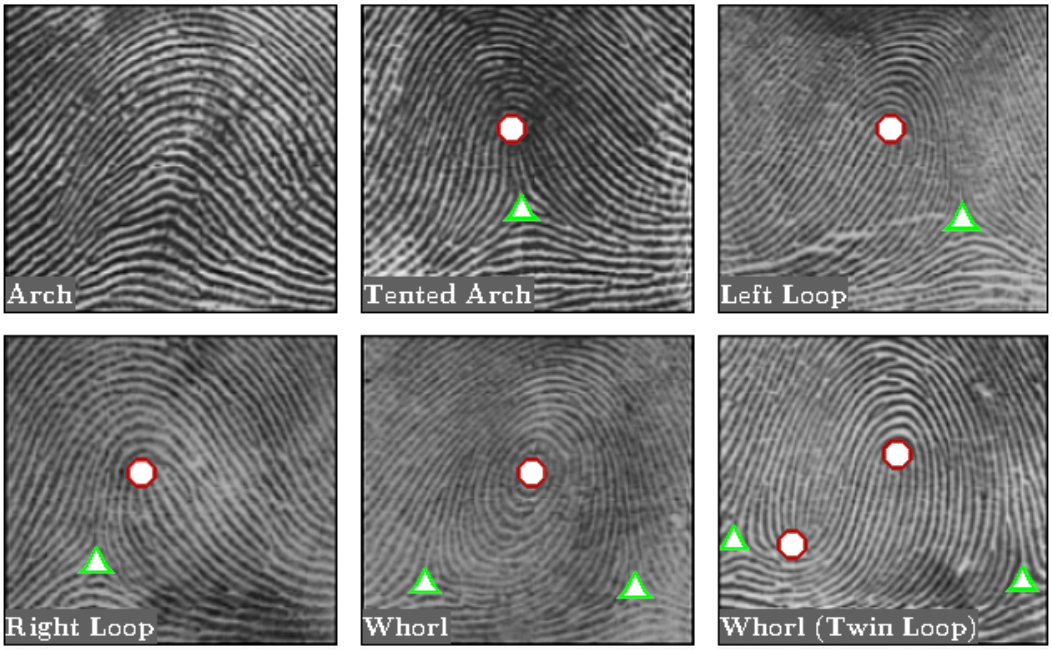


Figure 4.7: Fingerprint Classes defined by Henry

Following is a description of the six classes illustrated in Figure 4.7:

1. The Arch is characterised by ridges entering just above the distal phalanx joint on one side, rising to a small bump as they move toward the middle of the finger, and descending (after the apex) to exit on the other side of the finger

[2]The central pocket and accidental classes occur so rarely that they are usually not considered in most classification schemes (Maltoni et al., 2003).

[3]Available at: http://biolab.csr.unibo.it/ResearchPages/Graphics/FingClass.png

just above the distal phalanx joint. The arch configuration does not include loops and deltas, and therefore does not have a clearly distinguishable core point. A core point for this class can be hypothetically assumed as the point of maximum ridge line curvature (Karu and Jain, 1996).

2. The Tented Arch is similar to the arch, except that the ascending and descending ridges protrude higher up the finger, forming a spire shape with very steep sides. This usually results in a single ridge (the inner most ridge), a core point (at the apex of the inner most ridge), and a delta point (at the base of the inner most ridge where it bifurcates into two ridges).

3. The Left Loop is characterised by ridges entering just above the distal phalanx joint on the left side, circling nearly 180 degrees around the approximate centre, to exit on the left side above where they entered. The left loop pattern includes a core point (central to the loop) and a delta point where ridges converge toward the loop, below and on the right side of the loop.

4. The Right Loop is characterised by ridges entering just above the distal phalanx joint on the right side, circling nearly 180 degrees around the approximate centre, to exit on the right side above where they entered. The right loop pattern includes a core point (central to the loop) and a delta point where ridges converge toward the loop, below and on the left side of the loop.

5. The Whorl is characterised by at least one ridge (but often many) forming a complete circle approximately central to the fingerprint. The whorl pattern includes a core point (central to the whorl). It also contains two delta points where ridges converge toward the whorl, below and on either side of the whorl.

6. The Whorl (Twin Loop) is characterised by two loops. One similar to a left or right loop, and the other similar to its opposite inverted loop so that they fit side by side. The twin loop pattern includes two core points (one central to each loop), and two delta points where the ridges converge toward the loops, below and on either side of the combined loops.

It may be tempting to assume that the fingerprint classes are evenly distributed among the human population. However, this is not the case. According to Wilson et al. (1993), the approximate percentages, shown in Table 4.3, represent the proportion of each class in the human population.

| CLASS | PERCENTAGE |
|---|---|
| Arch | 3.7 |
| Tented Arch | 2.9 |
| Left Loop | 33.8 |
| Right Loop | 31.7 |
| Whorl | 27.9 |

Table 4.3: Proportion of Fingerprint Classes

Note that three of the classes (left loop, right loop, and whorl) make up 93.4% of the fingerprint classifications. Note also, that only five of the Henry classes are represented in Table 4.3. As previously stated, the central pocket and accidentals occur so rarely that they are generally not considered in most classification schemes; twin loops also fall into this category.

Even though partitioning fingerprints into definable classes has the advantage of minimising search space, there are challenges that make the classification task complex and difficult (Yager and Amin, 2004a):

1. As demonstrated in Table 4.3, three of the classes make up 93.4% of the fingerprint classifications in the general populace. This means that though the search space is reduced, it is reduced by only two thirds; which is still a substantial search space in a record system the size of the one that the FBI maintains. This makes one-to-one fingerprint comparison particularly challenging, because differentiation between two individual fingerprints still makes for a very exhaustive search based on these three classes. However, classification before attempting verification does provide some advantage.

2. Often extensive pre-processing is required to improve and enhance the quality of the original fingerprint image, in order to attain a true representation of the ridge patterns; and therefore allow for assignment into the correct class.

3. There is wide variability in the size and shape of patterns within each class.

4. Patterns from one class can sometimes very closely resemble the pattern from another. For example, the arch and tented arch are closely related and a poor fingerprint image could lead to an inconclusive or erroneous classification.

5. It is possible for a fingerprint to have characteristics of more than one class.

#### 4.4.5.1 Feature Extraction For Classification

Feature extraction for classification purposes involves the collection of information from input data that can be used to determine correct class membership.

Classification of fingerprints is based on global features, and so it is appropriate to utilise ridge information for this purpose. The four main methods for representing ridge information are (Yager and Amin, 2004a):

1. Ridge Features. Extracting useful information from fingerprint ridges depends on how ridge data is inspected. One common approach makes use of frequency analysis techniques (such as fourier transform and gabor filters) to extract frequency and orientation components. Another approach is to represent information about the ridge structure. This approach makes use of mathematical modeling methods such as geometric framework and hierarchical kernal fitting, as well as methods such as fudicial lines and ridge recurrence.

2. Local Ridge Orientation Field Estimation. A description of this technique was provided in section 4.4.3. Note that orientation field estimation is often useful for singularity detection (see next point).

3. Singularities occur in locally defined regions where special properties of the ridge pattern are clearly apparent. These were first recognised and described by Sir Edward Henry, and named core and delta points (Henry, 1900). As described in section 4.3.1, a core point is typically the point of maximum ridge line curvature, being the turning point around the inner-most ridge. A delta point (being formed in one of two ways) is the location where two ridge lines diverge in opposite directions from their source (either a bifurcation or two ridges running side by side). The most common method for detecting

singularities is the poincare index, which makes use of the orientation field estimation. By rotating the vectors along a curve in the orientation field, all points can be classified as core, delta, or normal points by inspecting their allocated poincare index (Maltoni et al., 2003).

4. Structural Features represent the correlation between structural elements of the fingerprint. One approach is to dissect the orientation field into regions where vectors (in each region) have similar orientation (Cappelli et al., 1999).

   Each region is represented by a node of a graph (labeled for their region), and adjoining regions are connected by an edge (labeled with difference information between the regions). Another approach attempts to extract shape information about ridges according to 10 defined basic patterns (Chang and Fan, 2002). A combination of these patterns (plain ridge, arch ridge, triangle ridge, left loop ridge, right loop ridge, circle ridge, whorl ridge, smile ridge, balloon ridge, double-loop ridge) are said to occur in all fingerprints.

All methods mentioned above can be used by the appropriate classification techniques as discussed in the next section 4.4.5.2.

#### 4.4.5.2 Classification Techniques

As the techniques used to extract fingerprint features for classification purposes vary according to the methods for representing ridge information, so the techniques used to classify fingerprint features vary according to the information extracted. The following are some of the techniques used on the information extracted, as described in the previous section 4.4.5.1:

- Structural Approach. This approach classifies fingerprints based on the relationship between low-level structural features extracted as described in section 4.4.5.1. In syntactic recognition, a language is developed (composed of a sequence of primitives) that expresses the features of the fingerprint classes. Each class has its own sequence of primitives that describes its characteristics. Matching is performed by comparing a given input with the language

syntax. For example, Chang and Fan (2002) developed their language syntax based the 10 patterns mentioned in section 4.4.5.1. Another approach uses graph mapping algorithms to determine the probability that two graphs have the same intrinsic structure. By modeling the fingerprint classes to a typical structure, a graph matching approach can be facilitated. For example, Cappelli et al., (1999) used the method described in section 4.4.5.1 to extract and describe five of the fingerprint classes (arch, tented arch, left loop, right loop, and whorl).

- Heuristic (or ruled-based) Approach. This approach makes use of human expert knowledge to formulate heuristic rules. They are typically based on the occurrence of singularities, global ridge structures, or both. According to Zhang et al., (2001) the possible combination of core and delta points for the six major classes are as shown in Table 4.4. This facilitates the definition of heuristic rules that can be used for classification. The information extracted about ridge features discussed in section 4.4.5.1 can also be used to define heuristic rules for classification. Also, because the information from either of these approaches can be utilised as a basis for heuristic classification, it is reasonable to suggest that both could be used in combination to formulate more comprehensive heuristic rules.

| Pattern Class | Number of Core Points | Number of Delta Points |
|---|---|---|
| Arch | 0 | 0 |
| Tented Arch | 1 | 1 (Middle) |
| Left Loop | 1 | 1 (Right) |
| Right Loop | 1 | 1 (Left) |
| Whorl | 1 | 2 |
| Whorl (Double Loop) | 2 | 2 |

Table 4.4: Fingerprint Classes and Their Singular Points

- Neural Approach. Since 1990, neural networks have been increasingly used for fingerprint classification, and this is now a commonly used approach (Yager and Amin, 2004a). Various Artificial Neural Network architectures, such as the Multi-Layer Perceptron (MLP); the Self Organised Map (SOM) (or Kohonen network); and the Probability Neural Network (PNN), have been utilised.

Some of the features used in the classification process, by the different research efforts, include singularities, orientation fields (with reduced dimensionality), wavelet coefficients, and the FingerCode technique developed by Jain et al., (1999b).

- Other Approaches. One approach is to use a hybrid classifier, such as a fuzzy neural network. Such a classifier utilises the advantages of neural networks and fuzzy logic techniques. Here, a neural network—through its learning algorithm—generates fuzzy logic rules during the training phase.

  The fuzzy rules incorporate explicit knowledge (in this case learned knowledge by the neural network), and uses high level reasoning to classify the input. Other approaches use support vector machines (SVMs) and hidden markov models (HMMs).

This section has provided an overview of fingerprint classification, and some of the techniques used. An in-depth review of the research efforts for the classification problem has not been provided, as the major focus of this study is the verification problem. For an adequate review of the classification research problem the reader is directed to that presented by Yager and Amin (2004a).

### 4.4.6 Fingerprint Verification

Fingerprint verification involves the comparison of two fingerprint samples. Thus, it is a one-to-one comparison, as opposed to classification which is a one-to-$N$ comparison (where $N$ is the number of fingerprint samples in the record or database). Therefore, classification can be thought of as a one-to-one comparison (or verification) performed $N$ number of times.

Prior to computerised techniques, verification was performed manually by trained fingerprint experts. The most common method was to compare the relative positions of minutiae (which form a unique configuration or pattern) from two fingerprint samples; this is essentially a pattern recognition task.

The skills involving the interpretation and critical analysis of such patterns are developed by human beings from experience. As with many tasks originally performed by human beings, developing acceptably accurate and robust automated computerised techniques is very complicated, and often intractable.

Computer programs can only do what they are programmed to do, and can only deal with the data provided to them; they do not have the inherent ability to gain knowledge from experience or interpret information. It is because fingerprint recognition is such a complex task that the automated systems developed to this point in time are still lacking in accuracy, robustness, and completeness (Yager and Amin, 2004b).

For verification purposes, one sample is the genuine sample obtained during enrollment. This sample is the one upon which comparison is based, and is very often referred to as the 'registered' or 'reference' sample. The other sample is the claimant's sample obtained at the time of attempted authentication, and is often referred to as the 'query' or 'input' or 'test' sample.

The verification process attempts to match a query sample with the registered sample, and indicates the probability that the query sample is a match for the registered sample. That is, the two samples are either confirmed or denied as belonging to the same individual.

In biometric systems, enrollment typically involves the collection of more than one sample to be registered for verification purposes. This is because biometric samples—of the same characteristic collected at different times—rarely match perfectly. So there is a need to collect multiple samples upon which to base comparison. Samples of the same characteristic, from the same individual, collected during enrollment are used to formulate a registered 'template', which is intended to reflect the true components of a 'representative' sample (for that characteristic for that individual).

#### 4.4.6.1 Feature Extraction For Verification

The methods for extracting feature information for verification purposes, were discussed in section 4.4.4. The point was made that minutiae-based matching is the most prominent approach used for verification. Also, a discussion of the two main approaches used to extract minutiae for the purpose of verification—extraction from a skeletal representation of the fingerprint and extraction from a grey-scale image of the fingerprint—was presented. Both approaches typically apply some form of post-processing to filter erroneously detected features. However, further validation of the extracted features is often applied in an effort to confirm correct detection. Heuristic rules and Artificial Neural Networks are sometimes used for this purpose.

The extraction approaches discussed have the following disadvantages (Yager and Amin, 2004b):

- They are computationally expensive because of the binarisation and thinning stages.
- They are unreliable for low quality images because of loss of data due to binarisation and thinning.
- They are vulnerable to the inherent property of non-linear distortion during the capture process.

As a result of these disadvantages, some of the other non-minutiae feature extraction methods attempted are wavelets, gabor filters, image verification, and optical processing (Yager and Amin, 2004b). On their own, these approaches vary in success, but can be, and sometimes are, applied to supplement the other more trusted approaches in order to mitigate the disadvantages listed above.

#### 4.4.6.2 Verification Techniques

As a direct consequence of the fingerprint representation (section 4.4.2), there are two extensive strategies adopted for verification. One strategy is to directly compare the grey-scale images of two fingerprints; using image matching techniques.

The other is to compare previously extracted features from the two fingerprint images; using feature matching techniques.

The many techniques employed for fingerprint verification can be divided into three broad categories (Maltoni et al., 2003):

1. Correlation-based matching is an image matching technique. It can be envisaged as overlaying the registered fingerprint image with the query fingerprint image, to determine the correlation between corresponding pixels under affine transformations (translation and rotation).

2. Minutiae-based matching is a feature matching technique. It is the most widely used technique, and the basis for fingerprint examination performed by fingerprint examiners. Minutiae are extracted from the two fingerprint images and stored as two separate sets of coordinates in a 2-D plane.

   The matching process involves determining the alignment which returns the maximum number of minutiae correspondences or pairings, when the two distributions of coordinate points are compared in various alignments.

3. Ridge feature-based matching is a feature matching technique. It involves the comparison of features extracted from the ridge patterns (not minutiae) of the two fingerprints under examination. This approach is more applicable to situations where the fingerprint images are of low quality, thus making the accurate extraction of minutiae (for minutiae-based matching) and pixels values (for correlation-based matching) problematic.

For the techniques that utilise extracted features for verification, accurate information from the previous stages of an AFIS (acquisition, representation, preprocessing, and minutiae extraction) is vitally important. Also of importance, are the common steps required by these feature matching techniques:

1. Alignment Stage. The fingerprint feature sets under comparison need to be in alignment; or at least in the best possible alignment. This stage involves the determination of transformation factors that best aligns the two feature sets. The transformation factors are then used in the matching stage.

2. Matching Stage:

    - Matching Process. This refers to the actual processing method applied to match the aligned feature sets. Depending on the method applied, additional information (from the feature extraction stage) may be utilised.
    - Matching Score. The process of determining the matching score based on the matching process. This typically involves the accumulation of the number of corresponding minutiae pairs.
    - Verification. The decision analysis method applied to determine verification, based on the matching score. That is, given the matching score, are the two feature sets similar enough to be considered as belonging to the same fingerprint?

A brief overview of some extracted feature techniques is presented below (Yager and Amin, 2004b):

- Minutiae pattern matching. This is another term for minutiae-based matching, and is achieved by locating pairs of corresponding minutiae between the registered and query feature sets. Matching can be visualised as super-imposing the feature set of the query sample over the feature set of the registered sample. When locating correspondences, it is typical to define a small region—called a bounding box—around each minutia in the registered feature set. This allows for correspondences to be made in the presence of non-exact minutia positions (in the query feature set), due to non-linear distortion that is an inherent property of the fingerprint capture process. A matching score is then calculated from the total of minutiae that are coincident in both feature sets (Yager and Amin, 2004b). A normalised matching score calculation was proposed by Jain et al., (1996). The score indicated the likelihood that the two feature sets belonged to the same finger.

  A very similar approach involves comparing neighbourhoods of nearby minutiae (i.e. 3 or more in close proximity) for similarity (O'Gorman, 1998). Each

minutia is a certain distance and orientation in relation to others in its neighbourhood. Each neighbourhood in the query fingerprint is compared with those in the registered fingerprint. If comparison indicates only a minor difference between specific neighourhoods from both samples, then these are said to possibly match. An exhaustive comparison is then performed for all neighourhoods from both samples, and if enough similarities are found, the fingerprints are said to match. The degree to which variance is permitted is determined by a user-defined threshold. This process has much in common with graph matching in mathematics.

- Structural matching. As the name suggests, these methods use structural information to denote relationships between the low-level features of two fingerprints under comparison. These approaches use local structures to perform initial alignment, and then global features to improve that alignment; a matching score is then calculable. One strategy is to use graph matching algorithms on topological representations (nodes and edges) of the ridge structure where a minutia is detected; ridge termination has a different topological representation than ridge bifurcation. Another strategy is to use the topological configuration or pattern associated with minutiae locations. This is typically calculated for each minutia, where the number of other minutiae within a nominated radius are identified and information about each—type, orientation, relative distance—is used to form the topological mapping. The overall topology of the two fingerprints are then compared and a matching score determined.

- Incorporating supplementary fingerprint information. A number of techniques utilise or represent fingerprint information other than minutiae location. These have been developed because of the complexity and computational expense of feature alignment. One approach is to extract the shape and location of ridges associated with extracted minutiae. This information can be used to achieve a course alignment between two feature sets. The locations of minutiae are then used to formulate a 'string' representation which is used for string comparison.

A matching score is then determined based on the results of the comparison. Another approach constructs a rotation and translation invariant key between triplets of minutiae. Features used for the key are distance and ridge count between minutiae pairs, and orientation angles. Still another approach is to use core points to determine the translation factors for course alignment, then use structural features to determine the rotational factor.

- Modeling fingerprint distortion. As it is known that the elasticity of the epidermal layer causes non-linear distortion of captured fingerprint images (at least to some degree), the resultant feature extraction and registration processes are affected. This means that completely accurate alignment of fingerprint features (of the same finger) is highly problematic (if at all possible to achieve); errors can only be minimised. Some matching techniques attempt to overcome this degree of uncertainty by modeling fingerprint distortion. A common technique is to use bounding boxes to compensate for location differences between possible matching minutia. If the modeling is performed after registration, the size of the bounding boxes can be reduced and a higher degree of accuracy achieved.
- Alternate matching techniques. These are techniques that utilise machine learning (such as ANNs or fuzzy neural networks) or statistical pattern recognition approaches. It may also include methods that attempt to use non-minutiae information.

Most research efforts involving feature matching utilise minutiae-based matching techniques. As this is the most prominent approach, the next section 4.5 reviews the research efforts undertaken in this particular field.

## 4.5 Minutiae-based Matching Related Research

Research in the field of fingerprint recognition has been plentiful because of the varied nature of the subject, the applications to which it may be applied, and the complex nature of the tasks involved (Jain et al., 1997). As discussed in section 4.4, there are five stages of an AFIS and each stage has seen increased research

efforts in the last two decades. The aim of these research efforts is to gain further knowledge in the respective stages, and to discover improved solutions to overcome, or compensate for, the complex tasks (some of which are intractable in nature).

This review will focus on fingerprint matching; the last stage of an AFIS. However, because of the breadth of research in this field, and to keep the review more relevant to the current study, it will concentrate on research efforts involving minutiae-based matching[4]. Studies that have used Artificial Neural Networks for minutiae-based matching will be discussed where possible.

The research findings and methodological issues relating to the experiments involving minutiae-based matching, as conducted by the authors of the papers reviewed, are summarised in Table 4.5[5]. As well a being useful as a quick reference for the following discussion, the information in Table 4.5 will be used in Chapter 7 to compare results achieved in the current study with those of previous research.

In early research, verification via minutiae-based matching was achieved by finding pairs of corresponding minutiae (that is, minutiae coincident in both the query sample and the registered sample) and calculating a matching score (Ratha et al., 1996). This involved an alignment process aimed at finding transformation factors that best aligned a query feature set with the registered feature set. Then after applying these transformation factors to the query feature set, a matching process was performed. This process aimed to determine the number of corresponding minutia pairs in both feature sets.

A common task of the alignment process is to determine a minutia common to both query and registered feature sets, upon which to base the alignment of both sets. The corresponding pair of minutia thus determined are generally termed the reference minutiae (one in the query feature set and one in the registered feature set).

[4]It should be noted that the review is by no means a comprehensive coverage of all work done in this area. Rather the research efforts reviewed here were chosen to provide an overview of the techniques developed in this field, and to provide figures with which to compare the results from the current experiment.

[5]Although some authors expressed the performance variables (FAR and FRR) as a percentage, columns 8 and 9 denote the actual rates (i.e. the percentage divided by 100). During the discussion, the performance variables will be presented as both the actual rates and their corresponding percentages (in parenthesis).

| Reviewed Paper | Number of Participants | Samples Per Participant | Feature Attributes | Alignment Method | Matching Method | Analysis Method | FAR | FRR |
|---|---|---|---|---|---|---|---|---|
| Jain et al., 1997 | $18 + 61$ | 10 | 3 | PPM-ARI | SMA | NMS | na | 0.16 |
| Luo et al., 2000 | 100 | 10 | 4 | PPM-ARI | SMA | AMS | na | 0.133 |
| Jiang and Yau, 2000 | 188 | 8 | 4 | PPM-LGS | MCL | NMS | 0.0 | 0.0997 |
| Lee et al., 2002 | 100 | 10 | 4 | PPM-LS | NDM | NMS | 0.0002 | 0.1666 |
| He et al., 2003 | na | na | 4 | PPM-ARI | MR | MMR | 0.0001 | 0.045 |
| Tong et al., 2005 | na | na | 4 | PPM-AFV | SL | NMS | 0.00001 | 0.07 |
| Qi and Wang, 2005 | 100 | 8 | 3 | PPM-FVG | SL | NMSOF | 0.0325 | 0.0605 |
| Jie et al., 2006 | 100 | 11 | 4 | PPM-LGS | MR | MMR | 0.00001 | 0.001 |
| Ozkaya et al., 2006 | 20 | 5 | na | PPM-LGS | SL | MSL | 0.03158 | 0.015 |
| Kumar and Deva Vikram, 2010 | 3,500 | 3 | 2 | na | AFR | ANN | 0.0113 | 0.015 |

| Legend | Alignment Method Description |
|---|---|
| na | Not available – information was not provided by the authors |
| PPM-AFV | Point Pattern Matching using Adjacent Feature Vector |
| PPM-ARI | Point Pattern Matching using Additional Ridge Information |
| PPM-FVG | Point Pattern Matching using Feature Vector based on Global structure |
| PPM-LS | Point Pattern Matching using Local Structures only |
| PPM-LGS | Point Pattern Matching using Local and Global Structures |

| Legend | Matching Method Description | Legend | Analysis Method Description |
|---|---|---|---|
| SMA | String Matching Algorithm | AMS | Accumulated Matching Score |
| MCL | Matching Certainty Level | NMS | Normalised Matching Score |
| NDM | Normalised Difference Measurements | NMSOF | Normalised Matching Score and Orientation Field |
| MR | Matching Result | MMR | Maximum Matching Result |
| SL | Similarity Level | MSL | Maximum Similarity Level |
| AFR | Alternative Feature Representation | ANN | Artificial Neural Networks |

Table 4.5: Summary of Reviewed Literature Involving Minutiae-Based Matching

For this purpose, Jain et al., (1997) proposed—in their seminal paper—the use of planar curve segments for the alignment process. Firstly, each minutia was denoted by three attributes; the x and y coordinates, and the orientation. Also information about the ridge associated with each minutia was obtained from the feature extraction process. This information was a one-dimensional discrete signal normalised by the average inter-ridge distance; essentially providing a normalised ridge length.

A curve segment represented a ridge originating at the coordinates of a minutia; the ridge was further defined by the orientation and associated ridge information. The idea was to determine transformation factors that would align a ridge from the query feature set with a corresponding ridge in the registered feature set.

Applying the transformation factors to all points in the query feature set, would then bring both feature sets into alignment (if indeed the two feature sets were from the same finger).

The set of ridges associated with the query feature set were denoted by $R^d$ and the set of ridges associated with the registered feature set were denoted by $R^D$. For each $d \in R^d$ matched against each $D \in R^D$, a similarity score was calculated according to Equation 4.1:

$$S = \frac{\sum_{i=0}^{L} d_i D_i}{\sqrt{\sum_{i=0}^{L} d_i^2 D_i^2}} \tag{4.1}$$

where $L$ was the number sampling points along the ridge with the smallest magnitude, and $d_i$ and $D_i$ were the distances from sampling point $i$ on the respective ridges $d$ and $D$ to the x axis. The sampling interval for $i$ along the ridges, was the average inter-ridge distance. Given $(0 \leq S \leq 1)$ and a threshold $T_r$, if $S > T_r$, candidate reference ridges (and thus minutia) are identified. In this case, continue processing; otherwise select the next pair of ridges.

For the candidate ridges, translation factors were determined such that the potential reference minutia of the query ridge curve and the registered ridge curve coincided. Then using the orientation and associated ridge information, the rotation factor required to fully align the two ridge curves was determined. Note that the scaling factor was assumed to be 1. Then all points in the query feature set were transformed according to the determined transformation factors.

For the matching process in general, if two feature sets are exactly aligned, then finding the correspondences is simply a matter of counting the coincident pairs. However because of non-linear distortion[6] and other issues associated with the capture process[7], exact alignment of corresponding minutiae rarely (if ever) occurs.

Therefore, it is common practice to utilise a bounding box when determining correspondences. A bounding box is calculated as a small region surrounding each minutiae of the registered feature set, which allows correspondences (between the query and registered feature sets) to be determined in the presence of inexact minutiae locations. Jain et al., (1997) used an 'elastic' bounding box when determining correspondences.

For their matching process, Jain et al., (1997) converted the attributes of each minutiae—in both feature sets—to the polar coordinate system with respect to the respective reference minutia. The transformation factors were calculated for each minutia $(x_i, y_i, \theta_i)$ in relation to the reference minutia $(x^r, y^r, \theta^r)$ according to Equations 4.2, 4.3, and 4.4:

$$r_i = \sqrt{(x_i - x^r)^2 + (y_i - y^r)^2} \tag{4.2}$$

$$e_i = tan^{-1}[(y_i - y^r)/(x_i - x^r)] \tag{4.3}$$

$$\theta_i = \theta_i - \theta^r \tag{4.4}$$

where $r_i$ was the resultant radial distance, $e_i$ was the resultant radial angle, and $\theta_i$ was the resultant orientation with respect to the reference minutia.

The polar attributes for all minutiae were concatenated (in ascending order according to the radial angle) to form a vector of symbolic strings; one symbolic string for each feature set.

[6]Because of the elasticity of the epidermal layer of the finger, non-linear distortion of a feature pattern (in relation to the true feature pattern) is typically caused by differing pressure applied by the applicant during the capture process.

[7]Because of the differing quality of scanning devices and varying environmental conditions, missing or erroneous artifacts are typically introduced.

A dynamic string matching algorithm was then used for the matching process. String matching can be thought of as the maximisation/minimisation of a cost function. The edit distance is a minimising cost function that indicates the cost of equalising two strings. The closer the two strings are to being a match, the lower the edit distance. Jain et al. (1997) incorporated an elastic term (by formulating a bounding box) into the edit distance calculation, to tolerate inexact minutia positions. Thus minutia pair correspondences were determined.

Jain et al. (1997) proposed a normalised matching score, calculated according to Equation 4.5:

$$M = 100\text{x}\frac{N_{pair}}{max\{R, Q\}} \tag{4.5}$$

where $M$ is the normalised matching score, $N_{pair}$ is the number of corresponding minutia pairs, $R$ is the number of minutiae in the registered feature set, $Q$ is the number of minutiae in a query feature set, and $max$ is the function to determine the larger of $R$ and $Q$. Note that values of the matching score range from 100 (for a perfect score) to 1 (for no match at all)[8].

Verification was performed by testing each fingerprint sample against all other samples. There were 2 data sets used in the experiment:

1. Samples collected by an Identix scanner consisted of 180 fingerprint samples; 10 samples of the same finger from each of 18 individuals. This permitted 32,220 (179 x 180) tests.

2. Samples collected by a Digital Biometrics scanner consisted of 610 fingerprint samples; 10 samples of the same finger from each of 61 individuals. This permitted 371,490 (609 x 610) tests.

The decision to accept a query sample as a match to another sample—or to reject it as non-match to another sample—was based on the matching score calculated for the two samples (refer Equation 4.5). Theoretically, a query sample should only

[8]The possibility of at least 1 match exists because the reference minutia from both feature sets must coincide.

match a sample from among the other nine samples provided by the same individual. Otherwise, it should not match another sample.

The matching score of the two samples was compared to a threshold. If the matching score was above a certain threshold, the query sample was said to match the registered sample, and the match was labeled 'correct'. Otherwise it was labeled 'incorrect'.

The performance metrics used were the verification rate and the reject rate. The authors did not define how the verification rate was calculated.

They defined the reject rate as the percentage of fingerprints that tested incorrect, when they were in fact correct (i.e. they did belong to the same individual, but did not match when tested).

Though not clearly stated, if it is assumed that the number of rejected samples was divided by the total number tested for a correct match, then the reject rate is equivalent to the commonly used performance variable the false rejection rate (FRR), except that the reject rate was multiplied by 100 to express it as a percentage. It is assumed that this was the case.

With a threshold value of 25, the authors reported a verification rate of 100% and a reject rate of approximately 0.16 (16%) for both data sets. To ensure that the rates were based on the query sample matching a template, rather than just one other sample, it had to match five or more of the other nine samples provided by the same individual before being accepted as a valid match.

There were some issues with this work that are noted:

- The threshold in relation to the possible perfect matching score, was quite low; that is, 25 out of a possible 100.
- There was no attempt to record false positives to attain a false acceptance rate (FAR)[9].
- For each participants, only 10 samples were available for testing a correct match. A rejection of such a test was recorded by the reject rate. This results in course granularity for the FRR performance variable.

[9]In some literature, the false acceptance rate (FAR) is referred to as the false match rate (FMR), and the false rejection rate (FRR) as the false non-match rate (FNMR).

Luo et al., (2000) introduced modified methods of those proposed by Jain et al., (1997). Firstly, the authors used four minutia attributes—the x and y coordinates, type and the orientation—as well as associated ridge information. Information recorded about the associated ridge was different to that used by Jain et al., (1997). Here, the representation was a sampling of points—at the average inter-ridge distance—along the ridge associated with the minutia.

For the alignment process, a minutia was chosen in both the registered and query feature sets. If the points achieved a certain degree of similarity by comparing their respective ridge curves, they were nominated as the reference minutiae (one from each feature set) for the alignment process.

This comparison of ridge curves was achieved by calculating a distance difference and an angle difference. The distance difference was a ratio of the sum of the distances of imaginary line segments (extending from the minutia associated with the curve and the sampling points mentioned previously) divided by the number of sampling points. The angle difference was a ratio of the sum of the angles, between the imaginary line segments and the orientation of the associated minutia, divided by the number of sampling points.

If the distance or angle differences were greater than a nominated threshold, that particular alignment was discarded. Otherwise, the transformation angle was calculated between the orientation upon which the reference minutia in the registered feature set was located, and the orientation upon which reference minutia in the query feature set was located.

Like Jain et al., (1997), matching was achieved using the polar coordinates of minutiae attributes in both registered and query feature sets, to form symbolic strings. That is, each feature set was represented as a vector where each element consisted of the polar coordinates of each feature in the feature set.

The polar coordinates of the query feature set were applied to the registered feature set, utilising an adjustable bounding boxes (surrounding each minutia in the registered feature set), to determine minutiae correspondences. If comparison indicated that the minutia fell within the adjustable bounding box and the ori-

entation difference between the ridge curves met a certain threshold, a minutia correspondence was registered by incrementing a matching score. This resulted in an accumulated number of corresponding minutiae pairs.

For the experiment, 100 individuals provided 10 sample fingerprints each (of the same finger), giving a total of 1,000 samples. Again, two samples were considered a match if the matching score exceeded a nominated threshold.

A registered template consisted of 10 samples from the same individual. Therefore, there were 100 registered templates; one for each individual. Every sample was tested against its own registered template (the nine other samples from the same individual), and the other 99 templates. For a sample to be considered a match to its registered template, it only needed to match one other sample in that template.

If a sample did match its registered template, then a correct verification (i.e. a true positive) occurred; this incremented the 'correct_num' variable. If a sample did not match its registered template, a rejection (i.e. a false negative) occurred; this incremented the 'reject_num' variable. If a sample did match a different registered template—other than its own—a false verification (i.e. a false positive) occurred; this incremented the 'false_num' variable.

The performance variables used to present the results were the verification rate and the reject rate, as defined by Equations 4.6 and 4.7 respectively.

$$verification\ rate = 100\text{x}\frac{correct\_num}{correct\_num + false\_num} \tag{4.6}$$

$$reject\ rate = 100\text{x}\frac{reject\_num}{1000} \tag{4.7}$$

Note that the verification rate, as defined by Equation 4.6, is referred to as the precision rate (or positive prediction value) in classification analysis literature, and is defined by Equation 6.7 in Chapter 6. Also, the reject rate is equivalent to the commonly used performance variable, the false rejection rate (FRR), expressed as a percentage.

The experiment reported the 'best' results as a verification rate of 100% and a reject rate of 0.133 (13.3%). Unfortunately, the threshold for these figures was not

specified. Other 'good' results were also reported, presumably at different thresholds; again these threshold values were not specified. It is noted that the 'false_num' variable was only used in the calculation of the verification rate. It could have been used to determine the false acceptance rate (FAR)—a commonly used performance variable—but no result was provided for this variable.

The approach taken by Jiang and Yau (2000), used both local and global structures of fingerprints. Local structures rely on relative distances and orientations—that form specific configurations—between minutiae in local areas. With this approach, the best matched local structure alignments provide the correspondences used for aligning the global structure.

Four minutia attributes—the x and y coordinates, orientation, and type—were used to denote each minutia. A feature vector for each minutia was denoted as $F_k = (x_k, y_k, \varphi_k, t_k)$. The representation of a local structure of three minutiae in close vicinity—$m_k$, $m_i$, and $m_j$—consisted of the following:

- the relative distances $d_{ki}$ and $d_{kj}$ between the minutia $m_k$ and the other two minutiae $m_i$ and $m_j$ respectively.
- the orientations $\varphi_{ki}$ and $\varphi_{kj}$, being the differences between the orientation of the ridge upon which $m_k$ is located and the orientation of the ridges upon which $m_i$ and $m_j$ are respectively located. These were defined by a function $d\phi(\varphi_{ki}, \varphi_{kj}) = \varphi_{ki} - \varphi_{kj}$.
- the radial angles $\theta_{ki}$ and $\theta_{kj}$, being the angles between the orientation of the ridge upon which $m_k$ is located and two imaginary line segments joining $m_k$ to $m_i$ and $m_k$ to $m_j$.
- the ridge counts $n_{ki}$ and $n_{kj}$, being the number of ridges between $m_k$ and $m_i$, and $m_k$ and $m_j$.
- the minutia types $t_k$, $t_i$, and $t_j$, being either the ridge termination or bifurcation of $m_k$, $m_i$, and $m_j$ respectively.

This local structure was used to directly match a registered fingerprint with a query fingerprint at the local level. To decide upon a match or non-match, a simi-

larity score was defined—in the continuous domain $[0, 1]$—to describe the matching certainty of local structure pairs. A level of 1 was a perfect match, whilst a level of 0 was a non-match.

The 'best-matched' local structure—from both registered and query fingerprints—served as a reliable correspondence of the two fingerprints. This determined the transformation factors required for the local level alignment, and identified the minutia (within both feature sets) to use as reference minutia.

Global alignment was performed based on the application of the local transformation factors to all minutiae in the query feature set. A 3-D bounding box was then used, when attempting to match corresponding minutia, to allow for deformations in the feature sets—at the global level—that are an inherent property of the fingerprint capture process. The local similarity score was incorporated in the calculation of the bounding box.

A matching certainty level—which also incorporated the local similarity score—was defined to tolerate fingerprint deformations, and determined the number of corresponding minutiae pairs. This matching certainty level was used in the calculation of a matching score, similar to that defined by Jain et al., (1997) (Equation 4.5), except that the matching certainty level was used instead of $N_{pair}$.

For the experiment, 188 participants provided 8 sample fingerprints each of the same finger, giving a total of 1,504 samples. Therefore when testing false acceptance, 1,503 samples were available. However, only 8 samples were available for testing false rejection.

The 'best' FAR presented in the paper was 0.0 with a FRR of 0.0997 (9.97%). This was stated as the findings at a particular threshold, though the value of that threshold was not reported. As it is typical to report the lowest value for the FAR, the above figures have been included in Table 4.5. There were also number of other values presented, at different thresholds, for these performance variables; though again the values of the thresholds were not provided. For example, the FAR of 0.0007 (0.07%), which would generally be considered an acceptable FAR, had an associated FRR of 0.0123 (1.23%).

Lee et al., (2002) also used local structures for alignment, but included the use of normalised distance between minutiae to minimise the effects of non-linear distortion which makes exact point pattern matching difficult.

To denote each minutia, four attributes were used; the x and y coordinates, orientation, and type. Also, the ridge frequency in the local region surrounding each minutia was obtained from the feature extraction process. According to the authors, this ridge frequency incorporates the measurement of local deformations. From the collective ridge frequencies, the average ridge frequency for all minutiae in a fingerprint sample was calculated.

The average ridge frequency was used in the calculation of the normalised distance between each minutia pair and the direction or orientation difference of the imaginary line connecting them.

After identifying reference minutia from both the registered and query samples—upon which to base the local structures for comparison—a nominated radius was used to surround the reference minutiae. Minutiae in this region (of both samples) were converted to the polar coordinate system—with respect to their reference minutia—and used for alignment.

For minutia within this circular region, the difference between the normalised distance of the query feature set polar coordinates and registered feature set polar coordinates was calculated (this calculation incorporated the average ridge frequency). Also, the difference between the radial angles and the directions (orientations) of both sets of polar coordinates were calculated.

Like Jiang and Yau (2000), an adaptive bounding box was used to decide upon minutia correspondences. However, the bounding box defined by Lee et al., (2002) incorporating the normalised difference measurements described above, rather than the similarity score used by Jiang and Yau (2000).

The matching score was calculated in a similar manner to Equation 4.5 proposed by Jain et al., (1997), except that the divisor was given as the number of matched pairs that the two samples have in common.

For the experiment, 100 participants provided 10 sample fingerprints each (of the same finger) giving a total of 1,000 samples. Therefore when testing, 1 sample was matched against the other 999 samples; doing this for all 1,000 samples, meant that 999,000 tests were performed.

Results were presented using the same performance variables as those used by Luo et al., (2000). The minutiae-based matching algorithm, using the normalised average distance proposed by the authors, achieved a verification rate of 100% and a reject rate of 0.121 (12.1%); though no threshold values for these figures were specified. Results were also presented in graphical form (via a ROC graph), from which the FAR 0.0002 (0.02%) and the approximate FRR 0.1666 (16.66%) can be derived; these figures have been reported in Table 4.5.

He et al., (2003) based alignment on a similar method to that proposed by Luo et al., (2000). That is, the use of sampled points from a minutia to points along the associated ridge. The minutia type was included in the attributes used to denote each minutia.

However, He et al., (2003) proposed a slightly modified method for minutiae-based matching. The attributes for all minutiae in both registered and query feature sets were converted to polar coordinates to form symbolic strings. An adjustable bounding box was defined by setting boundaries for the converted reference feature set polar coordinate attributes. The polar coordinate attributes of the query feature set were then applied to adjustable bounding box.

If comparison indicated that a query feature set minutia fell within the bounding box and the orientation difference between the ridge curves met a certain threshold, then a matching result was recorded. After comparing all matching results, the one with the largest magnitude was applied to an empirical threshold. If this matching result met the threshold condition, the fingerprints were accepted as being from the same person; otherwise the match was rejected.

For the experiment, the FVC2000 fingerprint databases—DB1_a, DB2_a, DB3_a, and DB4_a—were used to test the proposed method of minutiae-based matching. Receiver Operating Characteristics (ROC) graphs were provided, which allowed for

the determination of the FAR the FRR for each database; though no precise figures were provided by the authors.

The approximate results provided in Table 4.6 are figures derived from the ROC graphs presented by the authors. The figures reflect the common practice of selecting an appropriately low FAR, whilst maintaining an acceptable FRR. Of these, the results achieved for DB2_a have been recorded in Table 4.5, because they achieved the best figures with a FAR of approximately 0.0001 and a FRR of approximately 0.045.

| Database | FAR | FRR |
|---|---|---|
| DB1_a | 0.0001 | 0.133 |
| DB2_a | 0.0001 | 0.045 |
| DB3_a | 0.0001 | 0.225 |
| DB4_a | 0.0001 | 0.255 |

Table 4.6: Performance Metrics Experiment by He et al., 2003

Tong et al., (2005) proposed the use of adjacent feature vectors for fingerprint feature alignment and minutiae-based matching. The following preliminary information was required for the specification of the adjacent feature vector of a minutia $x$:

- Let $\bar{xy}$ be an imaginary line segment that indicates the orientation ($\varphi$) of the ridge upon which $x$ is located.
- Let four adjacent points $z_1, z_2, z_3, z_4$ be defined as equidistant from $x$, such that $|z_1| = |z_2| = |z_3| = |z_4| = AD$ where $AD$ is a constant.
- Let four angles $\angle yxz_1 = \frac{\pi}{4}$, $\angle yxz_2 = \frac{3\pi}{4}$, $\angle yxz_3 = \frac{5\pi}{4}$, and $\angle yxz_4 = \frac{7\pi}{4}$ define the directions from minutia $x$ to the four adjacent points $z_1, z_2, z_3, z_4$ respectively.
- Let $\theta_1$, $\theta_2$, $\theta_3$, and $\theta_4$ be the orientations (of imaginary line segments parallel to $\bar{xy}$) at the four adjacent points $z_1, z_2, z_3, z_4$ respectively.
- Let $o_1 = \theta_1 - \varphi$, $o_2 = \theta_2 - \varphi$, $o_3 = \theta_3 - \varphi$, and $o_4 = \theta_4 - \varphi$ constitute the four adjacent orientations (with respect to minutia $x$) at the four adjacent points $z_1, z_2, z_3, z_4$ respectively. Note, $o_1, o_2, o_3, o_4$ are said to be rotation and translation invariant.

- Finally, let $n_1 = xz_1$, $n_2 = xz_2$, $n_3 = xz_3$, $n_4 = xz_4$, $n_5 = z_1z_4$, $n_6 = z_2z_3$ denote the number of ridges crossing the specified imaginary line segments. Note, $n_1, n_2, n_3, n_4, n_5, n_6$ are also said to be rotation and translation invariant.

Given the results of the above calculations, an adjacent feature vector for minutia $x$ can be specified by Equation 4.8:

$$Afv(x) = \langle o_{x1}, o_{x1}, o_{x1}, o_{x1}, n_{x1}, n_{x2}, n_{x3}, n_{x4}, n_{x5}, n_{x6} \rangle \tag{4.8}$$

Provided the adjacent feature vector elements are valid for two minutiae $x$ and $y$, the unit distance of corresponding feature vector elements for two adjacent feature vectors can be calculated by Equation 4.9:

$$UnitDistance(x, y|c) = \frac{|x - y|}{c + |x - y|} * valid(x) * valid(y) \tag{4.9}$$

where $c$ is a constant, and valid is a function determining the validity of the feature vector elements.

The normalised distance between two vectors $Afv(x)$ and $Afv(y)$ can be determined by Equation 4.10:

$$AfvDis(x, y) = \left( \sum_{i=1}^{4} UnitDistance(o_{xi}, o_{yi}|m) + s * \sum_{i=1}^{6} UnitDistance(n_{xi}, n_{yi}|k) \right) * 10/validNum(x, y) \tag{4.10}$$

where $m$, $s$, and $k$ are constants; $validNum(x, y)$ is the total number of valid feature elements in both vectors.

Fingerprint matching constituted a three stage process for all minutiae in feature sets $R$ and $Q$:

1. If $AfvDis(r_i, q_i) < T_{afv}$ then add $(r_i, q_i)$ to $S_{pair}$. $T_{afv}$ is a threshold value, and $S_{pair}$ is a list of candidate corresponding matching pairs.

2. By successively selecting and testing two candidate minutiae pairs from $S_{pair}$, determine which two minutiae pairs have the best similarity in terms of relative distance and direction. Transform feature set $Q$ according to the determined relative distance and direction.

3. By successively selecting and testing two minutiae pairs—using Euclidean distance difference and direction difference—a list of candidate matches $M_{pair}$ is assembled. Then by successively selecting two candidate minutiae pairs from $M_{pair}$, and using a similarity level function based on the $AfvDis$ function (refer Equation 4.10), a matching score can be accumulated. This was normalised by dividing the matching score by the larger—in terms of the quantity of minutiae—of the two feature sets.

For the experiment, the FVC2000 fingerprint database—DB1_a, DB2_a, and DB4_a—was used to test the proposed method of minutiae-based matching. Receiver Operating Characteristics (ROC) graphs were provided, which allowed for the determination of the FAR and the FRR for each database; though no precise figures were provided by the authors.

The approximate results provided in Table 4.7 are figures derived from the ROC graphs presented by the authors. The figures reflect the common practice of selecting an appropriately low FAR, whilst maintaining an acceptable FRR. Of these, the results achieved for DB2_a have been recorded in Table 4.5, because they achieved the best figures with a FAR of approximately 0.00001 and a FRR of approximately 0.07.

| Database | FAR | FRR |
|---|---|---|
| DB1_a | 0.00001 | 0.12 |
| DB2_a | 0.00001 | 0.07 |
| DB4_a | 0.00001 | 0.19 |

Table 4.7: Performance Metrics Experiment by Tong et al., 2005

Qi and Wang (2005) proposed a novel approach to alignment using minutiae feature vectors. Three minutia attributes were used; the x and y coordinates and orientation.

Firstly, given a minutia $M_k$, three angles $\theta_1$, $\theta_2$, and $\theta_3$ were defined—with respect to $l_0$, the line passing through $M_k$ and parallel to the $x$ axis—such that $\theta_1 = \psi_k$, $\theta_2 = \psi_k + \frac{2\pi}{3}$, and $\theta_3 = \theta_2 + \frac{2\pi}{3}$, where $\psi_k$ was the orientation of $M_k$.

Lines $l_1$, $l_2$, and $l_3$ were drawn at angles $\theta_1$, $\theta_2$, and $\theta_3$—with respect to line $l_0$ and passing through $M_k$—respectively. $N_{l_m}^k$ equally distributed sample points, at interval $\tau$, were determined along lines $l_m$ for $(1 \leq m \leq 3)$. Therefore, the sample pattern consists of lines $l_1$, $l_2$, and $l_3$, at angles $\theta_1$, $\theta_2$, $\theta_3$, and points $P_{i,l_m}^k$ for $(1 \leq i \leq N_{l_m}^k, 1 \leq m \leq 3)$.

The relative orientation between $M_k$ and each point $P_{i,l_m}^k$ was calculated according to Equation 4.11:

$$\psi_{i,l_m}^k = d\phi(\psi_k, \psi_{i,l_m}^k) \tag{4.11}$$

where $d\phi(\psi_k, \psi_{i,l_m}^k)$ denotes the angle difference between $\psi_k$ and $\psi_{i,l_m}^k$.

Therefore, the feature vector is calculated according to Equation 4.12:

$$F_k = \left[ \{\psi_{i,l_m}^k\}_{i=1}^{N_{l_m}^k} \right]_{m=1}^{3} \tag{4.12}$$

A similarity level was then calculated according to Equation 4.13:

$$S(i,j) = \begin{cases} \frac{T - |F_i - F_j|}{T} & \text{if } |F_i - F_j| < T \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

where $T$ was a pre-defined threshold value. Note, $S(i,j)$ was in the interval $[0,1]$, where $S(i,j) = 1$ denotes a perfect alignment and $S(i,j) = 0$ denotes total non-alignment.

The matching process involved the following steps:

1. Using the above procedures, designate the reference minutia pair upon which to base feature set alignment. This was determined as the best-matched pair that maximises $S(i,j)$—that is, the pair with $S(i,j)$ closest to 1.

2. Perform registration by determining the orientation and translation difference between the registered and query reference minutia.

3. Apply restricted bounding box to determine a list of initial corresponding minutiae pairs. This inevitably identified those pairs with the highest similarity level values.

4. Apply triangular matching to attain true corresponding minutiae pairs. Using the minutiae pairs in the list of corresponding pairs (determined in point 3), let 3 corresponding minutiae pairs be considered vertices of two triangles—one from the registered feature set ($\triangle ABC$) and one from the query feature set ($\triangle A^{'}B^{'}C^{'}$). Determine the length difference $D\phi_l$ and the angular difference $D\phi_o$ between the two triangles.

5. Determine the similarity levels $S_l$ and $S_o$ between triangles $\triangle ABC$ and $\triangle A^{'}B^{'}C^{'}$ according to Equations 4.14 and 4.15.

$$S_l(\triangle ABC, \triangle A^{'}B^{'}C^{'}) = \begin{cases} \frac{T_l - D\phi_l(\triangle ABC, \triangle A^{'}B^{'}C^{'})/3}{T_l} & \text{if } \frac{D\phi_l(\triangle ABC, \triangle A^{'}B^{'}C^{'})}{3} < T_l \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

$$S_o(\triangle ABC, \triangle A^{'}B^{'}C^{'}) = \begin{cases} \frac{T_o - D\phi_o(\triangle ABC, \triangle A^{'}B^{'}C^{'})/3}{T_o} & \text{if } \frac{D\phi_o(\triangle ABC, \triangle A^{'}B^{'}C^{'})}{3} < T_o \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

6. By applying points 4 and 5 to all registered and query features, determine $S_l(\triangle A_1A_2A_3, \triangle B_1B_2B_3)$ and $S_o(\triangle A_1A_2A_3, \triangle B_1B_2B_3)$.

7. If $S_l(\triangle A_1A_2A_3, \triangle B_1B_2B_3) < T_{s1}$ and $S_o(\triangle A_1A_2A_3, \triangle B_1B_2B_3) < T_{s2}$—where $T_{s1}$ and $T_{s2}$ were pre-defined threshold values—the minutiae $A_1$ and $B_1$ are said to minimise the similarity levels and therefore are considered a corresponding minutia pair.

8. Using the method proposed by Jain et al., (1997) for calculation of the orientation field, orientation block pairing was performed. Denoting $(B_1, B_2)$ as the corresponding orientation block pair—$B_1$ from the query fingerprint and

$B_2$ from the registered fingerprint—the similarity level can be calculated using Equation 4.16:

$$S(B_1, B_2) = \begin{cases} \frac{T_1 - D\phi(B_1, B_2)}{T_1} & \text{if } D\phi(B_1, B_2) < T_1 \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

Matching scores were then computed based on the corresponding minutiae pairs (Equation 4.17) and the orientation block pairs (Equation 4.18) and in combination (Equation 4.19):

$$M_m = \frac{\sum_{i,j} S(i,j)}{max\{N_1, N_2\}} \tag{4.17}$$

where $(i, j)$ were successively the corresponding minutiae pairs from the registered and the query feature sets, $S(i, j)$ was computed by Equation 4.13, $N_1$ and $N_2$ were the number of minutiae in the overlapping areas of the registered and the query feature sets.

$$M_o = \frac{\sum_{Bi,Bj} S(B_i, B_j)}{N} \tag{4.18}$$

where $(B_i, B_j)$ were the corresponding orientation block pairs, $N$ was the number of overlapping blocks of both fingerprints, and $S(B_i, B_j)$ was computed by Equation 4.16.

$$M_s = \omega_m M_m + \omega_o M_o \tag{4.19}$$

where $(\omega_m, \omega_o)$ were weight vectors specifying the weights associated with the minutiae matching score $M_m$ and the orientation field matching score $M_o$.

For the experiment, the DB_3 database of FVC2002 was used to obtain 800 samples, where 8 samples were collected from 100 different fingerprints. The results illustrated that the matching score $M_s$ (incorporating both $M_m$ and $M_o$) out performed the matching score $M_m$ (i.e. the corresponding minutiae pairs only). This confirmed the authors assertions that incorporating both matching scores (including the triangular alignment method) complements the matching process.

The authors also compared their methods with other experiments, and demonstrated by the use of ROC graphs that they achieved better results; though no precise figures were provided. For the purpose of comparing results from the current study to those achieved by Qi and Wang (2005), approximate values have been derived from the ROC graph that compared their methods with those of Jiang and Yau (2000). These were a FAR of 0.0325 (3.25%) and a FRR of 0.0605 (6.05%), extracted from Figure 4 in the report by Qi and Wang (2005).

One of the issues with point pattern matching algorithms (incorporating alignment) considered thus far, is the complex nature of determining a corresponding minutia pair (from registered and query feature sets) upon which to base alignment and matching; that is, the selection of a reference minutia. Because of the complexities involved, implementation errors may be introduced (Jie et al., 2006).

In section 4.4.5.2, it was explained that core points are often used in classification techniques because they are relatively simple to detect; if present. Jie et al., (2006) used core points as a basis for their alignment and matching algorithm. They defined a core point as a point of maximum curvature of the concave ridges, and assumed that a core point was central to the registered and query fingerprints. If no actual core point was detected, then the point of maximum curvature was used.

Using the core point as the centre of a circle (radius $r_{reg}$ and $r_q$ for the registered and query samples respectively), only minutiae within these reference areas (circles) were used to search for reference minutia. A restriction was that only minutiae of the same type—ridge termination or bifurcation—were considered for selection as reference minutia. This approach cut down the requirement to search a large number of minutiae for the reference minutiae.

For each attempted pairing between minutiae in the registered and query samples (within their reference areas), a difference angle between their orientations was calculated and stored. Then all minutiae in both samples were converted to the polar coordinate system (with respect to their corresponding reference minutia). After sorting the radial angles of both the registered and query sets, in ascending order, the matching process was applied.

Firstly, a circular bounding region was determined based on the radial distance for each minutia in the registered coordinate set. If a query minutia fell within the bounded region of a registered minutia, and the orientation of the query minutia met a threshold condition in relation to a registered minutia, then a matching score was incremented for that combination. After attempting all combinations of minutiae in the reference areas, the largest of the matching scores was applied to a decision threshold. If the matching score exceeded the threshold, a match was recorded.

For the experiment, 1,100 fingerprint samples were obtained from 100 different fingers; 11 samples of each of the 100 fingers. Each sample was tested against the other 10 from the same finger, for correct acceptance. If a sample was rejected under such a test, a variable 'reject_num' was incremented by one. A sample was also tested against all other 1,089 samples (1,100 - 11) for false acceptance. If a sample was accepted under such a test, a variable 'accept_num' was incremented by one.

Results were presented as a FAR of 0.00001 (0.001%) and a FRR of 0.001 (0.1%). No indication of the values used for the decision threshold or the matching threshold were provided.

According to Leung et al., (1991), Artificial Neural Networks (ANNs) enable solutions to pattern recognition problems where algorithmic methods are too computationally expensive or do not exist at all.

There are four main steps in a pattern recognition system: image registration, pre-processing, feature extraction, and matching (Leung et al., 1991). The early research efforts to use ANNs for fingerprint recognition, did so primarily for either the image enhancement or feature extraction processes.

Ozkaya et al., (2006) used ANNs for image enhancement and the cross-number method for feature extraction. However, the matching process was achieved with the use of local and global structures. A 'matching area' was denoted by a circle of radius $r$, which was adjustable in magnitude. Firstly, the core point of a fingerprint was determined, and $r$ was set to the distance from the core point to the closest edge of the scanned fingerprint area[10].

[10]This was performed for each fingerprint under consideration.

By super-imposing a query feature set over the registered feature set (based on coordinate positions), minutiae were determined such that corresponding pairs were identified AND the pairs were located in the overlapping matching areas of both feature sets. By selecting one of these pairs, a sum of similarity scores was calculated for the current group of corresponding pairs. This was performed for all possible combinations in the current group of corresponding pairs.

The maximum similarity score (from the above process) was compared to a threshold value to decide if both fingerprints were from the same individual. The method used to calculate the similarity scores was not defined by the authors.

For the experiment, 100 fingerprint samples were provided by 20 individuals. Each sample was tested for correct recognition (against the other 4 provided by the same individual) and correct rejection (against all other samples from other individuals). At the threshold value of 0.61 a total accuracy rate of 95.34% produced a FAR of 0.031579 (3.1579%) and a FRR of 0.015 (1.5%).

Kumar and Deva Vikram (2010) used ANNs for matching purposes. Once minutiae were located from the skeletal representation of a fingerprint image[11], their approach was to isolate the detected minutiae in the original pixel image of size 512 x 512 pixels. This image was then reduced to a 15 x 15 matrix (i.e. 225 matrices).

In a matrix where there were minutiae, the intensity values of each pixel (in that matrix) were summed; this sum was then normalised to the interval $[0, 1]$. In a matrix where there were no minutiae the value 0 was allocated.

The matrices were then used as input to the input layer neurons of a multi-layer back-propagation neural network. Being a supervised learning algorithm, error signals are calculated at the end of each feed-forward cycle and then propagated backward through the network by updating the weights connecting the neurons.

For the experiment, 3,500 individuals provided 3 samples of the same finger (i.e. 10,500 samples). There were 6,000 samples used to train the ANN; consisting of two samples from each of 3,000 individuals.

[11]Note: the authors state that alignment of feature sets was unnecessary, because their representation method applied to ANNs accounted for relative orientation. This meant also that only the x and y coordinate attributes were required.

Thus there were 4,500 samples used to test the trained ANN; consisting of one sample each from the same 3,000 individuals whose samples had been used to train the ANN, and three samples each from the 500 individuals whose samples had not been used during training (3,000 + 1,500 = 4,500). Note that only one ANN was trained to recognise the 3,000 different individuals' fingerprints, and only two samples from each of these 3,000 individuals were used to train the ANN to recognise their fingerprint.

The results reported from the experiment were a FAR of 0.0113 (1.13%) and a FRR of 0.015 (1.5%). These figures did not attain the same level of accuracy as some research efforts discussed above. One reason may be that only two samples per individual were used when training the ANN. This seems to have impacted on the ANNs ability to accurately differentiate what should be distinctive patterns.

Whilst the representation of fingerprint local features, proposed by Kumar and Deva Vikram (2010), has some processing advantages—because of the use of ANNs—there are also possible disadvantages. A disconcerting impact of the representation method is that it may not preserve the uniqueness that is provided by the local feature configuration. The reason fingerprint recognition is widely accepted is the belief that each fingerprint has a unique local feature configuration. With the representation proposed by the authors, there is no guarantee that fingerprints from different individuals will have a completely distinct representation. That is, because of the lost data due to the processing required to obtain the representation, the uniqueness may be compromised and it is entirely possible that two fingerprints under comparison may end up with the same or a very similar representation.

## 4.6 Summary Of Minutiae-Based Matching Techniques

In an effort to identify key issues that require consideration when designing an experiment involving feature matching techniques, section 4.6.1 summarises the different approaches adopted for the minutiae-based matching techniques reviewed in section

4.5. Section 4.6.2 describes the approach adopted in the current experiment, and differentiates between this approach and the other approaches. Finally, section 4.6.3 discusses the reasons why this approach was adopted for the current experiment.

### 4.6.1 Approach Adopted By The Reviewed Research Efforts

This section reviews the approaches employed by researchers (who authored the research efforts reviewed in section 4.5) for accomplishing the task of verification using extracted features (more precisely minutiae-based matching techniques). It was explained in section 4.4.6.2 that there were two common steps required by feature matching techniques: feature alignment and the matching process.

The first step in most minutiae-based matching techniques is the alignment process. This process attempts to align the features in the query feature set with those of the registered feature set. Provided the two feature sets have indeed been collected from the same fingerprint, it should be possible to attain an accurate alignment. This task has seen different approaches investigated (a number of which are summarised below), and must be performed prior to attempting the second step (i.e. the matching process).

For the alignment process to be successfully achieved, a minutia common to both query and registered feature sets—upon which to base the alignment of both sets—must be determined. The corresponding pair of minutia thus determined are generally termed the reference minutiae (one in the query feature set and one in the registered feature set).

A number of the research efforts reviewed in section 4.5—for example, (Jain et al., 1997; Luo et al., 2000; He et al., 2003)—utilised a curved line segment in both feature sets as a basis for the alignment process (with both line segments originating at their respective reference minutia). This alignment process was performed for a localised area of the fingerprints under examination. Typically, nominated points along the ridge—originating at the reference minutia—were determined and utilised in the alignment process.

Other research efforts applied different strategies in order to achieve feature alignment prior to matching. Of those reviewed in section 4.5, the following strategies were employed:

- Jiang et al., (2000) utilised the configuration of local and global fingerprint structures to first align two samples; the same structure information was used in the matching process.

- Lee et al., (2002) calculated the normalised ridge distance of a fingerprint and used this in conjunction with the configuration of structures in a localised area (within a prescribed circle of arbitrary radius).

- Tong et al., (2005) based alignment on points along ridges (in the local area) adjacent to the ridge originating at the reference minutia, to construct adjacent feature vectors.

- Qi and Wang (2005) based alignment on a sampling of points along imaginary lines whose orientations were at nominated angles around a reference minutia. The angles around the minutia were specified as $\theta_1, \theta_1 + (2\pi/3), \theta_1 + (4\pi/3)$, where $\theta_1$ was the orientation of the ridge originating at the reference minutia (again a local area alignment).

- As discussed in section 4.4.5.2, global fingerprint features (such as core points) are sometimes used in the alignment process. Research efforts by Jie et al., (2006) and Ozkaya et al., (2006) were two such examples.

The matching process conducted by a number of the reviewed research efforts utilised converted polar coordinates (Jain et al., 1997; Luo et al., 2000; Jiang and Yau, 2000; Lee et al., 2002; He et al., 2003; Jie et al., 2006). According to Jain et al., (1997) polar coordinates minimise the radial distortion properties associated with non-linear deformations (particularly those occurring in local areas). Also, reduced error in the calculation of the rotational factor (during alignment) is more likely if polar coordinates (rather than cartesian coordinates) are used.

A number of the reviewed research efforts utilised bounding boxes during the determination of matching points. A bounding box allows two points to be consid-

ered to match even if their aligned locations are not precisely the same; that is, it provides a small degree of flexibility when attempting to match two points. This flexibility is required to allow for non-linear distortions (an inherent property of the capture process), and to account for any loss of precision during calculation of the transformation factors (when aligning fingerprint feature sets).

For their matching process, Jain et al., (1997) utilised converted polar coordinates of minutiae supplied to a string matching algorithm. A normalised matching score was defined, where its determination was based on the number of corresponding minutiae in both feature sets.

Luo et al., (2000) also utilised converted polar coordinates of minutiae supplied to a string matching algorithm. However, minutiae correspondences between the two feature sets were determined by an accumulated matching score.

Based on numerous sets of different candidate alignment factors, He et al., (2003) and Jie et al., (2006) utilised a matching result (which could be interpreted as a certainty level) to determine the best alignment factor set (i.e. those that resulted in the most number of correspondences between the converted polar coordinates of minutiae). The final match was based on the maximum matching result.

Jiang et al., (2000) utilised a matching certainty level for their matching process, and the normalised matching score proposed by Jain et al., (1997), to determine the final matching result. Lee et al., (2002) and Tong et al., (2005) also used the normalised matching score to determine the final matching result. However, their matching processes were different. Lee et al., (2002) based matching on normalised difference measurements, while Tong et al., (2005) utilised a similarity level.

Qi and Wang (2005) and Ozkaya et al., (2006) based their matching process on a similarity level. However, Qi and Wang (2005) determined the final matching result based on a normalised matching score of minutiae correspondences and a matching score based on the orientation fields. Ozkaya et al., (2006) determined the final matching result using the maximum similarity level.

Kumar and Deva Vikram (2010) based their matching method on an alternative feature representation, which did not require feature alignment. Matching was based on ANN classification.

The next section discusses the approach adopted in the current experiment.

### 4.6.2 Approach Adopted In The Current Experiment

This section discusses the approach adopted for treatment of fingerprint feature data in the current experiment. As with the research efforts reviewed in section 4.5 (involving minutiae-based matching techniques), the new fingerprint feature representation method developed for the current experiment also required fingerprint feature alignment and matching of corresponding minutiae between two feature sets (refer Chapter 5 sections 5.5.4 and 5.5.5). However, the processes involved different treatment of data than the other research efforts[12].

Firstly, fingerprint feature alignment was performed as a separate process in the current experiment (rather than as part of the verification technique itself), and was performed for all participants' fingerprint samples prior to feature selection (refer Chapter 5 section 5.5.5). As reported in Chapter 5 section 5.5.4, a point pattern matching algorithm proposed by Van Wamelin et al., (2004) was utilised because it demonstrated accurate results when tested. Also, the algorithm accommodates rotation, scale and translation variations, and is robust to missing data and/or introduced artifacts.

Feature selection was also performed as a separate process, to meet the requirements in relation to the determination of 8 local fingerprint features (and their attributes)—from each sample—that were to comprise the ANN input vectors (for training and testing). This process is discussed in Chapter 5 section 5.5.5.

Finally, there was a difference between the matching method used in the current experiment compared to those employed in the research efforts reviewed in section 4.5. Numerous research efforts used matching certainty levels or similarity levels to determine the number of corresponding minutiae between two feature sets.

[12]The reasons for the difference in treatment are associated with the objectives of the study, and are discussed in the next section 4.6.3.

This quantity was typically normalised in some manner to demonstrate the final matching score. The current experiment utilised Artificial Neural Network (ANNs) to discern patterns within the fingerprint feature data (determined by the new fingerprint feature representation method)[13]. Final classification was determined by the use of a decision threshold applied to ANN outcomes. One ANN was trained to recognise the pattern of each participant's fingerprint feature data (that is, there was one ANN trained for each participant)[14].

### 4.6.3 Rationale For The Adopted Approach

The reason that the above approach was adopted for the current experiment was influenced by two main factors:

1. It was an intension of the author to examine the use Artificial Neural Networks (ANNs) to perform classification of participants' fingerprint feature data (as well as their keystroke dynamics data, and data combining these two biometric characteristics at the feature level). In Chapter 2 section 2.4.3 it was explained that ANNs (and in particular the Multi-Layer Perceptron with back propagation) is ideally suited to classifying patterns in complex data. To facilitate ease of ANN training and testing, the input vectors to the ANNs were required to be of a standard length. Therefore, a new representation method for fingerprint feature data was required, to produce feature vectors of a standard length (refer Chapter 5 section 5.5.5). Thus this method required a different approach to the minutiae-based matching techniques.

2. As discussed in Chapter 1 section 1.2.2, the primary objective of the study was to fuse keystroke dynamics data and fingerprint feature data at the feature level (refer Chapter 5 section 5.6). The fusion process was facilitated by representing

[13] As explained in Chapter 2 section 2.4.3 the Multi-Layer Perceptron (MLP), as a pattern classifier, is ideally suited to certain pattern recognition tasks where the data is of a complex nature, such as the fingerprint and combined feature data sets involved in the current study. Tried and tested freeware applications of the MLP are available for ease of implementation. For these reasons, the MLP was used in the current study.

[14] To the author's knowledge this approach has not been attempted before, and differs to the approach taken by other researchers who generally train one ANN to classify all participants' samples.

fingerprint features in a standard length feature vector, which required the new fingerprint feature representation method. Again, this is a different approach to the minutiae-based matching techniques.

The details of how the author's new approach was implemented are discussed in Chapter 5. The next section presents a conclusion to this chapter.

## 4.7 Conclusion

This chapter has presented an overview of fingerprint recognition (section 4.2), including a discussion of its historical origins, and the question of the uniqueness of fingerprint characteristics.

A description of the global and local features that characterise fingerprints was then presented, as well a brief insight into their possible uses (refer section 4.3).

The five stages of an Automated Fingerprint Identification System (AFIS) were then outlined (section 4.4). This included a discussion of the acquisition, representation, pre-processing, feature extraction, and fingerprint matching stages. For the fingerprint matching stage, techniques involved in the classification and verification of fingerprints were discussed.

A review of literature involving minutiae-based matching techniques was presented (in section 4.5). This review discussed the different techniques that have been developed in this area of research, and provided summary results that can be used for comparison with the results of the current study.

Finally, section 4.6 summarised minutiae-based matching techniques, and highlighted the different approach adopted in the current experiment; the reasons for adopting this different approach were also explained.

# Chapter 5

# Experimental Methods

## 5.1 Introduction

The purpose of the experiment was to determine and compare the verification accuracy achieved by the following biometric characteristic data:

1. Keystroke dynamics data only.

2. Fingerprint feature data only.

3. The feature level fusion of keystroke dynamics and fingerprint feature data.

Therefore, the experiment was conducted in three phases (in the order listed). Note that the treatment of data (pre-processing) conducted for phases 1 and 2 of the experiment was performed to facilitate the requirements for phase 3 (feature level fusion). This was necessary in light of the objectives and research questions as specified in Chapter 1 sections 1.2.2 and 1.2.3.

The primary objective was to measure any accuracy improvement gained by fusing data at the feature level, as compared to the fusion of data at the decision or confidence score levels (achieved by other authors). Also, to compare feature level data fusion results with those involving individual characteristics, as achieved by other authors and the results of the current study (in phases 1 and 2). Using different treatment for the individual characteristics than that used for the fused data would have defeated this purpose, although it may have produced improved results for phase 1 and/or phase 2 of the experiment.

This chapter describes the experimental process, providing the following details for the three phases just outlined:

1. Overview of the experimental procedures.
2. Participant requirements and recruitment.
3. Software development for collection of data and conducting the experiment.
4. Data collection procedure.
5. Initial analysis and pre-processing of raw data.
6. Final analysis procedures.

## 5.2 Experimental Overview

As the experiment required human participation for the collection of data, it was necessary to gain approval from the Human Research Ethics Committee at Murdoch University. Initially, approval was sought to collect keystroke dynamics data as the fingerprint phase of the experiment was still being researched and collection requirements were not known at that time. Whilst awaiting approval to collect keystroke dynamics data, the software for its collection was developed. Once approval was granted, the recruitment of participants and collection of keystroke dynamics data proceeded (see sections 5.3 and 5.4.2 respectively).

When requirements for the collection of fingerprint data were known, approval was sought to collect fingerprint data. Whilst awaiting approval to collect fingerprint data, procurement of the necessary equipment was undertaken and software for its collection was developed. Once approval was granted, the recruitment of participants and collection of fingerprint data proceeded (see sections 5.3 and 5.5.2 respectively).

After data collection, software to extract feature information from the keystroke dynamics data and the fingerprint data was developed (see sections 5.4.3 and 5.5.3). Pre-processing of extracted data from both sources followed. As discussed in sections 5.4.4 and 5.5.4, the development of the pre-processing strategies and the consequent software were critical tasks, requiring rigorous and exhaustive investigation.

The experiment then proceeded with Artificial Neural Network (ANN) analysis for phases 1 and 2 of the experiment. The procedures are described in sections 5.4.5 and 5.5.6 respectively.

Whilst ANN training was proceeding for phases 1 and 2, the data fusion phase of the experiment was developed (refer section 5.6). For complementary data fusion, formation of data sets was simply a matter of concatenating the fingerprint data onto the keystrokes dynamics data, for each individual sample (see section 5.6.2.1). For cooperative data fusion, a feature selection method was determined (see section 5.6.3.1), followed by the fusion of the selected features from both sources (see section 5.6.3.2). The experiment then proceeded with ANN analysis of the phase 3 data. The procedures are described in sections 5.6.2.2 and 5.6.3.3.

The next section describes the requirements for, and recruitment of, participants for the experiment. The three sections that follow cover the experimental details for each of the three phases in relation to points 3 to 6 listed in section 5.1.

## 5.3 Participants

For the collection of keystroke dynamics data, participants needed to type on a standard computer keyboard on a regular basis. For the collection of fingerprint data, participants had to be in possession of their right index finger.

Participants were recruited from the staff and students of the School of Information Technology at Murdoch University, Western Australia under the terms of approval provided by the Human Research Ethics Committee. Participation was entirely voluntary, and participants could withdraw their participation at any time.

No demographic data about the participants was collected because of the sensitivity of the data (specifically the fingerprint data) and the consequent conditions under which Ethics Committee approval was granted for data collection. However, in general, gender was slightly biased towards males and there was a wide age range, biased slightly towards people in their twenties, and ethnicity was equally divided between Australian and overseas students (predominantly South East Asian). So findings should be reasonably generalisable.

Whilst the preference was to enlist at least one hundred volunteers, only ninety were able to be recruited within the available time frame. Upon recruitment (to facilitate file handling during the experiment) the ninety participants were assigned a participant number between 1 and 90 (i.e. 001, 002, ..., 090). Once a participant was assigned a participant number, no record of their real name was maintained. Thus, there was no way to relate recorded data to a participant's real name[1]. Also, any data files created during the experiment (from recorded data) for any participant, included a participant number rather than a real name.

Not all participants completed both the typing and finger scanning tasks. For reasons associated with gaining Ethics Committee approval for data collection, the collection sessions for each task were conducted in different semesters, and therefore not all participants were available to provide samples for both tasks. Where necessary, extra participants were recruited to ensure that there were equal numbers of data files for both the keystroke dynamics and fingerprint scanning.

The data files attained during the collection phase were of two types; keystroke dynamics data files were given a file name extension of '.txt' (eg: 001.txt) and fingerprint feature data files were given a file name extension of '.ftr' (eg: 001.ftr). The data files were matched according to the participant number given to file names during registration (eg: 001.txt was matched to 001.ftr).

It was considered unnecessary for data from the two types of data files to belong to the same participant, as there is presumed to be no relationship between the particular characteristics of an individual's fingerprints and their typing behaviour. Hence, any participant's typing data file could be paired with any participant's fingerprint data file, provided the link was not changed during the experiment.

It became apparent, after approximately 30 volunteers had provided fingerprint data, that a slight inconvenience was experienced by participants because of the time taken to complete the task (refer section 5.5.2). It was therefore decided to offer a nominal payment of $10 per participant to compensate for any inconvenience experienced.

[1]This was done because of Ethics Committee requirements. Therefore, for the remainder of this dissertation, participants will be referred to by a participant number rather than a real name.

## 5.4 Keystroke Dynamics

### 5.4.1 Software

The software required for this phase of the experiment was developed and implemented according to the requirements listed below:

1. A program to capture and record raw data (refer section 5.4.2).

2. Programs to calculate the metrics from the raw data, and select those used for the experiment (refer sections 5.4.3 and 5.4.4).

3. Programs to create the files for training and testing the ANNs (refer section 5.4.5).

4. Programs to execute training and testing of ANNs (refer section 5.4.5).

5. Programs to calculate results from the ANN testing output files.

Summaries of all programs developed for this phase of the experiment are provided in Appendix C section C.1. Where appropriate, algorithms are provided in the following sections to specify the necessary processing.

The program used to capture the raw data from keystroke events was developed using Borland C++ Builder. This program presented participants with a Graphical User Interface (GUI) to guide their data entry (see Figure 5.1), although all data collection sessions were supervised by the author. Time values associated with keystroke events were captured (and recorded) at a 1 millisecond resolution.

Samples were required to be correctly typed to ensure consistency for all typed samples and integrity of the data. Although only data from correctly typed samples were subsequently used in the experiment, all input samples were recorded.

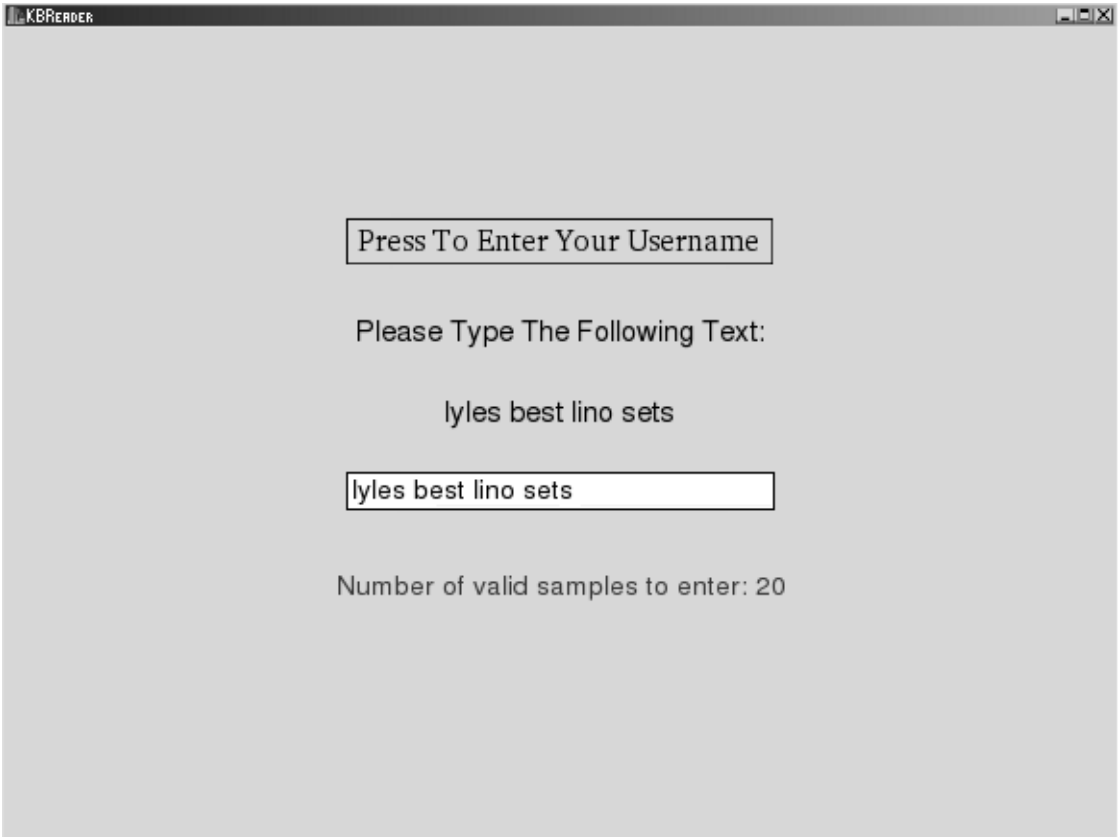


Figure 5.1: Graphical User Interface for Keystroke Dynamics Capture Program

Accordingly, the following restraints were incorporated into the capture program to determine whether entered text strings were valid or not:

- No short cuts were allowed. That is, using copy and paste.
- Correct spelling was required, and case sensitivity was imposed.
- An upper limit of 750 milliseconds, for time intervals between keystroke events, was imposed. Umphress and Williams (1985) determined that time intervals over 750 milliseconds indicated a significant lapse in concentration by the participant.

The actual recorded values were the key press event time and the key release event time for each character typed. The press and release of the 'Enter' key was recorded, so that the metric (digraph latency) could be obtained for the last character typed. Each participants data was written to a uniquely named file, with correctly typed samples preceded by the character 't' and incorrectly typed samples preceded by the character 'f'.

Consideration was given to the impact of system interrupts on the timing of keystroke event capture. However, as data collection was conducted on a standalone computer, if any interrupts occurred, it was considered that the number of samples entered by each participant could reasonably be expected to have nullified any significant effects.

The program developed to calculate the metrics from the raw data (refer section 5.4.3) was implemented as a Perl script. Only samples preceded by the character 't' were used in the calculation of metrics. Once calculated, the metrics were written to a file, uniquely named for each participant.

The "Statistical Package for the Social Sciences" (SPSS) software was utilised to calculate statistics required for the selection of metrics needed for the experiment (refer section 5.4.4). Once the requisite metrics were identified, Perl scripts were used to extract and record them accordingly.

The programs to create the files for training and testing the ANNs were also implemented as Perl scripts, and ensured that no data handling errors were introduced. The task was broken into the following sub-tasks (refer section 5.4.5):

- Randomly assign each participants metrics data file to the training or non-training group.
- Randomly select samples from each of the training group members metrics data files.
- Create ANN training input files, and cross validation files, for each training group member.
- Create ANN testing input files for each training group member.

Training and testing was performed utilising the Matrix Back Propagation Artificial Neural Network. This freeware can be downloaded (with a manual), for Windows and Unix/Linux operating systems (Anguita, 2007). The product was used because it has been in use and thoroughly tested for over fifteen years, and reported to be efficient and accurate (Anguita et al., 1993).

A program was written, in the C programming language, to prepare the ANN outputs for results analysis. Analysis involved the use of Receiver Operating Characteristics (ROC) graphs, to determine a final decision threshold and the subsequent two performance variables (FRR and FAR) for each training group member (refer Chapter 6 sections 6.3 and 6.4).

### 5.4.2 Data Collection

Participants were asked to provide typed samples of a 20 character string on a standard computer keyboard. The decision to use 20 characters was based on earlier research findings (Obaidat and Sadoun, 1997; Abernethy et al., 2004). Though the findings of these research efforts recommended the use of between 10 and 15 character strings, for prudence (to ensure that more than enough data was available) a 20 character string was used during data collection (although as explained later not all metrics calculated from these data were used in the experiment).

The composition of the character string was based on research by Gaines et al. (1980). Their work indicated that the more discernible character combinations are 'io', 'in', 'no', 'on', 'ul', 'il', 'ly', which are all typed with the right hand. The equivalent left hand character combinations are 'ew', 'eb', 'bw', 'wb', 'rs', 'es', 'st'.

In deriving the character string for the experiment, some of these character combinations were employed to form a four word phrase (of 20 characters) that was as sensible as possible. Sensibility was important, so that participants could more easily learn to type the phrase habitually.

The derived phrase was "lyles best lino sets". This constitutes 17 alphabetical characters and 3 space characters, making a total of 20 characters to be entered, and results in 40 key press and release events for each sample (the time values for which were recorded). For reasons explained in the next section 5.4.3, the key press and release time values for the 'Enter' key were also recorded for each sample.

As described in section 5.4.1, the keyboard data collection program checked for erroneous input as well as registering and recording all samples. If a sample was typed correctly, it was preceded by the character 't' when written to file. Otherwise, it was preceded by the character 'f' when written to file.

Samples preceded by the character 'f' were excluded from further processing. Participants were made aware that incorrectly typed samples were rejected by the capture program and had to be re-typed. Upon making a typing error during data entry, nearly all participants merely pressed the 'Enter' key and did not bother completing that sample. So, there was no way to anticipate the number of raw values that were written to file by the capture program for an incorrectly typed sample. It is unknown whether there was any significant impact on data from correctly typed samples, because of participants having to re-type incorrectly entered samples. Analysis of this impact (if any) could be investigated in future, but is beyond the scope of this study.

To ensure that enough samples were collected, each participant was required to provide 160 correctly typed samples (these were preceded by the character 't' when written to their file)[2]. A brief data collection session was held each week (for each participant), over an 8 week period ($8 \times 20 = 160$). Each session required a participant to enter 20 correctly typed samples (with each sample being entered one after the other). All samples from the first week's session (whether correctly typed or not) were treated as familiarisation for participants; therefore, these samples were removed from participants' raw data files. Thus only the samples collected in the last 7 collection sessions were used in the experiment. To ensure the integrity of the data, each data collection session was supervised by the author.

Therefore, each participant's raw data file contained 140 correctly typed samples (and numerous incorrectly typed samples), with correctly typed samples consisting of 43 space separated values in the format listed below:

- a preceding character ('t').
- 40 key press and release time values (one for each keystroke event corresponding to the typed phase).
- the key press and release time values for the keystroke event corresponding to the 'Enter' key.

[2]As indicated earlier, incorrectly typed samples were also written to the same file; these samples were preceded by the character 'f'.

As indicated earlier, files were uniquely named for each participant (using their participant number), and given a '.txt' extension (i.e. 001.txt, 002.txt, ..., 090.txt).

Once data collection was complete, no record of participant names was maintained. That is, there was no way to relate recorded data to a participant's name.

### 5.4.3 Metrics

Metrics were calculated for each sample in the data files 001.txt, 002.txt, ..., 090.txt, provided the sample was preceded by the character 't'. The methodology used to determine the metrics was as described in Chapter 3 section 3.3. For ease of implementation, each sample was stored in an array. As each correctly typed sample of raw data contained 43 values, an array storing the raw data had 43 elements (i.e. index numbers 0 – 42 inclusive). Calculation of metrics (per sample) then became a trivial iterative mathematical process, as demonstrated in Algorithm 5.1.

**Algorithm 5.1** Keystroke Dynamics Metric Extraction

Let $length \leftarrow 40$ be the required number of metrics.
Let $input \leftarrow length + 3$ be an array containing the captured raw data.
Let $metrics \leftarrow length$ be an array for storing the calculated metrics.
Let $index \leftarrow 0$ be a loop control variable to access elements of both arrays.

1: **If** $input[index] ==$'t' **then**
2: **For** $index < length$ **do**
3: $metrics[index] \leftarrow input[index + 2] - input[index + 1]$
4: $index \leftarrow index + 1$
5: **End For**
6: **End If**

As the extraction of metrics only occurred if the first element of the array ($input[0]$) was equivalent to the character 't' (Step 1 in Algorithm 5.1), the character 't' was omitted from calculations as its usefulness was finished (Step 3 in Algorithm 5.1). That is when $index == 0$, $input[index + 2]$ accessed the third element of $input$ and $input[index + 1]$ accessed the second element of $input$ (thus omitting the character 't' stored at the first element $input[0]$). Also note that when $index == 39$, $input[index + 2]$ accessed element 41 of $input$, the keystroke event time value corresponding to pressing the 'Enter' key. This was required to calculate the digraph latency for the last metric.

The result of this procedure was a 40 element array of 20 metric pairs. All 140 correctly typed samples in each data file (001.txt, 002.txt, ..., 090.txt) were thus processed, and the resultant metrics were written to corresponding file names numbered from 001 to 090, with each file name preceded by the character 'm' and followed by the extension '.txt' (m001.txt, m002.txt, ..., m090.txt).

### 5.4.4 Pre-processing

As just described in section 5.4.3, the result of the metrics extraction process was a metrics file for each participant, with each file comprising 140 samples of 20 keystroke duration and digraph latency pairs (i.e. 40 metrics). Though 40 metrics were available, only 20 metrics per sample were selected for the experiment. The decision to select 20 of the 40 possible metrics was based on previous research which demonstrated that a 10 character string was sufficient to attain acceptable accuracy (Obaidat and Sadoun, 1997). As a 10 character string relates to 20 metrics, it was decided to utilise that quantity of metrics for the experiment. Therefore, a selection method was required to determine the best 20 metrics from the 40 available per sample.

As discussed in Chapter 2 section 2.4.3, carefully and intelligently guiding the ANN training process can improve the capability of an ANN to accurately discern patterns in noisy data (Wong et al., 2005). That is, the presence of noise and variability in data can negatively impact on the ANNs recognition capability[3]. One technique that can be used to improve ANN recognition capability is cross validation during training (Wong et al., 2005). This involves selecting a small subset of the available samples[4] for validation during the training process, to reinforce correct learning by the ANN.

In addition, prior statistical analysis of the data can assist cross validation to further enhance the training process in its pattern recognition task (Wong et al., 2005). This involves the identification of metrics responsible for data noise. By selecting metrics less affected by noise, the ANN should then be better able to discern patterns in the data.

[3]It should be noted however that without some variability and noise, an ANN can over train.

[4]In the context of the current experiment, this would mean selecting a subset of the 140 correctly typed samples, provided they were not designated training samples.

Many classical statistical tests depend on normality assumptions (The Northwestern University Medical School, 2007b). An important aspect with relation to a normal (Gaussian) distribution, is the frequency of extreme values in the distribution (i.e. values at the tails of the normal distribution curve). The higher the variable frequencies at the tails, the more noise is evident in the data; this could negatively impact on the ability of an ANN to accurately discern patterns.

For the purpose of identifying variables responsible for data noise, statistical measures that can estimate a distribution's coincidence with a normal distribution (or deviation from it) are:

- Normality coefficient. The normality coefficient is a measure of how well the score frequencies for a variable fit the normal distribution curve. The two most commonly accepted tests of normality (for samples sizes under 2000) are the Kolmogorov-Smirnov test and the Shapiro-Wilk test (The Northwestern University Medical School, 2007b). The Shapiro-Wilk test for normality is specifically designed to detect departures from normality, without requiring the mean or variance of the data distribution to be specified in advance (the Kolmogorov-Smirnov test requires a priori knowledge of the mean and variance). For this reason the Shapiro-Wilk test for normality was chosen for this study.

  A Shapiro-Wilk normality coefficient equal to (or very close to) 1 indicates a normal distribution, whereas a coefficient equal to (or very close to) 0 attests to an extremely non-normal distribution. Therefore, for the purpose of selecting the 'best' 20 metrics, by eliminating noisy data, variables that exhibited normality coefficients closest to 1 were preferable.

  It should be noted that the normality coefficient will not indicate causes of non-normality, so examination of the kurtosis and skewness coefficients for data may provide clues as to why data exhibits non-normality (The Northwestern University Medical School, 2007b). Examining these other statistics may also be helpful in deciding between variables that are vying for selection, but have returned normality coefficients very close to each other.

- Kurtosis coefficient. The kurtosis coefficient is the most common measure of how much a distribution differs from the normal distribution (for symmetrical distributions) (Doric et al., 2007). Two issues that influence the kurtosis coefficient are the flatness or peakedness of the distribution curve, and the fatness or thinness of the tails of the distribution curve.

  Pearson's kurtosis coefficient was introduced as a measure of how flat a symmetric distribution is when compared to a normal distribution of the same variance (Pearson, 1905). Pearson referred to more flat-topped distributions as 'platykurtic'; they exhibit a lower, wider peak around the mean, suggesting that there is a lower probability (compared to a normal distribution) of values near the mean. 'Leptokurtic' distributions exhibit an acute peak around the mean, suggesting that there is a higher probability (compared to a normal distribution) of values near the mean. 'Mesokurtic' distributions resemble a normal distribution, suggesting that there is an equal probability (compared to a normal distribution) of values near the mean.

  Pearson's kurtosis coefficient evaluated a normal distribution to a coefficient of 3, which was adjusted by R. A. Fisher (1930) to evaluate to 0 (by subtracting 3 from Pearson's coefficient). As the coefficient deviates from 0 (with increasing negative values), the distribution is considered platykurtic (flat-topped). As the coefficient deviates from 0 (with increasing positive values), the distribution is considered leptokurtic (peaked); if the coefficient is equal to (or very close to) 0 the distribution is considered mesokurtic (relatively normal).

  So initially, the kurtosis coefficient was considered a measure of whether the data was peaked or flat relative to a normal distribution. However, some authors now accept that kurtosis coefficient measures the tails of a distribution and its shape near the mean value (Doric et al., 2007). De Carlo (1997) proposed that the kurtosis coefficient is primarily influenced by the fatness or thinness of the tails of the distribution, and secondarily by the flatness or peakedness of the distribution. Extremely non-normal distributions may have high positive or low negative kurtosis coefficients, while nearly normal

distributions will have kurtosis coefficients very close to 0 (De Carlo, 1997). The kurtosis coefficient is positive if the tails are 'heavier' than for a normal distribution (i.e. have fat and/or long tails) and negative if the tails are 'lighter' than for a normal distribution (i.e. have thin and/or short tails).

For example, a kurtosis coefficient of 0.062 could be considered to indicate a relatively normal distribution because it is close to zero. It indicates that there are unlikely to be significant variable frequencies in the tails of the distribution curve, and that the distribution is mesokurtic (i.e. normally high). As the kurtosis coefficient departs further from zero, a positive value indicates the increasing possibility of longer/fatter tails and a leptokurtic (i.e. peaked) distribution; a negative value indicates the increasing possibility of shorter/thinner tails and a platykurtic (i.e. flat-topped) distribution.

If the kurtosis coefficient indicates higher variable frequencies around the mean, then there may not be any harmful effects on ANN training. However, if the kurtosis coefficient indicates higher variable frequencies around the tails, then noisy data is evident. This knowledge could be useful when choosing between variables with similar normality coefficients.

- Skewness coefficient. The skewness coefficient is a measure of the degree and direction of asymmetry of the distribution of data (Wuenschk, 2007). A distribution is symmetric if it looks the same to the left and right of the centre point (i.e. arithmetic mean). A symmetric distribution has a skewness coefficient equal to (or very close to) 0. A distribution that has a negative coefficient is skewed to the left, and typically occurs when the mean is less than the median, and the tail is skewed left (i.e. the bulk of distribution is on the right). A distribution that has a positive coefficient is skewed to the right, and typically occurs when the mean is greater than the median, and the tail is skewed right (i.e. the bulk of distribution is on the left).

  A skewness coefficient close to 0 is favourable as this indicates a balanced distribution, as is demonstrated by a perfectly normal distribution (which has a skewness coefficient of 0).

For the purpose of determining noisy data, a skewness coefficient significantly removed from 0 indicates a non-normal distribution. Again, this knowledge (along with the kurtosis coefficient) could be useful when choosing between variables with similar normality coefficients.

- Standard deviation. The standard deviation is a measure of dispersion. It provides an indication of the spread of the data values in a distribution. That is, how much each value deviates from the mean (The Northwestern University Medical School, 2007a). Standard deviation is calculated as the square root of the variance, and therefore has the same linear units as the original data values (instead of the squared units of the variance).

  Like the mean, the standard deviation makes use of all the available sample data, and can be heavily influenced by any extreme values, or by skewed data. Because the standard deviation is based on squared deviations (the variance), a single aberrant value can make a huge difference in the calculated statistic (The Northwestern University Medical School, 2007a).

  Thus, a standard deviation closer to 0 would indicate less data variability (in relation to the tails of a normal distribution curve), but as the variance increases there is more likely to be increasing instances of variability. Again, this knowledge (along with the kurtosis coefficient and the skewness coefficient) could be useful when choosing between variables with similar normality coefficients.

A distribution with a normality coefficient (using the Shapiro-Wilk test) equal to (or very close to) 1 should inevitably have kurtosis and skewness coefficients close to zero. Significant kurtosis and skewness coefficients clearly indicate that data are non-normally distributed, and in particular are likely to contain noisy data.

#### 5.4.4.1 Keystroke Dynamics Feature Selection

In section 5.4.3, the calculation of metrics was explained, which resulted in a metrics file for each of the 90 participants (m001.txt, m002.txt, ..., m090.txt); each file contained 140 samples, with 40 metrics per sample. Each metric in a file was a

time value for a particular keystroke event, corresponding to a specifically typed character. The metric for a specifically typed character was in the same position in all 140 samples. That is, the keystroke duration for the first character 'l' was in the $1^{st}$ position in all 140 samples; the digraph latency for the first character 'l' was in the $2^{nd}$ position in all 140 samples; the keystroke duration for the second character 'y' was in the $3^{rd}$ position in all 140 samples; and so on, with the digraph latency for the last character 's' in the $40^{th}$ position in all 140 samples. This was true for all participants' metrics files.

The statistical analysis of data in these files utilised the SPSS software package. When loaded into the statistical package, the samples in the metrics files are denoted as records and the metrics are denoted as variables. This can be thought of as a matrix, where the samples/records are the rows of the matrix and the metrics/variables are the columns of the matrix. Thus, statistical analysis was performed on the metrics/variables (or columns of the matrix) across all 140 samples/records (rows).

For the feature selection process, the four statistics (discussed in the previous section) were employed to select the most representative 20 metrics/variables. Thus, the four statistics were determined for each of the 40 metrics/variables (or columns), across all 140 samples/records (or rows); this was performed for each participant.

The importance (to the selection criteria) of each of the statistics is reflected in the following priority listing:

- The normality coefficients were rated highest priority.
- The kurtosis coefficients were rated the next highest priority.
- The skewness coefficients were rated equal lowest priority with the standard deviation.

Table 5.1 demonstrates the priority rating scheme devised to facilitate the selection process (in relation to the importance of the four statistics just listed), by allocating weights to them[5].

[5]Note that for each statistic, the decrement was calculated by dividing 40 (i.e. the number of metrics/variables) into the Highest Placed rating.

| | Highest Placed | 2nd Highest Placed | Decrement |
|---|---|---|---|
| Normality coefficient | 1.0 | 0.975 | 0.025 |
| Kurtosis coefficient | 0.5 | 0.4875 | 0.0125 |
| Skewness coefficient | 0.2 | 0.195 | 0.005 |
| Standard deviation | 0.2 | 0.195 | 0.005 |

Table 5.1: Priority Ratings

The priority rating scheme illustrated in Table 5.1 was implemented as follows (for each participant):

1. The normality coefficient using the Shapiro-Wilks test was given highest priority. Simply, the chosen metrics should be as normally distributed as possible, as a normal distribution (with low noise in the tails) suggests a more discernible pattern, and should assist ANN training (Wong et al., 2005). Metrics/variables with coefficients closer to 1 were preferable.

   Therefore, the normality coefficients for all 40 metrics/variables were placed in descending order such that the value closest to 1 was placed highest and the value closest to 0 was placed lowest. The metric/variable with the highest coefficient was assigned the rating value 1. The metric/variable with the second highest coefficient was assigned the rating value 0.975, the metric/variable with the next highest coefficient was assigned 0.95, and so on. The metric/variable with the lowest coefficient was assigned the rating value 0.025.

2. The kurtosis coefficient was given next highest priority, because of its ability to estimate the degree to which a distribution differs from the normal distribution. The greater the number of high frequency scores near the tails, the more data variability was evident. This would affect the ability of the ANN's to accurately determine patterns during the training process. Metrics/variables with kurtosis coefficients closer to 0 were preferable, as this suggested less data variability. Because kurtosis coefficients may have positive or negative values, and because proximity to 0 is of primary importance for ranking purposes, their absolute values were utilised.

   Therefore, the absolute values of the kurtosis coefficients for all 40 metrics/variables were placed in ascending order such that the value closest to 0 was

placed highest and the value furthermost from 0 was placed lowest. The metric/variable placed highest was assigned the rating value 0.5. The metric/variable placed second highest was assigned the rating value 0.4875, the metric/variable placed next highest was assigned 0.475, and so on. The metric/variable with the lowest coefficient was assigned the rating value 0.0125.

3. The skewness coefficient was given equal lowest priority with the standard deviation statistic. A high skewness coefficient generally negatively impacts on the normality coefficient. Whilst the normality coefficient was given highest priority, the skewness coefficient was useful when deciding between metrics/variables where the normality and kurtosis coefficient coefficients were very close. Metrics/variables with skewness coefficients closer to 0 were preferable. Again, their absolute values were utilised.

   Therefore, the absolute values if the skewness coefficients for all 40 metrics/variables were placed in ascending order such that the value closest to 0 was placed highest and the value furthermost from 0 was placed lowest. The metric/variable placed highest was assigned the rating value 0.2. The metric/variable placed second highest was assigned the rating value 0.195, the metric/variable placed next highest was assigned 0.19, and so on. The metric/variable with the lowest coefficient was assigned the rating value 0.005.

4. The standard deviation was also given equal lowest priority with the skewness coefficient. Like the skewness coefficient, standard deviation was used to help decide between metrics/variables where the normality and kurtosis coefficient coefficients were very close. Consequently, it was given equal priority to skewness.

   Therefore, the standard deviation for all 40 metrics/variables, were placed in ascending order such that the value closest to 0 was placed highest and the value furthermost from 0 was placed lowest.

   The metric/variable placed highest was assigned the rating value 0.2. The metric/variable placed second highest was assigned the rating value 0.195,

the metric/variable placed next highest was assigned 0.19, and so on. The metric/variable with the lowest coefficient was assigned the rating value 0.005.

Note that in general, the standard deviations in the data distributions (for this experiment) were relatively high because the known high variability associated with keystroke dynamics raw data (refer Chapter 3 section 3.5).

Once the four individual statistics were ordered for all 40 metrics/variables and the assignment of weights was completed (according to the above description), a tally of the four assigned weights (corresponding to the individual statistics) was calculated for each metric/variable. That is, the weights assigned to a metric/variable for their ranked normality, kurtosis, and skewness coefficients and standard deviation were summed; this was done for all 40 metrics/variables (which resulted in 40 tallies). The forty tallies were then ranked in descending order, and the top 20 metrics/variables (based on these rankings) were selected to be used in the experiment. This process was conducted for each participant.

Recall that this statistical analysis was performed on each of the 40 metrics/variables across all 140 samples/records for each participant. Also recall, that each of the 40 metrics/variables had the same column position in all samples/records. So once selected, a metric/variable (by its column number) indicated the index number used to access the metric value (from within the array used to store the actual metrics data), for all 140 samples. That is, the same metric was selected from each sample. This was true for the 20 metrics/variables selected. The same process was applied to all participants' metrics data files.

As can be deduced from the priority rating scheme, points 2 to 4 (discussed above) only really affected the choice of the last 1–5 metrics/variables, where the normality coefficients may have been equivalent or very close together.

Appendix B provides an worked example of the metrics selection process. The example is presented in four stages, with the relevant data presented in Tables B.1, B.2, B.3, and B.4.

Of particular note is that the selected metrics for each participant were different. To illustrate, Tables 5.2 and 5.3 below provide an example. Firstly, recall that the actual metrics (time values) were stored as elements of an array, and that array index numbers (for this example) increment from 0 to 39. The array index numbers presented in Table 5.2 show the column positions of the selected metrics for participant one[6]:

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 9 | 11 | 13 | 14 | 16 | 19 | 23 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 36 | 37 | 38 | 39 |

Table 5.2: Indices Of Selected Metrics For Participant One

The array index numbers presented in Table 5.3 show the column positions of the selected metrics for participant three:

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 17 | 19 | 21 | 22 | 27 | 29 | 30 | 32 | 33 | 34 | 35 | 37 |

Table 5.3: Indices Of Selected Metrics For Participant Three

It is quite clear that the selected metrics are very different (as indicated by the array index numbers), and this was typical for all participants.

In an effort to add another dimension to each participants' pattern of typing (for data analysis), it was decided to include some global statistics about each sample (along with the selected metrics), to provide a descriptive summary of the sample.

The mean and standard deviation of the selected keystroke duration metrics and the mean and standard deviation of the selected digraph latency metrics were calculated[7]. As the quantity of each metric type (keystroke duration and digraph latency) varied for different participants, it was considered that this approach would possibly provide further differentiation of participant data. Note that even though the selected metrics/variables column positions indicated the index numbers to access (to obtain the actual metrics), the actual metric values at each of these index numbers were different from sample to sample (for each participant).

[6]These index numbers correspond to the selected metrics demonstrated in the worked example provided in Appendix B. Note that values in Table 5.2 are index numbers (starting from 0) and consequently are one less than the selected metrics demonstrated in Appendix B Table B.4.

[7]The ordinal property of array index numbers (which increment by 1 from 0) means that even numbered index numbers indicate keystroke duration metrics and odd numbered index numbers indicate digraph latency metrics. This facilitated easy access of the correct metrics and calculation of the summary statistics.

The 4 global statistics and the 20 selected metrics from each sample (for all 140 samples per participant) were used as input data for the remaining stages of the experiment. As an example, Table 5.4 shows one sample taken from the input file for participant 1, given the selected index numbers presented in Table 5.2. Note that each participants input file consisted of 140 samples, with the sample illustrated in Table 5.4 being one such example.

Once selection was complete, the metrics were normalised according to the min/max method (Indovina et al., 2003). Normalised metrics were then written to uniquely named files for each participant (according to their participant number). Files names were preceded by the character 'm', with a '.txt' extension (i.e. m001.txt, m002.txt, ..., m090.txt).

| | KDM | KDSD | DLM | DLSD |
|---|---|---|---|---|
| GLOBAL METRICS | 146.37 | 59.85 | 75.17 | 62.42 |

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DL | DL | DL | DL | KD | KD | DL | DL | DL | KD | DL | KD | DL | KD | DL | KD | KD | DL | KD | DL | KD | DL |
| SELECTED INDEX NUMBERS | 5 | 9 | 11 | 13 | 14 | 16 | 19 | 23 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 36 | 37 | 38 | 39 | | |
| CORRESPONDING METRIC VALUES | 40 | 40 | 151 | 150 | 80 | 150 | 110 | 110 | 40 | 110 | 110 | 80 | 70 | 191 | -80 | 260 | 150 | 80 | 150 | 81 | | |
| LEGEND | DESCRIPTION | | | | | | | | | | | | | | | | | | | | | |
| KDM | Global metric – the mean of the keystroke duration metrics in this sample | | | | | | | | | | | | | | | | | | | | | |
| KDSD | Global metric – the standard deviation of the keystroke duration metrics in this sample | | | | | | | | | | | | | | | | | | | | | |
| DLM | Global metric – the mean of the digraph latency metrics in this sample | | | | | | | | | | | | | | | | | | | | | |
| DLSD | Global metric – the standard deviation of the digraph latency metrics in this sample | | | | | | | | | | | | | | | | | | | | | |
| KD | Selected metrics – the keystroke duration metric | | | | | | | | | | | | | | | | | | | | | |
| DL | Selected metrics – the digraph latency metric | | | | | | | | | | | | | | | | | | | | | |

Table 5.4: Example of Global and Selected Metrics for a Participants Input File

### 5.4.5 Final Analysis Procedure

The files used in the final analysis procedure were those specified in the previous section (i.e. m001.txt, m002.txt, ..., m090.txt), which contained 140 samples with 24 normalised metrics per sample. Using the participant numbers assigned during recruitment (refer section 5.3), participants were randomly allocated to the training group and the non-training group. The normalised metrics files associated with each of the participants were segregated in the same manner.

As all 140 samples in each of the non-training group members' files were available to test for correct rejection, it was considered that forty data files would provide enough data for this purpose. Accordingly, fifty participants (and their associated files) were assigned to the training group, and the remaining forty participants (and their associated files) were assigned to the non-training group.

Henceforth, those participants (and their associated data files) allocated to the training group are referred to as training group members (and training group member files). Those participants (and their associated files) allocated to the non-training group are referred to as non-training group members (and non-training group member files).

As explained in sections 5.4.5.1 and 5.4.5.2, only a subset of the samples in each training group member's file were selected for training an ANN to recognise their typing pattern; the remaining samples in that file were used to test their trained ANN for correct recognition. Also, samples from all other training group members' files (that were not used for training their own ANN) were utilised to test a member's trained ANN for correct rejection.

The point must be made clear here, that one ANN was trained to recognise the pattern within one training group member's data. So for 50 training group members, there would be at least 50 individual ANNs trained. However, as explained in the next section, the number of middle layer neurons was varied when training the ANNs for each training group member; so in fact, there were many more that 50 individual ANNs trained.

The non-training group members' files were used for testing purpose only. That is, they were used to test all training group members' trained ANNs for correct rejection. The reason for this is to demonstrate the ability of the trained ANN to accurately reject patterns in data that it has not seen before (and more importantly that these data are not falsely recognised). This is known as the ability to generalise.

The experiment then proceeded with the training and testing phases. For the description in the following two sections, Figure 5.2 has been provided to assist with understanding the creation of training and testing files[8].

#### 5.4.5.1 Training Phase

Each training group member had an input file created for training an ANN to recognise their pattern; that is, one ANN per training group member. The training file for each training group member was generated as follows (refer Figure 5.2):

- 30 samples (of the 140 available) were randomly chosen from that member's data file, for the positive training case. Samples for the positive training case are those that the ANN is intended to discern patterns within.

- 1 sample was randomly chosen from each of the other training group members data files, for the negative training case. As there were 49 other training group members, this meant 49 samples. Samples for the negative training case are those that the ANN is not intended to recognise; they are used to help the ANN recognise patterns in the positive case samples. That is, when trying to recognise a pattern, it is helpful to present patterns that are *not* meant to be recognised.

Each training group member also had a file created for cross validation during the training process. These were generated as follows:

- 10 samples were randomly chosen for that member, from the same data file from which the 30 training samples were chosen (but excluding the training samples).

[8]For convenience of illustration, Figure 5.2 shows the training group data files numbered 1 to 50 and the non-training group data files numbered from 51 to 90. As just discussed, allocation of participants and their data files to these groups was performed randomly, not sequentially.

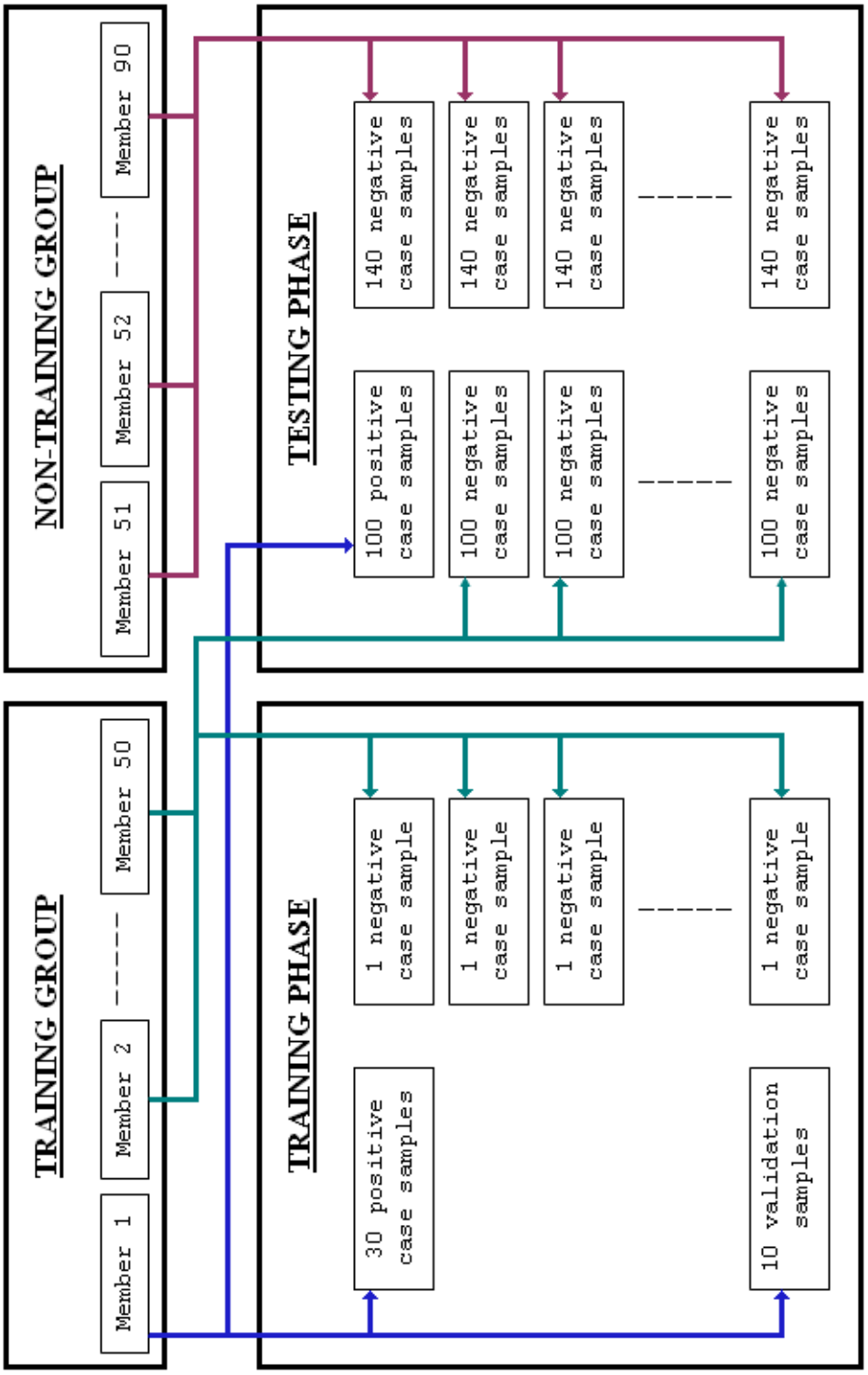


Figure 5.2: Creation of Training and Testing Files For ANN Processing

The choice of 30 positive case training samples, and 10 cross validation samples, was based on a rule of thumb suggested by Kasabov (1996). With 40 samples removed and used for training purposes, this meant that there were 100 remaining samples per training group member set aside for testing purposes.

Therefore, each training group members input file (for training) consisted of 79 input samples (30 for the positive training case plus 49 for the negative training case), with 24 metrics per sample. There were 50 such training input files (one for each participant). There were also 50 validation files (consisting of 10 samples, with 24 metrics per sample); one for each training group member, corresponding to each of their training files.

The objective of the training phase was to obtain a registered template associated with each training input file (i.e. for each participant). For the experiment, the back propagation Artificial Neural Network (ANN) architecture was used. When training the ANNs, each input layer node corresponded to each of the 24 metrics (per sample) from the training input file.

The number of hidden layer nodes was varied from 2 to 48 (inclusive), for each training input file. This was done because there is no standard rule that specifies how many hidden layer nodes should be used, and so the most appropriate ANN configuration for each member must be determined by trying different configurations and seeing which one performed best.

The reason for the upper limit of 48 hidden layer nodes was based on the fact that keystroke dynamics data demonstrates a high degree of variability. After some preliminary trial and error testing, it was thought prudent to compensate for this variability by utilising an upper limit that was double the number of input layer nodes (i.e. 24 input layer nodes times 2).

As a result, there were 2,350 individual ANNs trained (47 configurations for 50 participants). The ANN training phase for keystroke dynamics data (with the multiple configurations just described) was conducted using a desktop computer with a 2 Ghz AMD processor and 512 MB RAM. Training took 25 days (i.e. approximately 12 hours per participant).

Once training was completed, all ANN configurations were subjected to preliminary assessment to determine the single configuration (for each training group member) that returned the least number of false acceptances (Type I errors) and false rejections (Type II errors). This process was performed manually because the determination required assessing the trade-off between the two error types.

Basically, the configuration that returned the least number of Type I errors, whilst returning Type II errors at an acceptable level was selected. The same process was applied for all training group members, and the weights of the ANN configurations (thus selected) were used as registered templates from that point onwards. These were written to a file (such that the participant number and number of hidden layer nodes were indicated in the file name, and given a '.w' extension) and subsequently used during the testing phase.

#### 5.4.5.2 Testing Phase

Each training group member had an input file created for testing their trained ANN. The testing file for each training group member was generated as follows (refer Figure 5.2):

- 100 samples (i.e. 140 less those samples used in the training phase) for the member being tested were used for the positive testing case.
- 100 unused samples from each of the other training group members were used for the negative testing case.
- 140 samples from each of the non-training group members were used for the negative testing case.

Positive case testing examines whether the ANN has correctly recognised samples belonging to the member that it has been trained to recognise (samples it has not seen during training). Non-recognition of any of these 100 positive case samples are instances of Type II errors (false negatives) (refer Chapter 6 section 6.2).

Negative case testing examines whether the ANN has correctly rejected samples belonging to someone other than the member it has been trained to recognise.

Recognition of any of these negative case samples are instances of Type I errors (false positives) (refer Chapter 6 section 6.2).

So for each training group member, the samples from each relevant file (in the order listed above) were read and written to their testing input file. Therefore, each training group member had a testing input file consisting of 10,600 input patterns (i.e. 100 for the member being tested, plus 100 for each of the other 49 training group members, plus 140 for each of the 40 non-training group members), with 24 metrics per sample. There were 50 such testing input files (one for each training group member). The results of the testing phase are provided in Chapter 6, and a discussion of these results is presented in Chapter 7.

## 5.5 Fingerprint Recognition

### 5.5.1 Software

The software required for the experiment was developed and implemented according to the requirements listed below:

1. A program to capture fingerprint scans and extract fingerprint feature data (refer section 5.5.2).

2. Programs to align the extracted fingerprint feature data according to their position, and to select those features used in the experiment (refer sections 5.5.4 and 5.5.5).

3. Programs to create the files for training and testing the ANNs (refer section 5.5.6).

4. Programs to execute training and testing of ANNs (refer section 5.5.6).

5. Programs to calculate results from the ANN testing output files.

Summaries of all programs are provided in Appendix C section C.2. Where appropriate, algorithms are provided in the following sections to specify the necessary processing.

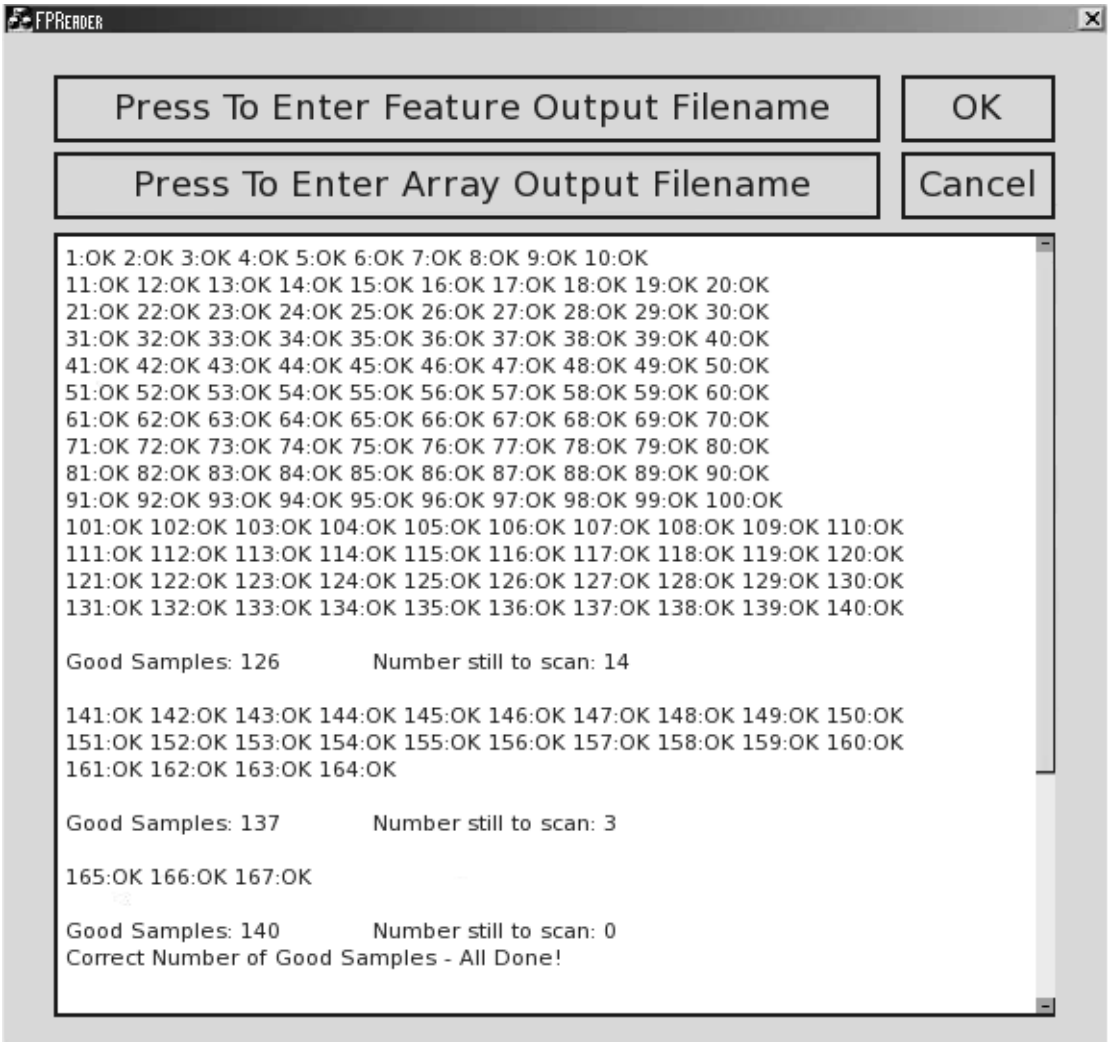


Figure 5.3: Graphical User Interface for Fingerprint Feature Capture Program

The program used to capture fingerprint feature data was developed using Visual C++. This program presented participants with a Graphical User Interface (GUI) to facilitate the fingerprint scanning procedure (see Figure 5.3), although all data collection sessions were supervised by the author. The program utilised the Verifinger Software Development Kit (SDK) 4.2 Visual C++ library to interface with the fingerprint scanning device and to extract the fingerprint features. Thus the first two stages of the AFIS (acquisition and feature extraction) utilised third party hardware and software. The program developed by the author for data collection was done at the application level.

It was a condition of the Ethics Committee approval that no actual fingerprint scans be permanently recorded. Therefore, only extracted feature data was recorded. The extracted feature data from all scans were recorded, even though identified

outlier samples were not used in the experiment. Outlier samples were determined as those whose minutiae count was not within one standard deviation of the mean number of minutiae for the same participant. Each participants extracted feature data was written to a uniquely named file (refer section 5.5.2).

The program to align or register the feature data was written in the C programming language. Registration entailed the alignment of the local feature data according to their positions in the two dimensional coordinate plane. This meant that a common alignment of feature layout was achieved, so that all samples from a particular participant could be compared (see section 5.5.4).

Once the features were aligned, those features present in 140 samples (per participant) were used to select features for use in the experiment (refer section 5.5.4). Once the requisite features were identified, they were extracted and written to uniquely named files.

The programs to create the files for training and testing the ANNs were implemented as Perl scripts, and ensured that no data handling errors were introduced. The task was broken into the following sub-tasks (refer section 5.5.6):

- Randomly assign each participants feature data file to the training or non-training group.
- Randomly select samples from each of the training group members feature data files.
- Create ANN training input files, and cross validation files, for each training group member.
- Create ANN testing input files for each training group member.

Training and testing was again performed utilising the Matrix Back Propagation ANN (Anguita, 2007).

The same program used to prepare the ANN outputs for results analysis in the keystroke dynamics phase (mentioned in section 5.4.1) was utilised again to prepare the ANN outputs for results analysis in the fingerprint recognition phase.

### 5.5.2 Data Collection

Fingerprint scans of each participants right index finger were captured using the Digital Persona U.ARE.U 4000 optical fingerprint scanner. Each scan resulted in a binary raster image of the fingerprint. This was represented as a grey-scale 320 by 350 pixel matrix, with each pixel consisting of a byte. However, only the features from each scan were extracted (in real time during the scanning process), and therefore only the features were recorded for use in the experiment. That is, the image of the scan was only stored temporarily in memory, to facilitate feature extraction.

The reasons for recording only fingerprint features for the experiment were:

- In light of the sensitivity associated with storing fingerprints, gaining Ethics Committee approval to retain full fingerprint images would have been a more involved and convoluted process.

- It was believed that gaining the appropriate number of participants would have been difficult unless they were convinced that their fingerprint images were not being stored.

- As mentioned in section 4.4.2, there are two schemes adopted for representation of a captured fingerprint image. One utilises grey-scale images and the other utilises extracted features. It was always the intention of the author to utilise extracted local features (minutiae) for verification, as the experiment relied on the detail available from such data. Therefore, the choice was made according to the intent of the treatment of the data.

In terms of the stages of an AFIS, pre-processing and feature extraction (stages 2 and 3 discussed in Chapter 4 sections 4.4.3 and 4.4.4) were performed by third party software. The author utilised the Verifinger Software Development Kit (SDK) 4.2 Visual C++ library to interact with the above mentioned scanner and to extract the fingerprint features (refer section 5.5.3). Any further treatment of the extracted fingerprint features was performed to represent the data for future use in the data fusion phase of the experiment (refer section 5.5.4).

As the fingerprint feature treatment procedure was to be newly developed, it was necessary to determine the correct number of samples to be collected from participants. The quantity of 140 samples was determined using data collected from a pilot study (Abernethy et al., 2005) and the following statistical formula (Brase and Brase, 2004):

$$N = (\frac{Z_c \times \sigma}{E})^2 \tag{5.1}$$

where $Z_c$ is the z value of the confidence required; $\sigma$ is the largest standard deviation value from the metrics in the pilot study; $E$ is an acceptable error. Whilst the result of the calculation was approximately 136, this was rounded to 140 for convenience, and because it matched the number of samples collected for the keystroke dynamics phase of the experiment (there being an intention to combine data sets).

Participants were asked to provide 140 scans of their right index finger, in one collection session. However, because of scanner inaccuracy and variability associated with collection of biometric data, participants on average had to provide approximately 200 scans in order to obtain the required 140 'representative scans'[9]. This took each participant approximately 10 to 15 minutes, depending on how many samples were actually collected.

The procedure for determining whether a scan was representative of a participant was based on statistical measures, as follows:

1. The mean and standard deviation of the number of minutiae for the first 140 scans were calculated.

2. Representative scans were identified as those where the minutiae count was within one standard deviation of the mean.

3. Participants were asked to provide additional scans to replace non-representative scans (i.e. those scans where the minutiae count was outside one standard deviation of the mean).

[9]Although only data from representative scans were subsequently used in the experiment, data from all input samples were recorded.

Steps 2 and 3 were repeated (utilising the same mean and standard deviation calculated in step 1) until 140 samples (considered representative) were obtained.

The scanner platen was cleaned after every 20 scans. Whilst cleaning was being performed, participants could wipe their finger with a clean cloth that was on hand, if they noticed moisture build up on their finger. It is now believed that a contributing factor for participants having to provide so many scans (in order to obtain their 140 representative samples), was inaccuracy caused by build up of bodily fluid on the scanner platen during the scanning procedure. In hindsight, it may have been more appropriate to clean the scanner platen after every 10 scans.

For each participant, feature information contained in all of their scans (representative and non-representative) were written to a uniquely named file in a binary format (by the Verifinger SDK). File names were numbered according to the participant numbers assigned during recruitment (refer section 5.3), with each file name followed by the extension '.ftr' (i.e. 001.ftr, 002.ftr, ..., 090.ftr).

### 5.5.3 Fingerprint Feature Extraction

Using the Verifinger SDK to extract features from the files obtained in the previous section (i.e. 001.ftr, 002.ftr, ..., 090.ftr), two global and all local feature information were accessible for extraction. The available global features were the ridge count and the minutiae count. The local feature information included (refer section 4.3.2):

- The minutia type. The type was returned as either ridge termination (designated type 1) and ridge bifurcation (designated type 2).
- The position. This information was returned as x and y coordinates in the two dimensional plane. The boundaries of the plane were 0 to 319 along the x axis and 0 to 349 along the y axis.
- The spatial frequency (i.e. the average distance between ridges in the neighbourhood of a minutia point).
- The orientation of the ridge that approaches a minutia point (i.e. the angle between the tangent to the ridge at a minutia position and the horizontal axis) - refer section 4.3.2.

- The curvature of the ridge as it approaches a minutia point (i.e. the rate of change of the ridge orientation as the ridge approaches a minutia point) - refer section 4.3.2.

The extracted fingerprint features from each scan were written to a uniquely named file for each participant. File names were numbered according to the participant numbers assigned during recruitment (refer section 5.3), with each file name preceded by the character 'm' and followed by the extension '.txt' (i.e. m001.txt, m002.txt, ... m090.txt). Once features were extracted and recorded, their alignment was necessary as discussed in the next section.

### 5.5.4 Local Feature Registration

When performing multiple scans of a finger, it is highly improbable that the finger will be placed in exactly the same position for any two scans (Digital Persona, 2004). This means that the features of a fingerprint image at scan $n$, will almost always be in different positions than the features of an image of the same finger at scan $n+1$. However, though the features may be in different absolute positions according to their x and y coordinates, they should still maintain their positions relative to each other, except for minor distortions as discussed in Chapter 4 section 4.3.2 and 4.4.3. Therefore, alignment of the features extracted from the images becomes necessary to render them rotation, scale and translation invariant. This process is termed registration.

As data collection for this study entailed extraction and storage of fingerprint features only, the registration process required the transformation of the local feature data sets extracted from images, rather than the images themselves. Given that these feature data sets represent points in a two dimensional plane, positioned according to their x and y coordinates, the only information that required transformation were the feature coordinate positions (the corresponding attributes of each feature needed no treatment, so long as the association between them remained intact).

So the registration process became an exercise in point pattern matching to determine the transformation factors that would align the points (features) in one feature data set to corresponding points (features) in another feature data set (provided they were from the same finger).

In the field of image registration, a model image is used to align other images of the same scene that may have been displaced by rotation, scale and/or translation. The processes utilised (in the current study) for the selection of the model image and the registration of scene images, are discussed in detail in the next two sections 5.5.4.1 and 5.5.4.2.

However, before elaborating on the process of registering the fingerprint feature sets, a simple example is presented to demonstrate the concept of registration. The example is illustrated in Figure 5.4, which consists of 6 images (a to f). Also, Table 5.5 presents the actual coordinate positions (rounded to 2 decimal places) of the points corresponding to those in the 6 images shown in Figure 5.4. Headings in Table 5.5 indicate the appropriate data set corresponding to each of the images.

**Image (a)** presents the model data set containing 5 hypothetical points.

**Image (b)** presents the scene data set with the same 5 points displaced by rotation, scale, and translation.

**Image (c)** shows the scene data set rotated by an angle of $15^0$.

**Image (d)** shows the rotated scene data set translated along the x axis by 75.0.

**Image (e)** shows the resultant scene data set (from the previous 2 transformations) translated along the y axis by $-50.0$.

**Image (f)** shows the resultant scene data set (from the previous 3 transformations) scaled by a factor of 1.05.

Therefore, image (f) illustrates the final location of the 5 points from the original scene data set after all transformations were performed. Note that in image (f) the final coordinates of the 5 points in the scene data set (after transformation) are not identical to the coordinates of the corresponding points in the model data set, but are relatively close. The minor differences can be attributed to lose of precision during processing.

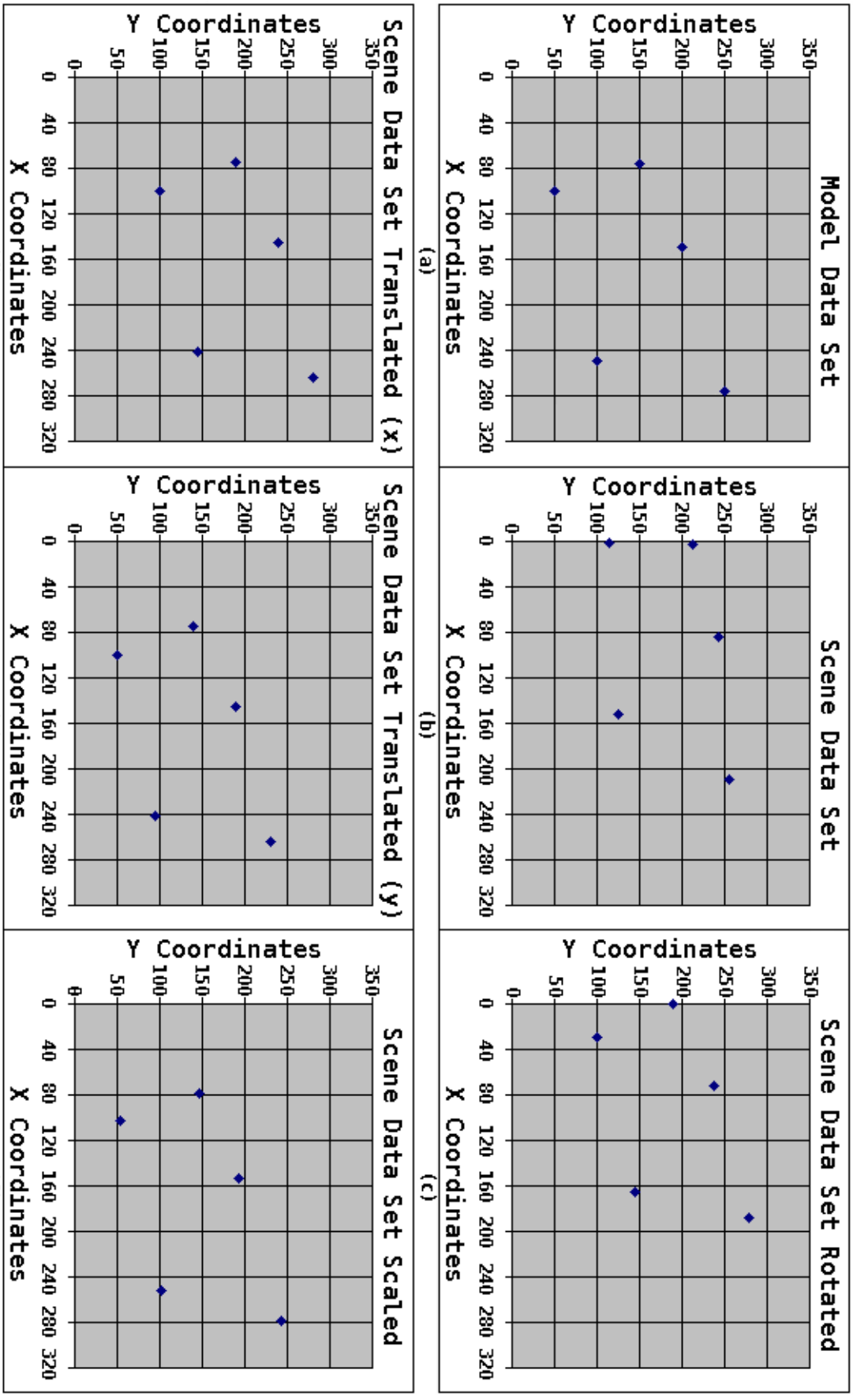


Figure 5.4: Alignment Example

| Model Data Set (a) | | Scene Data Set (b) | |
|---|---|---|---|
| X Coordinates | Y Coordinates | X Coordinates | Y Coordinates |
| 150.0 | 200.0 | 84.19 | 240.79 |
| 250.0 | 100.0 | 151.37 | 124.45 |
| 75.0 | 150.0 | 3.07 | 213.36 |
| 275.0 | 250.0 | 211.19 | 255.95 |
| 100.0 | 50.0 | 1.43 | 115.45 |

| Scene Data Set Rotated (c) | | Scene Data Set Translated (d) | |
|---|---|---|---|
| X Coordinates | Y Coordinates | X Coordinates | Y Coordinates |
| 71.25 | 234.15 | 146.25 | 234.15 |
| 166.25 | 146.35 | 241.25 | 146.35 |
| 0.00 | 189.20 | 74.99 | 189.20 |
| 189.99 | 279.52 | 264.99 | 279.52 |
| 23.75 | 100.78 | 98.75 | 100.78 |

| Scene Data Set Translated (e) | | Scene Data Set Scaled (f) | |
|---|---|---|---|
| X Coordinates | Y Coordinates | X Coordinates | Y Coordinates |
| 146.25 | 184.15 | 153.56 | 193.36 |
| 241.25 | 96.35 | 253.31 | 101.17 |
| 74.99 | 139.20 | 78.75 | 146.16 |
| 264.99 | 229.52 | 278.25 | 240.99 |
| 98.75 | 50.78 | 103.69 | 53.32 |

Table 5.5: Example Registration Tables

#### 5.5.4.1 Model Feature Set

In the current study, the method utilised to determine a model feature set for each participant was based on the average distance between local features in feature sets. That is, in a feature set—belonging to an individual participant—the distance between a local feature and all other local features in that feature set was calculated (according to Equation 5.2 on page 267). This process was repeated for all local features in the same feature set. For example, if a feature set contained 30 local features, then 30 distance measures would be determined (from each of the 30 local features to all other local features in that feature set).

Once all distances (from all local features to all other local features in a feature set) were determined, an average distance was then calculated for that feature set (based on the determined distances). This process was performed for all feature sets belonging to an individual participant.

A comparison was then made between the average distances for all feature sets (i.e. those from all scans) for that participant, and the feature set corresponding to the smallest average distance was nominated as the model feature set. The rationale for this decision was that the feature set with the smallest average distance would most probably be the most representative of the feature sets (when compared to the other feature sets belonging to the same participant), upon which to base feature alignment. This model feature set was then used in the registration of all of that participant's feature sets. The same process was applied in determining a model feature set for all participants.

Whilst this process was computationally expensive, the only other known option for selecting a model feature set for each participant was to graph all their feature sets and visually determine the most representative feature set. Comparing the characteristics of all feature sets to accurately choose the appropriate feature set was considered too onerous and prone to errors in judgment. Doing so for all 90 participants was considered an impractical and unrealistic solution, and not reliably repeatable.

#### 5.5.4.2 Scene Feature Set Alignment

**Overview**

The point pattern matching algorithm utilised to align participants' scene feature sets to their model feature set was that presented by Van Wamelin et al. (2004). The authors compared the different point pattern matching algorithms developed up to 2004, and demonstrated how their algorithm improved efficiency and accuracy in the registration process (Van Wamelen et al., 2004). The algorithm accommodates rotation, scale and translation variations and importantly is robust to missing data and/or introduced artifacts. These qualities were influential in the selection of the algorithm for this study.

The algorithm works by firstly attempting to find transformation factors that match feature sets based on a local area (typically central to the model feature set) (Van Wamelen et al., 2004). That is, factors that will transform a subset of points

in a scene feature set to a subset of corresponding points in the model feature set. If transformation factors are found that satisfy a threshold condition, then these factors are applied to the entire feature set for that scene.

However, transformation factors that provide the most accurate alignment based on a local area, do not necessarily result in the most accurate alignment for the entire feature set. Therefore, global error minimisation is applied to refine the local transformation factors (by making minor adjustments), with the aim to improve alignment of the scene feature set with the model feature set on a global basis.

To demonstrate the concept of local area alignment, a simplified example is provided in Figure 5.5. Also, Table 5.6 provides the actual coordinates for the model and scene image points (accurate to 2 decimal places). There are 6 images (a to f) in Figure 5.5, where the model image (a) contains 4 hypothetical points $(p, q, r, s)$. The central most point $q$ forms 3 dashed line segments $\bar{pq}$, $\bar{qr}$, and $\bar{qs}$ with the other 3 points $(p, r, s)$. Line segments in the model image are dashed to distinguish them from those in the scene images. The red coloured line segment $\bar{pq}$ is designated the primary line segment upon which comparison, with line segments in the scene images, is based during the alignment process.

Images (b), (c), (d), and (e) are all instances of the same scene image, where the 4 hypothetical points (from the model image) have been transformed by rotation, scale and translation factors. The transformation factors applied to the model image points to relocate them to their coordinates in the scene images were a rotation angle of $\theta = -90^0$, a scale factor of $s = 0.95$, and translation factors of $x = 40$ and $y = -30$. It should be noted that these transformation factors were nominated purely to provide a demonstration of local area alignment; they were not meant to approximate a genuine case of local area alignment.

Each of the four scene images (b, c, d, and e) illustrate the process of determining 3 line segments from one point in the scene image to the other 3 points in the same image. Each image does this from a different starting point. For example, image (b) performs the task from the point $p$ $(192.0, 60.25)$, image (c) from the point $q$ $(192.0, 136.25)$, image (d) from the point $r$ $(120.75, 174.25)$, and image (e) from the point $s$ $(263.25, 174.25)$ (refer Table 5.6).

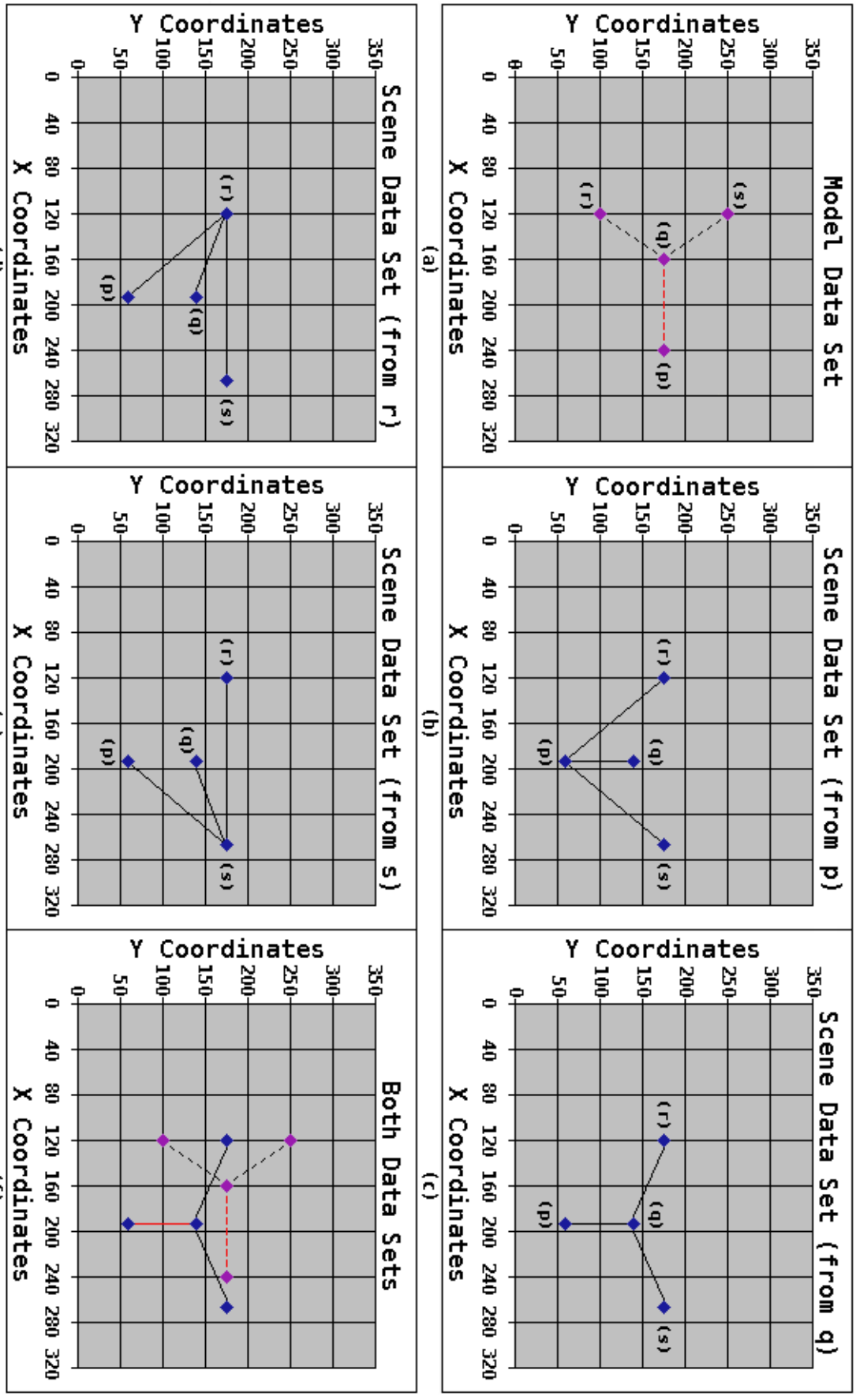


Figure 5.5: Local Area Alignment Example

| Point | Model Data Set | | Starting Point In Scene Data Set | |
|---|---|---|---|---|
| $q$ | 160.0 | 175.0 | 192.0 | 136.25 |
| $p$ | 240.0 | 175.0 | 192.0 | 60.25 |
| $r$ | 120.0 | 100.0 | 120.75 | 174.25 |
| $s$ | 120.0 | 250.0 | 263.25 | 174.25 |

Table 5.6: Local Area Alignment Coordinates

The alignment process involves comparing each of the line segments (in a scene image) with the primary line segment in the model image. For example in Figure 5.5 image (b), line segments $\bar{pq}$, $\bar{pr}$, and $\bar{ps}$ are compared (in turn) with the primary line segment $\bar{pq}$ in the model image. Firstly, transformation factors are determined that align $\bar{pq}$ in the scene image with $\bar{pq}$ in the model image. All points in the scene image are transformed according to the determined factors, and the distance between all transformed scene image points and the model image points is calculated (according to Equation 5.2 on page 267). If that distance meets a threshold condition, then a candidate set of transformation factors is indicated and they (and the corresponding distance) are recorded. If not, they are discarded. The same process is applied to the other two line segments $\bar{pr}$ and $\bar{ps}$ in image (b).

It should be clear from visually comparing image (b) with image (a) that no transformation factors successfully align the two images, based on the comparison between the three line segments $\bar{pq}$, $\bar{pr}$, and $\bar{ps}$ in image (b) and the primary line segment in image (a).

The same process is applied to the other scene images (as demonstrated in Figure 5.5 images (c), (d), and (e)), where for each image the starting point (for determining the line segments) is different.

At the completion of this process, the candidate transformation factor set associated with the smallest distance is nominated the 'winning' set, as the smallest distance between points indicates the closest alignment. These transformation factors are applied to all points in the scene image, and should result in the most accurate alignment (on a local basis).

In the example illustrated in Figure 5.5, image (c) should provide the winning transformation factors. To demonstrate this, image (f) shows the model image and

imposes image (c) on top of it so that the correspondence is evident (taking into consideration that the transformation factors required to align the two images have not been applied in image (c). If they had been, the imposed image would cover the model image).

The following sub-sections explain in greater detail (with the use of formulae and algorithms) the alignment process. A description of the software that was developed in the C programming language is presented. Slight modifications, to the Van Wamelin et al (2004) algorithm, were made during implementation to accommodate different data storage methods within the program. This did not change the method of the algorithm, but was done merely for implementation purposes. Also, some preliminary steps (in relation to the storage of feature sets and the calculation of the threshold) were required to facilitate the registration process.

### Feature Set Storage

The implementation stored information about each point of a feature set in a record along with other information used during the registration process. An array of such records was used to store the information about all points in a feature set. As the program was developed using the C programming language, the data structure called a 'struct' was used for each record.

The struct elements were:

1. x - the x coordinate for the current point.

2. y - the y coordinate for the current point.

3. pointNumber - the point number of the current point in the feature set. The point number started at the number 0 to correspond with the index number of the array storing the structs. In C, array elements start at index 0.

4. type - the minutia type of the current point.

5. distance[] - the distances from the current point to all other points in the feature set, stored in an array. Each distance $d_j$ was calculated as the Euclidean distance (refer Equation 5.2) between the point $p_{pointNumber}$ and the remaining

points in the same feature set ($p_j$,$p_{j+1}$,...,$p_{n-1}$ in $n$-space). The distances were sorted in ascending order and stored in successive array elements.

$$d_j = \sqrt{(p_i - p_j)^2} \tag{5.2}$$

where $j = 0, 1, \ldots, n$ and $i \neq j$.

6. indices[] - the point numbers of the points in distance[] (struct element 5). Stored in array elements according to the their corresponding sorted distances.

7. size - the size of the arrays used for struct elements 5 and 6.

8. isMatched - a boolean sentinel value to indicate when the current point gets matched to a point in another feature set. Used during the alignment process, this element is initialised to –1, and is set to 1 if the current point gets matched.

9. matchPoint - the point number of the point in the other feature set that the current point gets matched to.

10. matchDistance - the Euclidean distance to the point that the current point gets matched to.

**Threshold**

A threshold value was used to determine whether two points from different feature sets were within an acceptable proximity to each other. Van Wamelin et al., (2004) suggested that if all model feature set points are encompassed by an imaginary circle, then it seems reasonable to assume that the average distance to the nearest neighbour of a given point is within a smaller circle calculated in relation to the radius of the encompassing circle and the number of points in the model feature set (refer Equation 5.3).

$$\frac{r}{2 \times \sqrt{n}} \tag{5.3}$$

where $r$ is the radius of the imaginary circle encompassing all points in the model feature set, $n$ is the number of points in the model feature set.

It also seems reasonable that a constant fraction of this value could be used to estimate an acceptable threshold of proximity (Van Wamelen et al., 2004). This can be calculated according to Equation 5.4.

$$t = \lambda \left( \frac{r}{2 \times \sqrt{n}} \right) \tag{5.4}$$

where $t$ is the calculated threshold, $r$ is the radius of an imaginary circle encompassing all points in the model feature set, $n$ is the number of points in the model feature set, and $\lambda$ is a matching factor.

The nature of data in this experiment demonstrated a substantial variation in the value for $n$ (from a minimum of 18 to a maximum of 69 across the 90 participants). That is, the distribution of points across the model feature sets was disparate.

Intuitively, it seems appropriate to allow the threshold for a sparse distribution to be greater than the threshold for a dense distribution. That is, it seems acceptable to allow two points to be further apart in a sparse distribution than in a dense distribution, and still meet the threshold condition. Conversely, it seems acceptable to require two points to be closer together in a dense distribution than in a sparse distribution, in order to meet the threshold condition.

For example, in a sparse distribution (such as 18) relaxing the acceptable distance between two points seems appropriate, as the points are further apart than they are in a dense distribution (such as 69). Similarly, tightening the acceptable distance in a dense distribution (such as 69) seems appropriate, as the points are closer together than they are in a sparse distribution (such as 18).

Therefore, because of the differences in the density of distributions in this experiment, it was considered advisable to adjust the value of $\lambda$ in relation to the value of $n$. Accordingly, $\lambda$ was adjusted (in relation to $n$) as follows:

If $n \leqslant 20$, $\lambda = 0.535$

If $20 < n < 50$, $\lambda = 0.428$

If $n \geqslant 50$, $\lambda = 0.321$

The value 0.428 for $\lambda$ was the suggested value by Van Wamelan et at., (2004). The other two possible values for $\lambda$ are equidistant from the suggested value of 0.428[10] and were determined by trial and error. It can be seen that the values for $\lambda$, when applied to the calculation of the threshold, reflect the reasoning in the above discussion.

Given Equation 5.4 and the above inequalities, the threshold between two points is likely to be greater if $n$ was less than 21, than it would be if $n$ was greater than 49. Conversely, the threshold between two points is likely to be smaller if $n$ was greater than 49, than it would be if $n$ was less than 21. Of course, the value for $r$ may also influence the outcome.

The following two sub-sections explain the alignment process in detail. Algorithms 5.2, 5.3, and 5.4 have also been provided to help understand the implementation.

**For the Model Feature Set**

- Determine, sort and store distances (and corresponding point numbers) in the appropriate arrays for all points in the model feature set $M$, as previously described for struct elements 5 and 6.

- Randomly select 5 points within the central region of $M$. Each of these 5 points were used as a starting point for the alignment process, in case one or more of these 5 points do not result in an acceptable alignment. Algorithm 5.2 demonstrates the practical method used to implement the selection of the 5 points. The algorithm firstly used the modulus operator to determine if the number of points in the model feature set $M$ was even or odd[11].

  If there were an even number of points in the model feature set, that number divided by 2 obtained the central point, and the point number of that central point was stored in the *random*[] array at index 0 (Steps 1—2 in Algorithm 5.2). If there were an odd number of points in the model feature set, that

[10] Plus or minus 0.107

[11] The modulus operator determines the remainder of an integer or whole number division. Therefore, if any integer when divided by 2 results in a remainder of 0, then that integer is even (otherwise it is odd).

number subtract 1 and then divided by 2 obtained the approximate central point, and that point number was stored in the *random*[] array at index 0 (Steps 3—4 in Algorithm 5.2). Once obtained, this point number was used to determine four other points in the region approximately central to $M$ (Steps 6—9 in Algorithm 5.2).

**Algorithm 5.2** Selection of 5 Random Points Central to $M$

Let *mlength* be the number of points in the model image feature set $M$.
Let *random* be a 5 element array.
Let the symbol % signify the modulus operator.

1: **If** $mlength$ % $2 == 0$
2: $\quad random[0] \leftarrow mlength/2$
3: **Else**
4: $\quad random[0] \leftarrow (mlength - 1)/2$
5: **End If-Else**
6: $random[1] \leftarrow random[0] - 1$
7: $random[2] \leftarrow random[0] + 1$
8: $random[3] \leftarrow random[0] - 3$
9: $random[4] \leftarrow random[0] + 3$

**For Each Scene Set**

- Determine, sort and store distances (and corresponding point numbers) in the appropriate arrays for all points in the current scene feature set $S$, as previously described for struct elements 5 and 6.

- Determine candidate local transformation factors that accurately transform a small subset of points in $S$ to a corresponding subset of points in $M$, as described below[12]:

  - Firstly, access the randomly selected point $random[0]$ in $M$ and assign to $p$ (Step 2 in Algorithm 5.3); access the 6th closest point to it[13] and assign to $a$ to determine the line segment $\bar{pa}$ in $M$ (Step 4 in Algorithm 5.3). Next, access the first point in $S$ and assign to $q$ (Step 6 in Algorithm

[12]The description makes reference to steps listed in Algorithm 5.3 on page 276 (Steps 1 to 40).
[13]The reason for accessing the 6th closest point to $p$ is that the distance from $p$ to its 6th closest neighbour should determine a line segment long enough to allow for the calculation of reasonably accurate local transformation factors. Points closer to $p$ may not provide the same accuracy.

5.3); access the 6th closest point to it[14] and assign to $b$ to determine the line segment $\bar{qb}$ in $S$ (Step 8 in Algorithm 5.3). Then determine local transformation factors $s$ (scale), $\theta$ (angle), $tx$ (translation along the $x$ axis), and $ty$ (translation along the $y$ axis) between the two line segments $\bar{pa}$ and $\bar{qb}$ (Steps 9 to 14 in Algorithm 5.3) according to Equations 5.5, 5.6, 5.7, and 5.8.

$$s = \frac{\sqrt{(ax - px)^2 + (ay - py)^2}}{\sqrt{(bx - qx)^2 + (by - qy)^2}} \tag{5.5}$$

$$\theta = atan2(ay - py, ax - px) - atan2(by - qy, bx - qx) \tag{5.6}$$

$$tx = px - qx \times s \times cos(\theta) + qy \times s \times sin(\theta) \tag{5.7}$$

$$ty = py - qx \times s \times sin(\theta) - qy \times s \times cos(\theta) \tag{5.8}$$

where $x$ and $y$ are the associated x and y coordinates of the points $p, a, q, b$.

- By iterating from the 6th to 10th (inclusive) closest points to $p$ in $M$, 5 different line segments $\bar{pa}$ were determined. As there were 5 possible random points for $p$, there were 25 line segments determinable in $M$. Also, by iterating from the 6th to 10th closest points to $q$ in $S$, 5 different line segments $\bar{qb}$ were determined. The total number of line segments determinable in $S$ was dependent on the number of points in $S$. For example, if $S$ contained 30 points, the number of line segments $\bar{qb}$ would be 150. A different set of local transformation factors $s, \theta, tx, ty$ were determined between all successive line segments $\bar{qb}$ in $S$ and all successive line segments $\bar{pa}$ in $M$. In the above example, the total number of local transformation factor sets calculated would be 3,750 (25x150).

- Obviously only a few of the local transformation factor sets will result in accurate alignment. In order to determine candidate local transformation factor sets resulting in an acceptably accurate registration, the method suggested by Van Wamelin et al, (2004) was adopted. Firstly, the local

[14] Again, the reason for accessing the 6th closest point to $q$ is to determine a line segment long enough to allow for the calculation of reasonably accurate local transformation factors.

transformation factors were restricted to lie within the boundaries listed in Table 5.7 (also refer to Algorithm 5.3 Step 15). Because data collection was supervised, it was confidently concluded that transformation factors outside the the ranges specified in Table 5.7 could not realistically achieve accurate registration. The scale factor is affected by the pressure applied when the fingertip comes into contact with the scanner surface. From observation of some preliminary registration attempts, the limits provided in Table 5.7 were determined. For the angle $\theta$, supervised data collection meant that participants should not have deviated from the vertical (in relation to the scanner surface) by more than $30^0$ in either direction. For the vertical and horizontal translation, if participants deviated by more than the specified ranges then one quarter of the fingerprint would not have been captured (in which case registration would have been very doubtful). In any of these cases, if any factor was outside the ranges specified in Table 5.7, then that local transformation factor set was not considered further.

| FACTOR | LOWER BOUNDARY | UPPER BOUNDARY |
|---|---|---|
| $s$ | 0.94 | 1.06 |
| $\theta$ | $-30^0$ | $30^0$ |
| $tx$ | $-80$ | 80 |
| $ty$ | $-87.5$ | 87.5 |

Table 5.7: Boundary Limits For Candidate Transformation Factors

– If the local transformation factors $s, \theta, tx, ty$ met the above restrictions, the closest nine points to $q$ in $S$ (and $q$ itself) were transformed according to Equations 5.9 and 5.10 (Steps 16 to 24 in Algorithm 5.3).

$$qx = tx + px \times s \times cos(\theta) - qy \times s \times sin(\theta) \tag{5.9}$$

$$qy = ty + px \times s \times sin(\theta) + qy \times s \times cos(\theta) \tag{5.10}$$

where $p$ is the current point in $M$, $q$ is the current point in $S$, $x$ and $y$ are the associated x and y coordinates of the associated points $p$ and $q$.

- The number of matching points $m$ were then determined between the 10 points in $S$ just transformed and the current point $p$ in $M$ and its closest nine neighbours (Step 20 in Algorithm 5.3). Matching criteria was based on two points meeting the threshold condition ($t$) calculated according to Equation 5.4.
- If there were six points matching between the two subsets of $M$ and $S$ (using the matching probability of $\rho = 0.6$ suggested by Van Wamelin et al, (2004)), it was considered that the local transformation factor set was a definite candidate set, in which case the average distance was calculated (Steps 26 and 27 in Algorithm 5.3). The average distance $ad$ was calculated between all points in the subset of $M$ and subset of $S$, where distances were determined according to Equation 5.2.
- If the average distance $ad$ was less than a required distance $rd$ (initially $rd \leftarrow 1000$[15]), the required distance was updated to that of the average distance and the local transformation factors were stored (Steps 28 to 31 in Algorithm 5.3).
- At this point, only the local transformation factors that achieved the best alignment were stored. The entire process was repeated to access the next randomly selected point in the $random[]$ array (in $M$), which was assigned to $p$ (Step 2 in Algorithm 5.3). This was repeated until all 5 random points had been accessed.

- At the end of the previous processing, only the one local transformation factor set that resulted in the lowest required distance was stored for use in further processing.

- If the required distance $rd$ was less than 1000 (Step 41 in Algorithm 5.3), a definite candidate local transformation factor set had been found and was

[15]The reason for this initial required distance can be explain as follows. Because of the resolution of the scanned fingerprint image, there were 320 units along the x axis, and 350 along the y axis. This results in 112,000 possible unit intersections. As the local alignment was conducted for 10 points in both feature sets, 1000 divided 10 would mean that (on average) each subset point (in $S$) needed to be within 100 units of its prospective matching subset point (in $M$). This seemed a reasonable expectation when attempting to determine candidate local transformation factor sets.

applied to all points in $S$. A global error minimisation was then applied to all points in $S$ (Steps 42 to 46 in Algorithm 5.3).

- Apply candidate local transformation factors $s, \theta, tx, ty$ to the entire scene feature set $S$.
- Determine the number of matching points $m$ between the entire scene feature set $S$ and the entire model feature set $M$. Matching criteria was based on two points meeting the threshold condition ($t$) calculated according to Equation 5.4, as applied to all points in scene set $S$ (transformed by the local transformation factors) and all points in the model feature set $M$.
- Apply global error minimisation to $s, \theta, tx, ty$ using the ranges specified in Table 5.8 (refer Algorithm 5.4).

| FACTOR | LOWER BOUNDARY | UPPER BOUNDARY | INCREMENT |
|---|---|---|---|
| $s$ | 0.98 | 1.02 | 0.005 |
| $\theta$ | $\theta - 5^0$ | $\theta + 5^0$ | $0.25^0$ |
| $tx$ | $tx - 5$ | $tx + 5$ | 0.25 |
| $ty$ | $ty - 5$ | $ty + 5$ | 0.25 |

Table 5.8: Global Adjustment Ranges

- Determine again the number of matching points $m$ between the entire scene feature set $S$ and the entire model feature set $M$[16].
- Output the following information to three uniquely named files (per participant):
  1. The adjusted coordinates (and their associated attributes) for the scene feature set $S$. File names were preceded by 'm' and numbered from 001 to 090, with a '.txt' extension (i.e. m001.txt, m002.txt, ..., m090.txt).

[16]Matching criteria was again based on two points meeting the threshold condition ($t$) calculated according to Equation 5.4, as applied to all points in scene set $S$ (after global error minimisation) and all points in the model feature set $M$.

2. The index numbers of points in the scene feature set that were matched to points in the model feature set. A full description of this output format is provided in the next section 5.5.5. File names were numbered from 001 to 090, with a '.tab' extension (i.e. 001.tab, 002.tab, ..., 090.tab).

3. Information related to local transformation including the local transformation factors along with the resultant number of matches ($m$) and the average distance ($ad$) between the two feature sets based on those factors. Also, the adjusted transformation factors after global error minimisation, the resultant number of matches ($m$) and the average distance ($ad$) based on those factors. File names were numbered from 001 to 090, with a '.err' extension (i.e. 001.err, 002.err, ..., 090.err).

- If the required distance $rd$ was greater than or equal to 1000, a candidate local transformation factor set had not been found. In this case, no further processing was performed for the current scene feature set as registration was unlikely. Recall from section 5.5.2, that feature information from all fingerprint scans (both representative and non-representative) were recorded. It was very possible that a small number of the non-representative scans (from each participant) may not have been accurate enough to be registered. If so that scene feature set was removed from the experiment. As there were in excess of the required representative scans available, this was considered a better option than attempting further processing in order to align the scene feature sets corresponding to non-representative scans[17].

[17] As testimony to the accuracy of the Digital Persona U.ARE.U 4000 optical fingerprint scanner and the Verifinger (SDK) (feature extraction software), this particular scenario occurred rarely during alignment process in this experiment.

**Algorithm 5.3** Fingerprint Feature Registration Method

Let $slength$ be the number of points in the scene feature set $S$.
Let $index, i, j, k, h, l$ be loop control variables for accessing array elements.
Let $pIndex$ store the point number of the current point $p$ in $M$, successively accessed from $random[]$; $aIndex$ successively store the point numbers of the 6th to 10th closest points to the current point $p$ in $M$; $qIndex$ successively store the point numbers for all points $q$ in $S$; $bIndex$ successively store the point numbers of the 6th to 10th closest points to the current point $q$ in $S$.
Let $\rho = 0.6$, where $\rho$ is the matching probability suggested by Van Wamelin et al, (2004).
Let $m$ be the number of matching points between two sets (or subsets) of points in $M$ and $S$ that meet the threshold condition ($t$).
Let $ad$ be the average distance between the two entire feature sets $M$ and $S$.
Let $rd \leftarrow 1000$ be the initial minimum required average distance between two feature sets $M$ and $S$.

**For** $index = 0$ **to** $4$ **do**
  $pIndex \leftarrow random[index]$
  **For** $i = 5$ **to** $10$ **do**
    $aIndex \leftarrow p[pIndex].indices[i]$
    **For** $j = 0$ **to** $slength$ **do**
      $qIndex \leftarrow q[j].pointNumber$
      **For** $k = 5$ **to** $10$ **do**
        $bIndex \leftarrow q[qIndex].indices[k]$
        $s = \dfrac{\sqrt{(p[aIndex].x - p[pIndex].x)^2 + (p[aIndex].y - p[pIndex].y)^2}}{\sqrt{(q[bIndex].x - q[qIndex].x)^2 + (q[bIndex].y - q[qIndex].y)^2}}$
        $anglePA \leftarrow atan2(p[aIndex].y - p[pIndex].y, p[aIndex].x - p[pIndex].x)$
        $angleQB \leftarrow atan2(q[bIndex].y - q[qIndex].y, q[bIndex].x - q[qIndex].x)$
        $\theta \leftarrow anglePA - angleQB$
        $tx \leftarrow p[pIndex].x - q[qIndex].x * s * cos(\theta) + q[qIndex].y * s * sin(\theta)$
        $ty \leftarrow p[pIndex].y - q[qIndex].x * s * sin(\theta) - q[qIndex].y * s * cos(\theta)$
        **If** $(0.94 \leqslant s \leqslant 1.06$ **AND** $\theta \leqslant |30^0|$ **AND** $tx \leqslant |80|$ **AND** $ty \leqslant |87.5|)$
          **For** $h = 0$ **to** $h < 9$ **do**
            **For** $l = 0$ **to** $l < 9$ **do**
              $q[k].x \leftarrow tx + p[h].x * s * cos(\theta) - q[k].y * s * sin(\theta)$
              $q[k].y \leftarrow ty + p[h].x * s * sin(\theta) + q[k].y * s * cos(\theta)$
              Determine $m$
              $l \leftarrow l + 1$
            **End** $l$ **For**
            $h \leftarrow h + 1$
          **End** $h$ **For**
        **End** $factors$ **If**
        **If** $(m > \rho * 10)$
          Calculate $ad$ (using current transformation factors $s, \theta, tx, ty$)
          **If** $(ad < rd)$
            $rd \leftarrow ad$
            Store the current transformation factors $s, \theta, tx, ty$
          **End** $ad$ **If**
        **End** $m$ **If**
        $k \leftarrow k + 1$
      **End** $k$ **For**
      $j \leftarrow j + 1$
    **End** $j$ **For**
    $i \leftarrow i + 1$
  **End** $i$ **For**
  $index \leftarrow index + 1$
  **End** $index$ **For**
**If** $(rd < 1000)$ (a candidate set of local transformation factors has been found)
  apply stored local transformation factors $s, \theta, tx, ty$ to the entire scene feature set
  calculate $m$
  apply global error minimisation to $s, \theta, tx, ty$ (refer Algorithm 5.4 and Table 5.8)
  re-calculate $m$
  output to three separate uniquely named files
**Else**
  no candidate set of transformation factors has been found – output appropriate message
**End** $rd$ **If**

**Algorithm 5.4** Global Error Minimisation

Let $i, j, k, l$ be loop control variables for incrementing successive transformation factor values.
Let $ad$ be the average distance between the two entire feature sets $M$ and $S$.
Let $rd \leftarrow 1000$ be the initial minimum required average distance between two feature sets $M$ and $S$.

**For** $i = (\theta - 5^0)$ **to** $(\theta + 5^0)$ **do**
  rotate all points is **S** by factor $i$
  **For** $j = 0.98$ **to** $1.02$ **do**
    scale all points in $S$ by factor $j$
    **For** $k = (x - 5)$ **to** $(x + 5)$ **do**
      translate the x coordinates of all points in $S$ by factor $k$
      **For** $l = (y - 5)$ **to** $(y + 5)$ **do**
        translate the y coordinates of all points in $S$ by factor $l$
        Calculate $ad$
        **If** $(ad < rd)$
          $rd \leftarrow ad$
          store the current transformation factors $i, j, k, l$
        **End If**
        $l \leftarrow l + 0.25$
      **End** $l$ **For**
      $k \leftarrow k + 0.25$
    **End** $k$ **For**
    $j \leftarrow j + 0.005$
  **End** $j$ **For**
  $i \leftarrow i + 0.25^0$
**End** $i$ **For**
Transform all points in **S** by factors $i, j, k, l$.

By the processes just described, the transformations were conducted for each participants scene feature sets, aligning them with their model feature set. The aligned feature data (as well as the transformation factors and other information) were written to the files specified.

### 5.5.5 Feature Selection

As discussed in section 5.4.4, careful selection of metrics can be adopted to enhance ANN training. In relation to keystroke dynamics, the reason for adopting such steps was because of the variability of data for that biometric characteristic. For fingerprint data, accurately registered features should not exhibit such degrees of variability.

However, the keystroke dynamics data consisted of a consistent number of metrics per sample (for all participants). That is, there were no missing or extra metrics in any sample. This facilitated metrics selection for the keystroke dynamics data, and simplified the process of determining input vectors of consistent length for ANN training and testing.

For fingerprint data processed to this stage (i.e. registration), the same could not be achieved so simply. The number of features extracted from a participant's scans typically vary from sample to sample. Any specific feature may or may not have been detected in any particular sample, due to different finger positioning and pressure and/or the presence of moisture, dirt or other environmental factors.

Also, the number of features extracted from fingerprint scans typically vary from participant to participant. This is because each participant may have a different number of actual features (and a different number of the two types of features). For example, one participant may have 22 minutiae and another may have 57. Even if more than one participant had 22 minutiae, one may have had 17 terminating minutiae and 5 bifurcating minutiae while the other may have had 15 terminating minutiae and 7 bifurcating minutiae.

Taking the above two points into consideration, formulating ANN input vectors becomes a difficult challenge (if consistency of input vector length is to be maintained). Therefore, it was considered desirable that (as with keystroke dynamics) the fingerprint feature information be further processed to provide samples of a consistent length also. So, feature selection was performed to ensure that each sample for each participant had the same number of features. Importantly, this would also be beneficial for the next phase of the experiment (the fusion of keystroke dynamics data and fingerprint feature data).

Before determining how many features should be selected, it was important to determine which local feature attributes to utilise. Typically, the x and y coordinates and the orientation are the attributes used in minutiae matching algorithms (Ratha et al., 1996). With the x and y coordinates considered as 2 separate attributes, this results in 3 attributes.

However, because the primary objective of this experiment was heavily based on feature level fusion of the two data sources, it was considered that the use of all available local feature attributes might be more distinctive (and possibly more beneficial to the pattern recognition process) than just the 3 typically used.

Thus there were 6 available attributes for each local feature (refer section 5.5.3). As far as is known, no other study has incorporated these 6 local feature attributes.

Once the decision was made to utilise all available local feature attributes, it was decided to select 8 features per sample for the experiment. Typically, fingerprint matching for law enforcement (the most stringent type of application) requires a 12 point match to be considered incontrovertible[18]. However, as all 6 attributes per local feature were being utilised (instead of just 3) a quantity of 8 features was considered adequate for this experiment (which is in the context of computer user authentication). Recall that there were 24 metrics per sample in the keystroke dynamics data; it was therefore considered that using more than 8 fingerprint features (which results in a total of 48 metrics—8 features with 6 attributes) could unduly affect findings when the two data were combined in phase 3 of the experiment.

For the same reasons expressed in section 5.4.4, it was determined to also include some global information about each fingerprint. As previously mentioned, the Verifinger Software Development Kit (SDK) 4.2 Visual C++ library allowed for the extraction of only two global features: the ridge count and the minutiae count (refer section 5.5.3). However, it was considered that distinguishing between the two different types of minutiae (that is, ridge termination and ridge bifurcation) may be advantageous. Therefore, a total for the two types of minutiae was also recorded as global data. This resulted in 4 global metrics[19] per sample; added to the 48 local feature attributes, a total of 52 metrics per sample was obtained.

After registration, and deciding upon the metric composition for each sample, the selection of local features was conducted (for each participant). This process was based on the following criteria:

[18] In the USA, a 12 point match is required: in the UK, a 16 point match is required: in France, a 17 point match is required (Hughes and Green, 1991).

[19] The ridge count, the minutiae count, the number of ridge terminations, and the number of ridge bifurcations.

1. Sample Selection: determine the 140 samples with the largest quantity of local features. As indicated in section 5.5.2, data from all scans was recorded in a participants' raw data files. The task here was to select 140 (for each participant) corresponding to their most 'representative' scans.

2. Feature Selection: determine the common local features (i.e. features occurring in all 140 selected samples for a participant). It was identified that samples consisted of varying quantities of common local features, and the quantity of local features common to all samples may have numbered greater or less than the required 8. Wherever possible, the selection of 8 local features common to all 140 samples was desired. If any given sample had less than the 8 required common local features, the 'missing' feature (or features) from that sample had 0 assigned to each attribute.

To achieve point 1, the '.tab' files output by the registration process were utilised. A '.tab' file contained the same number of lines as there were feature sets or samples for a particular participant. The number of fields per sample was the same as the number of features in the model feature set. At the end of the registration process for a scene feature set, the index number of a scene feature point that was successfully matched to a model feature point was written to the file, in a position relative to the index position of the model feature point it was matched to. If no match in the scene feature set was found for a point in the model feature set, a $-1$ was written in a position relative to the index position of the model feature point.

Table 5.9 provides an example of three lines from a '.tab' file for one participant. The first row labels the column numbers. The second row (labeled Model) contains the index numbers for the model feature set, and indicates the order of each point. The remaining rows show the output for three different scene feature sets after registration.

For example, the row labeled Scene 1 at column 8 shows index number 7 of that scene feature set matching with index number 7 of the model feature set. The row labeled Scene 3 at column 8 shows a $-1$, indicating that there was no point in this scene feature set that matched that point in the model feature set. Column 10 shows

that index 9 in the Scene 1 row, index 10 in the Scene 2 row, and index 14 in the Scene 3 row matched to the same feature at index 9 of the model feature set.

It can be seen in Table 5.9, that eleven points in all three scene feature sets matched with their corresponding points (by location, rather than number) in the model feature set, even though some correspondences indicate different point numbers (particularly for Scene 3). These occurred in columns 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13.

| Column | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| Scene 1 | -1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 11 | 12 | 13 | 15 | 16 | 17 | 18 | 19 | 20 |
| Scene 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | -1 | 19 |
| Scene 3 | 4 | 6 | 7 | 9 | 11 | 8 | 10 | -1 | 12 | 14 | 15 | 16 | 17 | -1 | -1 | -1 | -1 | -1 | -1 |

Table 5.9: Example Output From '.tab' File

This data in the '.tab' files were used to rank all feature sets per participant. As each row in the '.tab' file related to a scene feature set, a tally of the number of positive numbers on each row was obtained. The tally for each **ROW** represented the number of matching points in both the scene feature set and the model feature set. The tallies were ranked in descending order such that the feature set with the highest number of matching points was ranked highest. The feature set with the second highest number of matching points was ranked second highest, and so on.

Using the examples provided in Table 5.9, the rows labeled Scene 1 and Scene 2 would be ranked highest with 18 matching points, while Scene 3 would be ranked lowest with 12 matching points. Note that the association between the tallies and their corresponding feature sets (or samples) was maintained during ranking. Once ranked, the top 140 samples were used for the remainder of the experiment; this fulfilled the requirement for the correct number of samples (per participant).

To achieve point 2 (selection of the appropriate 8 common local features from each sample) the data in the '.tab' files were again utilised. Firstly, a tally of the number of positive numbers in each column was obtained. The tally for each **COLUMN** represented the number of feature points present across all 140 samples (selected as described above). The tallies were ranked in descending order such that

the feature that occurred most frequently was ranked highest. The feature that was next highest in frequency was ranked second highest, and so on.

The features that were ranked in the top 8 (according to the column tallies) were the features selected for the experiment. If any of the selected features did not occur in any particular feature set (or sample), the value 0 was assigned for that feature's attributes in that sample only. This occurred in a number of samples for a number of participants. Table 5.10 lists the participants who had a number of missing features in their selected feature sets (or samples). For all other participants, the common 8 features were selected from each of their 140 feature sets (samples). Note that all those participants listed in Table 5.10 had multiply samples with 1 feature missing (as is evident in column 2), and two participants (19 and 80) had multiple samples with 2 features missing (as is evident in column 3).

| PARTICPANT | NUMBER OF SAMPLES WITH 1 FEATURE MISSING | NUMBER OF SAMPLES WITH 2 FEATURES MISSING |
|---|---|---|
| 3 | 4 | – |
| 5 | 29 | – |
| 14 | 10 | – |
| 16 | 4 | – |
| 19 | 43 | 8 |
| 23 | 20 | – |
| 26 | 9 | – |
| 75 | 6 | – |
| 76 | 24 | – |
| 80 | 31 | 2 |
| 81 | 40 | – |
| 90 | 15 | – |

Table 5.10: Participants with Unmatched Features After Selection

Once selection was complete, the metrics were normalised according to the min/max method (Indovina et al., 2003). Normalised metrics were then written to uniquely named files for each participant (according to their participant number). File names were preceded by 'm', with a '.txt' extension (m001.txt, m002.txt, ..., m090.txt). Note these files were written to a different directory than that used to store the registration output.

### 5.5.6 Final Analysis Procedure

There were 90 participants' metric data files (consisting of 140 samples; 52 metrics per sample) for this phase of the experiment. Rather than randomly selecting a new set of training group members for this phase of the experiment, the 50 training group members that were randomly selected in the keystroke dynamics final analysis procedure (refer section 5.4.5) were also allocated to the training group for this phase of the experiment. This was done because participants files were numbered, and data files from the first two phases needed to be paired (i.e. have the same numbered file names) for the data fusion phase. The experiment then proceeded with the training and testing phases.

#### 5.5.6.1 Training Phase

Each training group member had an input file created for training an ANN to recognise their pattern; that is, one ANN per training group member. The selection of samples to make up these files followed a similar process to that used for the keystroke dynamics.

However, there was a slight difference. The process would normally select from all available 140 samples. As explained in the previous section 5.5.5, some samples had 1 or 2 points missing as a result of the registration and selection processes. In these cases, a 0 value was entered for the metrics corresponding to the attributes of the missing points. It was therefore deemed advisable to disallow those samples from random selection for training and validation purposes (though they were still eligible for testing purposes), because they may not be considered a truly 'representative' sample (compared to a sample with no metrics missing).

With this exception in mind, the training file for each training group member was generated as follows:

- 30 samples (of the 140 available) were randomly chosen from that member's data file, for the positive training case (provided a metric value of 0 for the minutia type attribute was not present[20]).

[20]Recall that a value of 1 designated a ridge termination and a value of 2 designated a ridge bifurcation. A value of 0 denotes a missing feature.

- 1 sample was randomly chosen from all other training group members data files, for the negative training case (provided a metric value of 0 for the minutia type attribute was not present). As there were 49 other training group members data files, this meant 49 samples.

As explained in section 5.4.5, positive and negative case training samples are used to help train the ANN to discern the intended pattern.

Each training group member also had a file created for cross validation during the training process. These were generated as follows:

- 10 samples were randomly chosen for that member (provided a metric value of 0 for the minutia type attribute was not present). These samples were chosen from the same data file from which the 30 training samples were chosen (but excluding the training samples).

The reasons for the number of samples for their respective purposes were again explained in section 5.4.5. With 40 samples removed and used for training purposes, this meant that there were 100 remaining samples per participant set aside for testing purposes (refer section 5.5.6.2).

Therefore, each training group members input file (for training) consisted of 79 input samples (30 for the positive training case plus 49 for the negative training case), with 52 metrics per sample. There were 50 such training input files (one for each participant). There were also 50 validation files (consisting of 10 samples, with 52 metrics per sample); one for each training group member, corresponding to each of the training files.

The objective of the training phase was to obtain a registered template associated with each training input file (i.e. for each participant). For the experiment, the back propagation Artificial Neural Network (ANN) architecture was used. When training the ANNs, the 52 metrics (per sample) from the training input file became the input layer nodes for the ANN.

The number of hidden layer nodes was varied from 2 to 26 (inclusive), for each training input file. This was done because there is no standard rule that specifies how many hidden layer nodes should be used, and so the most appropriate ANN

configuration for each member must be determined by trying different configurations and seeing which one performed best. The reason for the upper limit of 26 hidden layer nodes was based on the known accuracy of fingerprint data. After some preliminary trial and error testing, it was considered acceptable to utilise an upper limit of half the number of input layer nodes to reflect this confidence (i.e. 52 input layer nodes divided by 2).

As a result there were 1,250 individual ANNs trained (50 x 25). The ANN training phase for fingerprint data (with the multiple configurations just described) was conducted using a desktop computer with 2 Ghz AMD processor and 512MB RAM. Training took 12.5 days (i.e. approximately 6 hours per participant).

Once training was completed, all ANN configurations were subjected to preliminary assessment to determine the single configuration (for a participant) that returned the least number of false acceptances (Type I errors) and false rejections (Type II errors). This process was performed manually because the determination required assessing the trade-off between the two error types.

Basically, the configuration that returned the least number of Type I errors, whilst returning Type II errors at an acceptable level was selected. The same process was applied for all training group members, and the weights of the ANN configurations (thus selected) were used as registered templates from that point onwards. These were written to a file (such that the participant number and number of hidden layer nodes were indicated in the filename, and given a '.w' extension) and subsequently used during the testing phase.

#### 5.5.6.2 Testing Phase

Each training group member had an input file created for testing their trained ANN. The testing file for each training group member was generated as follows:

- 100 samples (i.e. 140 less those samples used in the training phase) for the member being tested were used for the positive testing case.
- 100 unused samples from all other training group members were used for the negative testing case.

- 140 samples from all non-training group members were used for the negative testing case.

Positive case testing examines whether the ANN correctly recognises samples belonging to the member that it has been trained to recognise (samples it has not seen during training). Non-recognition of any of these 100 positive case samples are instances of Type II errors (false negatives) (refer Chapter 6 section 6.2).

Negative case testing examines whether the ANN correctly rejects samples as belonging to someone other than the member it has been trained to recognise. Recognition of any of these negative case samples are instances of Type I errors (false positives) (refer Chapter 6 section 6.2).

So for each training group member, the samples from each relevant file (in the order listed above) were read and written to their testing input file. Therefore, each training group member had a testing input file consisted of 10,600 input patterns (i.e. 100 for the member being tested, plus 100 for each of the other 49 training group members, plus 140 for each of the 40 non-training group members), with 52 metrics per sample. There were 50 such testing input files (one for each training group member). The results of the testing phase are provided in Chapter 6, and a discussion of these results is presented in Chapter 7.

## 5.6 Feature Level Data Fusion

### 5.6.1 Introduction

As discussed in Chapter 2 sections 2.3.1.2 and 2.3.2.1, the concept of feature level data fusion consists of utilising the features from multiple sources of data. This is done to take advantage of the distinctive features from each source; to enhance the quality and richness of data for improvement in accurate and robust verification. Also discussed was the possible requirement for pre-processing (for example, data alignment) and feature selection. As discussed in sections 5.4.4 and 5.5.4, these requirements have been taken into consideration in the first two phases of this experiment, with a view to facilitating feature level data fusion (the third phase).

However, the appropriate data fusion paradigm is another important issue that needs consideration. There are three possible paradigms to consider when combining multiple sources of data (refer Chapter 2 section 2.3.1.1). These are the competitive, complementary, and cooperative approaches.

The competitive approach involves the individual data sources competing with each other. The data source that best represents an object/entity/identity is utilised for the purpose under consideration; all other data sources are not.

For the current study, this would mean that only the characteristic (keystroke dynamics OR fingerprint recognition) that best represents an identity would be selected and utilised for verification. In terms of biometrics this is basically a uni-modal approach, and because of a superior accuracy rating (refer Table 2.1), fingerprint features would probably be selected over keystroke dynamics in most instances. Even considering two more comparatively equivalent characteristics (such as fingerprint features and iriscode), still only one gets utilised (though the chosen characteristic could vary in this case).

The purpose of this study was the feature level fusion of multiple sources of data for improved accuracy in the verification process. That is, a multi-modal approach which should return more accurate and robust results. Therefore, as the competitive approach did not fit the aim of the this study it was not considered further. Consequently for this phase of the experiment, the other two paradigms (complementary and cooperative) were applied to the processed data from the two previous experimental phases (keystroke dynamics and fingerprint recognition), with the outcomes from the current phase to be compared with the results achieved during the previous phases (refer Chapters 6 and 7).

### 5.6.2 Complementary Data Fusion Approach

The complementary data fusion approach consists of the combination of all available metrics from all sources of data. That is, 100% of the data from all sources is utilised. This is usually achieved by the concatenation of the metrics from those data sources. There is no known advantage to any particular order for the concatenation.

Therefore for this study, the keystroke dynamics data and fingerprint feature data were merged in that order, for no particular reason.

The following section describes the process for the fusion of keystroke dynamic metrics and fingerprint feature metrics using the complementary approach.

#### 5.6.2.1 Complementary Fusion of Keystroke Dynamics and Fingerprint Feature Data

For the fusion process, there were two conceivable ways to create the data files:

1. Utilising the metrics files that were obtained for the individual biometric characteristics (from the previous two phases). The determination of the metrics files for all participants was described in sections 5.4.4 and 5.5.4. The fusion process would involve the concatenation of the keystroke dynamic metrics and the fingerprint feature metrics for each sample for each participant. This would be followed by the selection of samples for creating the training, testing and validation files by the same processes described in sections 5.4.5 and 5.5.6 for the creation of such files for the previous two phases of the experiment.

2. However, a simpler method was to utilise the training, testing and validation files that had already been created for ANN analysis in the previous two phases. These files had already undergone the selection of samples and metrics, and therefore fusion became merely a matter of concatenating the keystroke dynamic metrics and the fingerprint feature metrics for each sample for each training group member from the appropriate training, testing and validation files. This method was adopted for the current experiment as it greatly simplified the fusion process.

Algorithm 5.5 demonstrates the process that was applied to all samples in the training, testing and validation files (from the previous two phases), for each training group member. Note that in Algorithm 5.5, the training files had $samples = 79$, the testing files had $samples = 10,600$, and the validation files had $samples = 10$.

In Algorithm 5.5, Steps 2—5 demonstrate that a row of keystroke dynamic metrics was copied first into a row of $dfMatrix$ (the two dimensional array for storing

the combined metrics). Then Steps 6—9 demonstrate that a row of fingerprint feature metrics was appended to the same row of $dfMatrix$.

**Algorithm 5.5** Complementary Data Fusion

Let *samples* be the number of samples in each data file.
Let $kdMatrix$ be the two dimensional array containing the keystroke dynamic metrics.
Let $kdLength$ be the number of keystroke dynamic metrics in each sample.
Let $fpMatrix$ be the two dimensional array containing the fingerprint feature metrics.
Let $fpLength$ be the number of fingerprint feature metrics in each sample.
Let $dfMatrix$ be the two dimensional array for storing the combined metrics.
Let $dfLength \leftarrow kdLength + fpLength$, the number of combined metrics in each sample.
Let $i$ be the loop control variable for accessing the rows in all matrices.
Let $j$ be the loop control variable for accessing columns in all matrices.

1: **For** $i = 0$ **to** *samples* **do**
2: **For** $j = 0$ **to** $kdLength$ **do**
3: $dfMatrix[i][j] \leftarrow kdMatrix[i][j]$
4: $j \leftarrow j + 1$
5: **End** $j$ **For**
6: **For** $j = kdLength$ **to** $dfLength$ **do**
7: $dfMatrix[i][j] \leftarrow fpMatrix[i][j - kdLength]$
8: $j \leftarrow j + 1$
9: **End** $j$ **For**
10: $i \leftarrow i + 1$
11: **End** $i$ **For**

In order to access the correct column numbers of $dfMatrix$ (after the keystroke dynamic metrics have been copied into it), Step 6 has $j = kdLength$, the number of keystroke dynamic metrics. This meant that in Step 7, $kdlength$ required subtraction from $j$ in order to access the correct column of $fpMatrix$ (the fingerprint feature matrix).

As a result of the fusion process, there were training, testing and validation files created (for the same 50 training group members determined in the previous two phases of the experiment), with each sample in the files consisting of 76 metrics (24 keystroke dynamic metrics and 52 fingerprint feature metrics). The relative number of metrics from the two sources reflects the greater validation power of fingerprint recognition compared with keystroke dynamics, as discussed in Chapter 2 section 2.2.4. The experiment then proceeded with the training and testing phases.

#### 5.6.2.2 Final Analysis Procedure

The training and testing phases for the final analysis (adopting the complementary data fusion approach) were conducted in the same manner as the phases described in sections 5.4.5 and 5.5.6. So for each of the 50 training group members, the following files were utilised:

- The training files consisted of 79 samples (30 for the positive training case and 49 for the negative training case, with 76 metrics per sample).
- The testing files consisted of 10,600 samples (100 for the member being tested, plus 100 for the other 49 training group members, plus 140 for the 40 non-training group members, with 76 metrics per sample).
- The validations files consisted of 10 samples (with 76 metrics per sample).

When training the ANNs, the 76 metrics (per sample) from a training input file became the input layer nodes for the ANNs. The number of hidden layer nodes was varied from 2 to 26 (inclusive), for each training input file. This was done so the most appropriate ANN configuration for each member could be determined by trying different configurations and seeing which one performed best.

The reason for the upper limit of 26 hidden layer nodes was because of the accepted knowledge that fingerprint recognition is a more accurate biometric characteristic than keystroke dynamics. It was therefore deemed appropriate to utilise the same upper limit, as was applied to the fingerprint recognition training, for training the combined data.

As a result there were 1,250 individual ANNs trained (50 x 25). The ANN training phase for the complementary fused data (with the multiple configurations just described) was conducted using a desktop computer with 2 Ghz AMD processor and 512 MB RAM. Training took 14 days (i.e. approximately 6.7 hours per participant).

Once training was completed, all ANN configurations were assessed to determine the single configuration (for a participant) that returned the least number of false acceptances (Type I errors) and false rejections (Type II errors). This process was performed manually because the determination required assessing the trade-off

between the two error types. The configuration that returned the least number of Type I errors, whilst returning Type II errors at an acceptable level was selected. The same process was applied for all training group members, and the weights of the ANN configurations (thus selected) were used as registered templates from that point onwards. These were written to file (such that the participant number and number of hidden layer nodes were indicated in the file name, and given a '.w' extension) and subsequently used during the testing phase.

When testing the ANNs, the 76 metrics (per sample) from a testing input files became the input layer nodes for the ANNs. Each training group member had a testing input file consisting of 10,600 input patterns (i.e. 100 for the member being tested, plus 100 for each of the other 49 training group members, plus 140 for each of the 40 non-training group members), with 76 metrics per sample. There were 50 such testing input files (one for each training group member). The stored weights (from the training phase) were applied to the ANNs, thus resulting in the correct configuration[21].

Positive case testing examines whether the ANN has correctly recognised samples belonging to the member that it has been trained to recognise (samples it has not seen during training). Non-recognition of any of these 100 positive case samples are instances of Type II errors (false negatives) (refer Chapter 6 section 6.2).

Negative case testing examines whether the ANN has correctly rejected samples belonging to someone other than the member it has been trained to recognise. Recognition of any of these negative case samples are instances of Type I errors (false positives) (refer Chapter 6 section 6.2).

The results of the testing phase are provided in Chapter 6, and a discussion of these results is presented in Chapter 7.

### 5.6.3 Cooperative Data Fusion Approach

Unlike the complementary data fusion approach, where 100% of available metrics from all data sources are included in the fusion process, the cooperative data fusion

[21]Note that the stored weights from this phase of the experiment were also utilised in the cooperative data fusion metrics selection process (refer section 5.6.3.1).

approach utilises the 'best' features from each data source. That is, cooperative data fusion is based on the assumption that only features (from each source) that best characterise an object/entity/identity are combined to enhance recognition. This has the advantage of reducing the size of data sets (by discarding unnecessary features), which decreases the amount of processing (and time) required to perform the recognition task (Dash and Liu, 1997).

Cooperative data fusion then, necessarily involves a process of feature selection. The aim of feature selection is to determine those features (from all sources) that allow for the deduction of structures or patterns within the data, which are most likely to achieve the intended goal (John et al., 1994).

Like most real-world classification tasks, recognition requires the use of supervised learning (Dash and Liu, 1997). As described in Chapter 2 section 2.4.2, supervised learning involves providing a training set of data (representing instances of the class to be learned) to a learning algorithm. It also requires providing 'target' outputs for each sample in the training set. Thus targets provide a goal for the learning process, by providing positive reinforcement for the training set samples.

In order to effectively learn from the training set the algorithm attempts to deduce structures that correctly classify a large enough subset of the entire training set, yet not so large as to overfit the data and thus become too specific (or less generalisable) (John et al., 1994). That is, the algorithm should use only the subset of features that leads to the best performance.

This task involves determining the specific features (within the training set) that result in optimum prediction accuracy, and is known as the feature subset selection problem. Typically, selection is based on the relevance of features to the accuracy of prediction. That is, if certain features are more relevant than others, they are considered the 'most representative' features for accurate prediction. So the question is, what constitutes relevance?

John et al. (1994) conclude that though relevance (in terms of feature subset selection) may seem a trivial matter, there are varying degrees of relevance. They define relevance and irrelevance as follows (John et al., 1994):

- Relevance refers to a feature's contribution to prediction accuracy. To quantify relevance, two levels of contribution can be considered:

    1. strongly relevant - the feature is essential to prediction accuracy. If it is removed from the training set, loss of prediction accuracy will definitely occur.

    2. weakly relevant - the feature sometimes contributes to prediction accuracy or contributes to varying degrees. If it is removed from the training set, loss of prediction accuracy may or may not occur; the impact on accuracy will depend upon the degree to which the feature is relevant.

- Irrelevance indicates that the feature does not contribute to prediction accuracy at all. If it is removed from the training set, no loss of prediction accuracy will occur as the feature is superfluous.

Obviously, any feature subset selection process would attempt to determine relevant features and discard irrelevant features. The application to which cooperative data fusion is to be applied will determine the accepted degree of relevance. For example, mission critical applications would insist on the inclusion of only strongly relevant features (i.e. excluding irrelevant and weakly relevant features), thus imposing very stringent restrictions to minimise false classification. Less critical applications may accept weakly relevant features (as well as strongly relevant features), at the possible expense of an increase in false classification.

Based on this understanding of relevance, there are of a number of issues to be resolved when deciding upon the feature subset selection criteria for any application:

1. What percentage of the available data (from all sources) should be utilised in the selection process? For example, if features are selected from the combined data used in a complementary data fusion approach (remembering that this constitutes 100% of the available data), what percentage of that data would best serve for cooperative data fusion? For example, 50% of the combined data? OR 75%? OR 90%?

2. What proportion of the chosen percentage should each of the individual data sources contribute to the newly created fused data set, and how should these proportions be determined? For example, given two data sources, if the features from one data source are more accurate for verification purposes than the features of the other data source, should the more accurate data source constitute a larger proportion (of features) in the fused data set? If so, what proportion (of the nominate percentage) should it contribute?

3. What method or criteria should be used to determine the relevance of features (and consequently their selection)? Will a different feature selection method be required for each data source? Note that the selection method may need to be decided upon before the percentages and proportions can be determined. Conversely, the required percentages and proportions may impact on the choice of the selection method.

In relation to question 1, depending on the determined proportions of features from each data source and the selection method, it is entirely possible that certain percentages will not be practicable. The percentages chosen for this phase of the current experiment, and the reasons why some percentages were not practicable, are discussed in the next section 5.6.3.1.

In relation to question 2, depending on the accuracy (for verification purposes) of the data sources and the selection method, an uneven proportion of metrics from the different data sources may be required.

For example in the current study, the keystroke dynamics data consisted of 24 metrics and the fingerprint feature data consisted of 52 metrics. The first two phases of the study returned much more accurate results for the fingerprint feature data than for the keystroke dynamics data (refer to Chapters 6 and 7). Given that fingerprint features were more accurate than keystroke dynamics, it was inevitable that fingerprint feature data would constitute a much larger proportion of each sample in the combined data sets. Again, the calculation of the proportions used in this phase of the experiment is discussed in the next section 5.6.3.1.

The following section describes the method upon which feature selection was based for the current study (in answer to question 3) for the cooperative data fusion of keystroke dynamics data and fingerprint feature data at the feature level[22].

#### 5.6.3.1 Selection of Keystroke Dynamics and Fingerprint Feature Metrics

Firstly, because percentages and proportions may be affected by the selection criteria, the determination of a selection method was required before the percentages and proportions could be quantified. That is, the answer to question 3 (above) required answering before the answers to questions 1 and 2 could be determined.

Consequently, the following discussion describes the method of feature selection and then the determination of the respective percentages and proportions. It then goes on to explain the processes developed to achieve the cooperative data fusion.

The data files from which features (from both sources) were extracted were copies of those created for the complementary data fusion experiment. That is, the files obtained as described in section 5.6.2.1. As explained in that section, the complementary data fusion process resulted in training, testing and validation files created for the 50 training group members with the training files containing 79 samples, the testing files containing 10,600 samples, and the validation files containing 10 samples; each sample in the files consisted of 76 metrics.

Also used from the complementary data fusion experiment were the weight files, that were recorded at the completion of ANN training, for that phase of the experiment; these files were used for feature selection in this phase of the experiment.

An ANN can be trained (via an iterative supervised learning algorithm) to recognise patterns within training data sets. During the iterative processing, the importance or relevance of each input node (or feature) is gradually learned by the ANN[23].

[22] As noted in Chapter 8 section 8.4, other feature selection methods (than those used in the current experiment) may be equally applicable to the task of feature selection, and may perform as well or better. However, the method chosen was for convenience because trained ANN weights were readily available as a result of phases 1 and 2 of the experiment. Also, evaluating other feature selection methods was beyond the scope of this study.

[23] In certain ANNs, this is facilitated by back propagating error values (calculated at the hidden and output layer nodes) through the network and subsequently updating the connecting weight values between input, hidden and output layer nodes.

The relevance of an input node (or feature) becomes apparent by observing and assessing the weight values between it and the hidden layer nodes that it is connected to (Schuschel and Hsu, 1998). Typically, input nodes that become more important (or relevant) result in an increase to the associated weight values. The value of the associated weights corresponding to irrelevant input nodes tend to diminish or decrease. So by determining the most relevant nodes during the training process, ANNs actually perform feature subset selection because they start by processing all features and gradually determine the most relevant.

However, an ANN learning algorithm could be considered an inefficient method for feature subset selection, because initially the training process requires processing the entire feature space[24]. Despite the ANNs inefficiency as a complete feature subset selection solution, the weight values from previously trained ANNs (if available) can be an effective method of determining the most relevant nodes (or features) and thus be utilised for feature subset selection (Schuschel and Hsu, 1998).

According to John, et al., (1994), there are two primary approaches to feature subset selection:

1. The filter model: where features are selected independently from the learning/induction algorithm. As demonstrated in Figure 5.6, features are filtered as part of a pre-processing step, prior to them being supplied to the learning/induction algorithm.

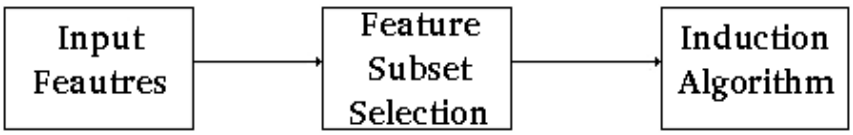


Figure 5.6: Filter Model

2. The wrapper model: where the learning/induction algorithm becomes a part of the selection process. As demonstrated in Figure 5.7, this model involves searching the feature space, evaluating feature subsets, and applying the subsets to the learning/induction algorithm. By iteratively processing these steps, the 'best' subset is found. Note that the learning/induction algorithm directly guides the selection process.

[24] Other intelligent methods of feature subset selection have been well reviewed by Dash and Liu (1997).

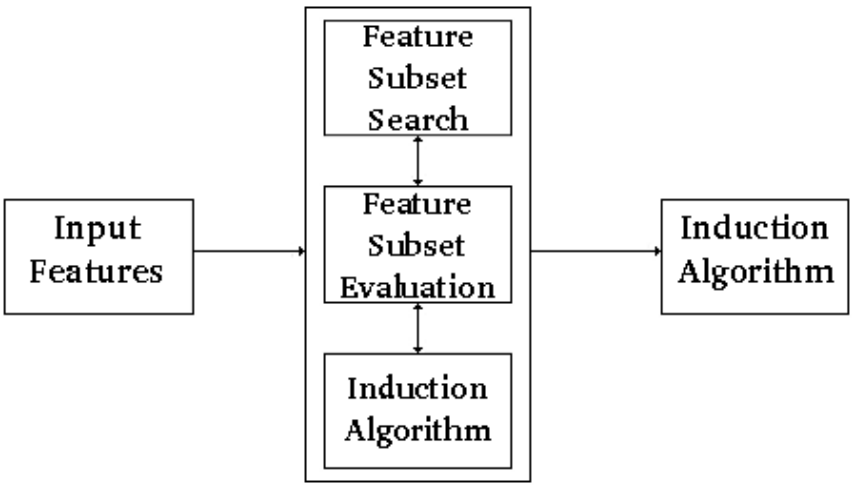


Figure 5.7: Wrapper Model

Given that previously trained ANN weight values were available for this phase of the experiment (from the complementary data fusion phase of the experiment), it was considered inappropriate to utilise the wrapper model for this study as that would require further unnecessary iterative processing. That is, as the previously trained ANNs had already determined the most relevant features, there was no need to further process these features using the wrapper methodology. Therefore, the selection of features for the cooperative phase of the experiment, was achieved utilising the filter model. That is, by selecting a subset of features based on the weight values from the already trained ANNs.

Schuschel and Hsu (1998) suggested the use of a subset selection method that accumulates the weights (connected to an input layer node) from a trained ANN. This accumulated value was termed the Approximate Relative Local Gain (ARLG). An ARLG is calculated for each input layer node, by accumulating all the weights connected to it. Once calculated, the ARLGs can be ordered by magnitude, and the nodes associated with the largest magnitude should relate to the most relevant features. The nodes associated with the smallest magnitude should relate to the least relevant features. This information can then be utilised to determine the most appropriate proportion of features from each biometric characteristic, and to select the most relevant feature subsets.

Schuschel and Hsu (1998), described the derivation of the ARLG, and provided Equation 5.11 for its calculation.

$$LG_{ik} = \sum_{j} |W_{ij} \times W_{jk}| \tag{5.11}$$

where $LG_{ik}$ represents the ARLG for each input layer node, $W_{ij}$ are the weight values connecting the current $i^{th}$ input layer node and its associated hidden layer nodes, and $W_{jk}$ are the weight values connecting the current $j^{th}$ hidden layer node and its associated output layer nodes.

Algorithm 5.6 shows the implementation of Equation 5.11 for the calculation of the ARLGs, utilising the ANN weight value files from the training phase of the complementary data fusion experiment. Note that when implemented, the value of $k$ from Equation 5.11 was set to 1. This was because all ANNs in the current study had only 1 output layer node.

The result of the calculations described in Algorithm 5.6 was a matrix where each row contained all ARLG values for a sample, with each column containing the individual ARLG values (corresponding to each input node) in that sample. Note that the column index number corresponding to each input layer node was also stored in a separate matrix (*Indices*) for the subsequent selection process.

Each sample's ARLGs (i.e. column values) were then sorted into descending order, whilst maintaining the correspondence between the ARLGs and the original indices (feature positions). This was important for the selection process. Once the ARLGs were ordered, the association between the highest values and their corresponding indices/positions meant that the correct metrics could be accessed when and if selected.

As an example, Table 5.11 shows the 24 keystroke dynamics ARLGs for 3 samples for participant 1. In Table 5.11, the ordered local gains (to 2 decimal places) and their corresponding index positions are shown next to each other for the 3 samples. It can be seen that each sample's ARLGs decrease in magnitude as row numbers increase, and that no index number is repeated in any of the associated columns.

**Algorithm 5.6** Approximate Relative Local Gain

Let *samples* be the number of samples in the training files corresponding to each weight file.
Let *inputToMiddle* be a matrix containing the weights between each of the input layer nodes (rows) and all hidden layer nodes (columns).
Let *ilneurons* be the number of rows (i.e. the number of input layer nodes) in *inputToMiddle*.
Let *mlneurons* be the number of columns (i.e. the number of hidden layer nodes) in *inputToMiddle*.
Let *middleToOutput* be an array containing the weights between each of the hidden layer nodes and the single output layer node.
Let *ARLG* be a matrix with *samples* number of rows and *ilneurons* number of columns, for storing the accumulated relative local gains.
Let *Indices* be a matrix with *samples* number of rows and *ilneurons* number of columns, for storing the indices corresponding to the ARLGs.
Let $s$ be the loop control variable for accessing each sample.
Let $i$ be the loop control variable for accessing the rows in *inputToMiddle*.
Let $j$ be the loop control variable for accessing the columns in *inputToMiddle* and the elements in *middleToOutput*.

1: **For** $s \leftarrow 0$ **to** *samples* **do**
2: **For** $i \leftarrow 0$ **to** *ilneurons* **do**
3: **For** $j \leftarrow 0$ **to** *mlneurons* **do**
4: $ARLG[s][i] \leftarrow ARLG[s][i] + |inputToMiddle[i][j] * middleToOutput[j]|$
5: $j \leftarrow j + 1$
6: **End j For**
7: $Indices[s][i] \leftarrow i$
8: $i \leftarrow i + 1$
9: **End i For**
10: $s \leftarrow s + 1$
11: **End s For**

| Sample 1 Local Gain | Sample 1 Indices | Sample 2 Local Gain | Sample 2 Indices | Sample 3 Local Gain | Sample 3 Indices |
|---|---|---|---|---|---|
| 41.25 | 2 | 54.38 | 18 | 10.94 | 9 |
| 35.35 | 4 | 49.63 | 15 | 9.93 | 7 |
| 23.32 | 3 | 42.95 | 3 | 8.46 | 12 |
| 20.47 | 7 | 41.39 | 0 | 8.39 | 15 |
| 18.38 | 17 | 38.79 | 23 | 6.52 | 19 |
| 17.52 | 23 | 35.21 | 21 | 5.93 | 0 |
| 17.16 | 20 | 34.43 | 14 | 4.61 | 23 |
| 16.27 | 18 | 30.49 | 17 | 4.24 | 17 |
| 15.89 | 19 | 30.14 | 16 | 3.95 | 6 |
| 15.89 | 5 | 29.76 | 4 | 3.86 | 14 |
| 15.58 | 11 | 29.27 | 20 | 3.85 | 3 |
| 15.32 | 8 | 26.16 | 2 | 3.72 | 18 |
| 14.31 | 14 | 23.91 | 5 | 3.58 | 16 |
| 13.99 | 12 | 21.69 | 22 | 3.54 | 21 |
| 10.93 | 13 | 20.85 | 11 | 3.11 | 8 |
| 10.82 | 16 | 19.87 | 6 | 2.34 | 2 |
| 10.25 | 10 | 19.62 | 10 | 2.27 | 11 |
| 10.05 | 6 | 18.01 | 7 | 2.07 | 4 |
| 10.01 | 15 | 16.54 | 8 | 1.95 | 1 |
| 9.28 | 21 | 11.92 | 1 | 1.80 | 13 |
| 8.49 | 0 | 11.90 | 19 | 1.69 | 5 |
| 7.96 | 1 | 11.81 | 13 | 1.63 | 10 |
| 5.93 | 9 | 6.95 | 9 | 1.32 | 22 |
| 4.89 | 22 | 5.57 | 12 | 0.93 | 20 |

Table 5.11: Approximate Relative Local Gain for Keystroke Dynamics

| Participant | Keystroke Dynamic Average ARLG | Fingerprint Feature Average ARLG | Proportionate Ratio |
|---|---|---|---|
| 1 | 15.3879 | 0.9592 | 16.0425 |
| 2 | 13.8977 | 1.2843 | 10.8215 |
| 3 | 26.3020 | 0.8618 | 30.5204 |
| 5 | 18.4693 | 0.6466 | 28.5647 |
| 7 | 4.1945 | 1.0317 | 4.06573 |
| 9 | 13.0952 | 0.7669 | 17.0733 |
| 12 | 23.9413 | 0.8560 | 27.9685 |
| 14 | 7.0749 | 0.8416 | 8.4066 |
| 16 | 5.5218 | 0.9826 | 5.6195 |
| 18 | 4.9053 | 0.9359 | 5.2411 |
| 20 | 8.2078 | 0.7025 | 11.6832 |
| 21 | 7.3238 | 0.7847 | 9.3332 |
| 23 | 10.5154 | 1.4024 | 7.4982 |
| 24 | 10.5546 | 0.9010 | 11.7137 |
| 25 | 9.9607 | 0.6893 | 14.4502 |
| 27 | 9.4348 | 0.8471 | 11.1373 |
| 29 | 20.0864 | 0.8971 | 22.3898 |
| 32 | 4.9117 | 0.8783 | 5.5919 |
| 34 | 9.7762 | 0.7272 | 13.4433 |
| 36 | 10.1369 | 0.8042 | 12.6048 |
| 38 | 5.4568 | 0.5514 | 9.8959 |
| 40 | 4.1378 | 0.7905 | 5.2347 |
| 41 | 11.4390 | 0.9329 | 12.2614 |
| 43 | 14.7119 | 0.6953 | 21.1603 |
| 45 | 6.5489 | 0.8869 | 7.3842 |
| 46 | 8.1231 | 0.8104 | 10.0235 |
| 47 | 17.4786 | 0.8763 | 19.9458 |
| 49 | 17.6859 | 0.6785 | 26.0673 |
| 52 | 3.5083 | 0.5685 | 6.1709 |
| 54 | 10.5214 | 0.8015 | 13.1274 |
| 56 | 4.9203 | 0.9306 | 5.2870 |
| 58 | 26.5361 | 1.2193 | 21.7634 |
| 60 | 8.7201 | 0.7571 | 11.5184 |
| 61 | 15.7636 | 0.7455 | 21.1447 |
| 63 | 15.9693 | 0.8857 | 18.0309 |
| 65 | 12.0272 | 1.0177 | 11.8185 |
| 67 | 16.1491 | 0.9436 | 17.1136 |
| 68 | 8.5989 | 0.8016 | 10.7275 |
| 69 | 11.6031 | 0.7239 | 16.0278 |
| 72 | 10.0226 | 0.9169 | 10.9309 |
| 74 | 17.3575 | 0.9463 | 18.3430 |
| 76 | 8.6560 | 1.4955 | 5.7880 |
| 78 | 9.7989 | 1.1174 | 8.7695 |
| 80 | 14.6645 | 0.8547 | 17.1575 |
| 81 | 21.6240 | 0.5763 | 37.5204 |
| 83 | 11.4662 | 0.7021 | 16.3304 |
| 85 | 3.7297 | 0.7141 | 5.2227 |
| 87 | 17.98439 | 0.9687 | 18.5651 |
| 89 | 14.2733 | 0.8463 | 16.8659 |
| 90 | 11.9442 | 0.8029 | 14.8763 |
| Average | | | 14.1848 |

Table 5.12: Average Local Gain Proportions

After the calculation and sorting of the ARLGs, a method was required to determine how many samples from each characteristic to utilise. It was decided to determine an average ARLG for each of the two biometric characteristics for each participant (over all of their samples). From the two averages, a proportionate ratio was established for each participant, by dividing the average ARLG for fingerprint features into the average ARLG for keystroke dynamics. Table 5.12 shows the average ARLGs for keystroke dynamics (column 2) and fingerprint features (column 3) and the resultant proportionate ratios (column 4).

The proportionate ratio for each participant was then used to determine the number of keystroke dynamic metrics to select (from the available metrics), across all samples for a given percentage. The remainder of the required number of metrics (to make up the given percentage) were selected from the fingerprint feature metrics.

It was discovered at this point that the calculated proportionate ratios affected the range of percentages that could be tested during the experiment (refer question 1 in section 5.6.3). As an example, 80% of the 76 metrics (from the complementary data fusion experiment) is 61 metrics (rounded to the nearest whole number). If the selection process indicated that 7 keystroke dynamic metrics were considered relevant, this means that 54 fingerprint feature metrics would be required to make up the remaining metrics to give the total of 61. As there were only 52 fingerprint metrics available, this would not be possible.

In fact, percentages above 70% and below 40% were not practicable because once the proportion levels were determined for most participants, there were either not enough fingerprint feature metrics available to make their quota or the keystroke dynamic metrics were not represented at all. It was therefore decided to conduct the cooperative data fusion experiment for 4 different percentages—40%, 50%, 60%, and 70%—of the 100% available. Only four percentages were used because this phase of the experiment was primarily to demonstrate some preliminary results that may be achieved using cooperative data fusion. Percentages rounded to 10 were utilised for ease of calculation. The required number of metrics (rounded to the nearest whole number) for these percentages (of the 76 available metrics) are shown in Table 5.13.

| Percentage of Available Metrics | Required Number of Metrics |
|---|---|
| 40% | 30 |
| 50% | 38 |
| 60% | 46 |
| 70% | 53 |

Table 5.13: Number of Metrics Per Percentage

The following worked example demonstrates the determination of the number of metrics that were selected from both data sources for participant 1. Firstly, Table 5.13 indicates that the required number of metrics for 50% (of the 76 available metrics) is 38 metrics. Secondly from Table 5.12, the proportionate ratio for participant 1 is 16.0425. Dividing 16.0425 into 38, returns a result of 2.3687. Rounding this value to the nearest whole number, indicates that 2 keystroke dynamic metrics should be used; thus 36 fingerprint feature metrics would be needed to make up the required number of metrics.

Algorithm 5.7 shows the implementation used to determine the number of metrics to utilise from both data sources. The quantities were used across all samples for each participant, for a given percentage. The same process was applied to all the percentages used in the experiment (refer next section 5.6.3.2).

**Algorithm 5.7** Proportion Calculation For Cooperative Data Fusion

Let *participants* be the number of training group members.
Let *metrics* be the required number of metrics for the particular percentage of available metrics (refer Table 5.13).
Let *proportions* be an array containing the proportions for each participant (from the last column of Table 5.12).
Let *kdRatios* be an array for storing the number of keystroke dynamic metrics to use (calculated for each participant).
Let $fpRatios$ be an array for storing the number of fingerprint feature metrics to use (calculated for each participant).

1: **For** $i \leftarrow 0$ **to** *participants* **do**
2: $kdRatios[i] \leftarrow metrics/proportions[i]$ (rounded to the nearest whole number)
3: $fpRatios[i] \leftarrow metrics - kdRatios[i]$
4: $i \leftarrow i + 1$
5: **End i For**

Once the number of metrics per characteristic were determined (according to Algorithm 5.7), the selection process required the storage of the index numbers corresponding to the required number of ordered keystroke dynamic ARLGs and fingerprint feature ARLGs.

So, if 2 keystroke dynamic metrics and 36 fingerprint feature metrics were required, the index numbers—corresponding to the 2 largest keystroke dynamic ARLGs and the 36 largest fingerprint feature ARLGs—were stored so that the appropriate metrics could be extracted and fused (refer next section 5.6.3.2).

Algorithm 5.8 shows the method for accessing and storing the appropriate index numbers corresponding to the ARLGs chosen by magnitude.

---

**Algorithm 5.8** Index Determination For Cooperative Data Fusion

---

Let *participants* be the number of training group members.
Let *samples* the number of samples in the training files.
Let *kdRatios* be an array containing the number of keystroke dynamic metrics to use, calculated for each participant according to Algorithm 5.7.
Let $fpRatios$ be an array containing the number of fingerprint feature metrics to use, calculated for each participant according to Algorithm 5.7.
Let *kdIndices* be a matrix containing the index numbers (corresponding to the ordered keystroke dynamics ARLGs calculated according to Algorithm 5.6).
Let $fpIndices$ be a matrix containing the index numbers (corresponding to the ordered fingerprint feature ARLGs calculated according to Algorithm 5.6).
Let *kdNeurons* be a matrix for storing the indices (of the selected ARLGs) used to access the corresponding keystroke dynamics metrics.
Let $fpNeurons$ be a matrix for storing the indices (of the selected ARLGs) used to access the corresponding fingerprint feature metrics.
Let $i, j, k$ be the loop control variables for accessing arrays and matrices.

1: **For** $i \leftarrow 0$ **to** *participants* **do**
2: **For** $j \leftarrow 0$ **to** *samples* **do**
3: **For** $k \leftarrow 0$ **to** $kdRatios[i]$ **do**
4: $kdNeurons[j][k] \leftarrow kdIndices[j][k]$
5: $k \leftarrow k + 1$
6: **End k For**
7: **For** $k = 0$ **to** $fpRatios[i]$ **do**
8: $fpNeurons[j][k] \leftarrow fpIndices[j][k]$
9: $k \leftarrow k + 1$
10: **End k For**
11: $j \leftarrow j + 1$
12: **End j For**
13: $i \leftarrow i + 1$
14: **End i For**

---

As an example, using the just mentioned proportions and the data from Table 5.11, the two selected keystroke dynamic metrics for Sample 1 would be those at index 2 and index 4 of the array storing the metrics; for Sample 2 the selected keystroke dynamic metrics would be those at index 18 and index 15; for Sample 3 the selected keystroke dynamic metrics would be those at index 9 and index 7. In each case, the remaining 36 metrics would be selected from the fingerprint feature metrics in the same manner.

The next section 5.6.3.2 describes the extraction of actual metrics and the fusion of those metrics.

#### 5.6.3.2 Cooperative Fusion of Keystroke Dynamics and Fingerprint Feature Data

Once the proportions of each data set and the particular indices of the selected features (from both data sources) had been determined, extraction of the appropriate metrics from the combined metrics files proceeded, utilising the training, testing and validation data files from the complementary phase of the experiment.

The fusion process was achieved in the same manner as that used for the complementary data fusion phase. That is, simple concatenation. However, unlike the complementary data fusion process (where 100% of all available metrics were combined), the cooperative data fusion process combined the appropriate proportion of metrics (and specifically the selected metrics) according to the 4 different percentages previously mentioned (i.e. 40%, 50%, 60%, and 70%).

Algorithm 5.9 shows the extraction and fusion method implemented for this phase of the experiment. Note that Steps 3 to 7 in Algorithm 5.9, copy the keystroke dynamic metrics into each row of $dfMatrix$ (the two dimensional array for the fused data). Step 4 ($kdIndex \leftarrow kdNeurons[i][j]$) accesses the correct column number to copy and assigns it to $kdIndex$ (which was then used to access the actual metric in Step 5). Steps 10 to 15 copy the fingerprint feature metrics into the same row of $dfMatrix$. Again, Step 11 ($fpIndex \leftarrow fpNeurons[i][counter] + 24$) accesses the correct column number to copy and assigns it to $fpIndex$ (which was then used to access the actual metric in Step 12).

**Algorithm 5.9** Feature Extraction and Cooperative Data Fusion Method

Let *participants* be the number of training group members.
Let *samples* be the number of samples in each data file (79 for training files, 10600 for testing files, and 10 for validation files).
Let *Metrics* be the matrix for storing the training, testing, or validation metrics read from the appropriate data files (containing 76 metrics per sample).
Let *kdRatios* be an array containing the number of keystroke dynamic metrics to use, calculated for each participant according to Algorithm 5.7.
Let *fpRatios* be an array containing the number of fingerprint feature metrics to use, calculated for each participant according to Algorithm 5.7.
Let *kdNeurons* be the matrix containing the indices corresponding to the selected keystroke dynamic metrics (as determined by Algorithm 5.8).
Let *fpNeurons* be the matrix containing the indices corresponding to the selected fingerprint feature metrics (as determined by Algorithm 5.8).
Let *dfMatrix* be the matrix for storing the combined metrics.
Let *dfLength* the number of combined metrics for each sample.
Let $i, j, p$ be the loop control variables for accessing the rows and columns in all matrices.
Let *kdIndex*, *fpIndex*, *counter* be variable for accessing specific column indices.

1: **For** $p \leftarrow 0$ **to** *participants* **do**
2: **For** $i \leftarrow 0$ **to** *samples* **do**
3: **For** $j \leftarrow 0$ **to** $kdRatios[p]$ **do**
4: $kdIndex \leftarrow kdNeurons[i][j]$
5: $dfMatrix[i][j] \leftarrow Metrics[i][kdIndex]$
6: $j \leftarrow j + 1$
7: **End j For**
8: $counter \leftarrow 0$
9: $dfLength \leftarrow kdRatios[p] + fpRatios[p]$
10: **For** $j \leftarrow kdRatios[p]$ **to** *dfLength* **do**
11: $fpIndex \leftarrow fpNeurons[i][counter] + 24$
12: $dfMatrix[i][j] \leftarrow Metrics[i][fpIndex]$
13: $counter \leftarrow counter + 1$
14: $j \leftarrow j + 1$
15: **End** $j$ **For**
16: $i \leftarrow i + 1$
17: **End** $i$ **For**
18: $p \leftarrow p + 1$
19: **End p For**

Note that in Step 11, *counter* was used to access the correct column number of *fpNeurons* because $j$ must keep incrementing to access the correct column of *dfMatrix*. Also note that 24 was added to $fpNeurons[i][counter]$ so that the correct column in the *Metrics* matrix was accessed (remembering that the data files from the complementary phase have 24 keystroke dynamic metrics first, followed by the 52 fingerprint feature metrics per sample).

#### 5.6.3.3 Final Analysis Procedure

The training and testing phases for the final analysis (adopting the cooperative data fusion approach) were conducted in the same manner as the complementary data fusion approach, described in section 5.6.2.2.

However unlike the procedure used for the complementary approach, the procedure for the cooperative approach required 4 stages; one stage for each of the four percentages (40%, 50%, 60%, 70%) of the complementary data. This meant creating training, testing, and validation files (with the correct proportion of metrics) for each of those four stages.

Using the number of metrics designated in Table 5.13, the following process was applied for the creation of the relevant files for each stage:

- 40% – for each of the 50 training group members, the training files consisted of 79 samples (30 for the positive training case and 49 for the negative training case), with 30 metrics per sample. The testing files consisted of 10,600 samples (100 for the member being tested, plus 100 for the other 49 training group members, plus 140 for the 40 non-training group members), with 30 metrics per sample, and the validation files consisted of 10 samples (with 30 metrics per sample).

- 50% – for each of the 50 training group members, the training files consisted of 79 samples (with 38 metrics per sample). The testing files consisted of 10,600 samples (with 38 metrics per sample), and the validation files consisted of 10 samples (with 38 metrics per sample).

- 60% – for each of the 50 training group members, the training files consisted of 79 samples (with 46 metrics per sample). The testing files consisted of 10,600 samples (with 46 metrics per sample), and the validation files consisted of 10 samples (with 46 metrics per sample).

- 70% – for each of the 50 training group members, the training files consisted of 79 samples (with 53 metrics per sample). The testing files consisted of 10,600

samples (with 53 metrics per sample), and the validation files consisted of 10 samples (with 53 metrics per sample).

When training the ANNs for each of the four stages, the designated number of metrics (per sample), as stated above, for each participant's training input file became the input layer nodes for their ANN.

Regardless of the number of input layer nodes, the number of hidden layer nodes was varied from 2 to 26 (inclusive), for each training input file. This was done so the most appropriate ANN configuration for each member could be determined by trying different configurations and seeing which one performed best. The reason for the upper limit of 26 hidden layer nodes was because that was the number used for training the fingerprint data and the fused data (in the complementary phase), and there seemed to be no reason to change adopting the same rationale.

As a result there were 1,250 individual ANNs trained (50 x 25) for each of the four stages, giving a total of 5,000 individual ANNs for the entire cooperative fusion training phase. The ANN training phase for cooperatively fused data (with the multiple configurations just described) was conducted using a desktop computer with 2 Ghz AMD processor and 512 MB RAM. As there were four stages, the following outlines the time taken for each stage:

- 40% – Training took 9 days (i.e. approximately 4.3 hours per participant).
- 50% – Training took 9 days (i.e. approximately 4.3 hours per participant).
- 60% – Training took 10 days (i.e. approximately 4.8 hours per participant).
- 70% – Training took 11 days (i.e. approximately 5.3 hours per participant).

Once training was completed, all ANN configurations were assessed to determine the single configuration (for a participant for each stage) that returned the least number of false acceptances (Type I errors) and false rejections (Type II errors). This process was performed manually because the determination required assessing the trade-off between the two error types. The configuration that returned the least number of Type I errors, whilst returning Type II errors at an acceptable level was

selected. The same process was applied for all training group members, and the weights of the ANN configurations (thus selected) were used as registered templates from that point onwards. These were written to a file (such that the participant number and number of hidden layer nodes were indicated in the file name, and given a '.w' extension) and subsequently used during the testing phase.

When testing the ANNs for each of the four stages, the designated number of metrics (per sample), as previously stated, for each participant's testing input file became the input layer nodes for their ANN. The stored weights (from the training phase) were applied to the ANNs, thus resulting in the correct configuration.

Positive case testing examines whether the ANN has correctly recognised samples belonging to the member that it has been trained to recognise (samples it has not seen during training). Non-recognition of any of these 100 positive case samples are instances of Type II errors (false negatives) (refer Chapter 6 section 6.2).

Negative case testing examines whether the ANN has correctly rejected samples belonging to someone other than the member it has been trained to recognise. Recognition of any of these negative case samples are instances of Type I errors (false positives) (refer Chapter 6 section 6.2).

The results of the testing phase are provided in Chapter 6, and a discussion of these results is presented in Chapter 7.

## 5.7 Experimental Validity

Experimental validity refers to the manner in which variables influence the results of the research, and whether these results can be generalised to the population at large (Heffner, 2004). In the design of an experiment, there is a desire to establish a causal relationship, where it can be established beyond doubt, that the observed effects (on the dependent variable) were caused only by the manipulation of the independent variable under consideration, and not by some other influence (Emory and Cooper, 1991; Albright and Malloy, 2000). That is, do the observed results truly represent the causality of manipulating the independent variable, or has some extraneous influence impacted?

In designing and conducting the current experiment—making decisions related to the choices described in this chapter—the methodological issues relating to experimental validity were given careful consideration. In particular, the relevant internal and external factors were deliberated.

### 5.7.1 Internal Validity

Whether an experiment employs accepted or recommended research design practices or not, there are factors which may bring into question the truth of observations and subsequent conclusions (Emory and Cooper, 1991). Internal validity deals with factors that could interfere with the ability to infer a truly causal relationship. The question to pose is, did the manipulation of the independent variable lead to the observed results (i.e. the affect on the dependent variable) or was some extraneous factor responsible?

Controlled treatment of experimental conditions and random assignment of participants data greatly enhances the ability to overcome threats to internal validity (Albright and Malloy, 2000). By adopting such measures, the legitimacy of conclusions drawn from observations gains credibility.

Of the major threats to internal validity (Emory and Cooper, 1991; Heffner, 2004), the following three were considered to have had minimal or no influence on the current study:

- History refers to capability of an event, outside of the research study (eg: environmental conditions), to alter or effect participants' performance. As the only involvement by participants in this study was to provide samples during the data collection phase, the only outside events that could have threatened the experiment were physiological changes. For example, if a participant injured their hand or fingers (during the data collection period), which could have affected their ability to type normally or to provide fingerprint scans of the right index finger. Such events did not occur to the participants who provided samples for this study.

- Maturation refers to the natural physiological or psychological changes that may take place in participants over a period of time (during participation in a study). For example, tiredness, boredom, hunger, aging. During keystroke dynamics data collection, participants were required to supply 160 correctly typed samples of the given phrase (as discussed in section 5.4.2). If participants had been asked to provide the entire 160 in one collection session, they may have grown tired or bored (which may have affected their performance).

  In an attempt to alleviate this possible concern, the decision was taken to collect only 20 samples in one collection session, and hold such collection sessions once per week for eight weeks (for each participant). This also facilitated the goal of keystroke dynamics, which is to differentiate typing patterns of different individuals based on their habitual typing style. So to allow participants to develop a habitual typing style for the given text, the experiment actually required the above decision to be made. It is doubtful that this collection process resulted in an inappropriate maturation effect.

- Selection refers to the manner in which participants are selected to participate in a study, and the manner in which they are assigned to groups. In the current study, participants were recruited on a volunteer bases, and so no selection by the researcher was involved other than the choice of populations from which the participants were drawn, which is discussed below. The random assignment of participants data to the training and non-training groups should have accounted for any validity threats. Also, during the creation of training, testing, and validation files for the training group members, samples were randomly selected.

The remaining possible threats to internal validity (Emory and Cooper, 1991; Heffner, 2004) were considered to have had no influence on the current study:

- Testing refers to the chance that participants will perform better in post-tests than they did in pre-tests (that is, when given the same test or a test of similar difficulty). This is a result of familiarity with the test. There were no tests

applied to participants in this study, and so no effects from this threat are anticipated.

- Instrumentation refers to changes in the measurement criteria or instruments during the course of the study. Such changes did not occur in this study.
- Statistical regression refers to the tendency for participants who score very high or very low in initial testing to score more toward the mean on subsequent testing. There were no tests applied to participants in this study, and so no effects from this threat are anticipated.
- Experimenter Bias refers to the possible bias a researcher may exhibit toward anticipated results. This could affect observations, and possibly even result in research methodological errors, that could skew conclusions drawn from the study. The only possibilities for bias in this study were associated with the data treatment after data collection. The processes discussed in this chapter were well considered and conform to accepted research practices. It was therefore considered that experimenter bias did not threaten validity.
- Mortality (or participant drop out) refers to changes in the composition of study groups over the duration of an experiment. Any drop out of participants in this study occurred during the data collection phase. Assignment to the training and non-training groups was not performed until after data collection was completed. It was therefore considered that mortality did not threaten validity.

In relation to the first three threats to internal validity discussed above, the processes described in this chapter were implemented to reduce any influence on the experiment. In relation to the latter five threats to internal validity, it is believed they had no influence on the experiment. Because of the nature of the current experiment, participants were not subjected to any testing or selection criteria. Collected data only were subjected to random assignment or selection as described in this chapter. Also, all data were subject to the same processes; there was no deviation in treatment[25].

[25]Copies of the data sets collected for, and used in, the current experiment are available upon request from the author or the author's supervisors.

### 5.7.2 External Validity

External validity is concerned with the interaction of the independent variable with factors other than those that may threaten internal validity (Emory and Cooper, 1991). It deals with the correctness of generalising inferences across time, settings and persons. Unlike threats to internal validity, threats to external validity cannot be controlled by random assignment (Albright and Malloy, 2000). The question to pose is, are the results of a study (with data collected from a sample population) truly representative of the entire population?

The following possible threats to external vadility (Emory and Cooper, 1991; Heffner, 2004) were considered to have had minimal or no influence on the current study:

1. Reactivity of Testing refers to the sensitising of participants by a pre-test so they respond differently in subsequent testing. There were no pre-tests conducted in this study, and so no sensitising of participants was possible.

2. Interaction of Selection refers to threats posed by the process of selecting participants (i.e. participants may not be truly representative of the wider population). In the current study participants were volunteers and were recruited by the processes described in section 5.3. Because of the nature of the groups from which the participants were drawn and the range of demographics, and the particular nature of this experiment, the results are not considered to be significantly affected by this threat to validity.

3. Other reactive factors - interaction between participants because of artificial settings (could encourage role playing). Data collection was supervised by the author and participants did not interact with one another during data collection, so it is believed that no role playing was possible.

4. Demand Characteristics refers to the ability of participants to pick up on cues about the anticipated results. Data collection was the only participation by participants. As the data collection involved repeatedly typing the same

phrase or repeatedly having the index finger scanned, there was no way for participants to influence results by any perceptions they may have formed.

5. Hawthorne Effects - if participants are aware of being observed, it may cause them to modify their performance. As the only other person present during all data collection sessions was the author, there was no reason or incentive for participants to modify their normal typing style. Monitoring by the author was done from a distance of approximately 3 metres; this to ensure that participants actually typed all samples (i.e. did not attempt to 'trick' the data capture program). For the fingerprint scanning, the author sat next the participants to ensure that scans were correctly captured.

6. Order Effects refer to the order in which treatments are performed (if multiple treatments are used). Whilst there was no treatment in terms of participants, there was treatment of their data as described in this chapter. The order of data treatment was conducted by necessity. That is, metrics extraction was required before the selection process, and selection process was necessary prior to final analysis.

As there were only 90 participants recruited for this study, the question of any results being generalisable needs consideration. Data belonging to the non-training group members (for all phases of the experiment) were not exposed to the ANNs during the training phases. So when testing the trained ANNs, the non-training members data were meant to represent the wider population. This meant that the results achieved could be considered generalisable to a large degree (though there were only 40 data sets in the non-training group).

## 5.8 Conclusion

In this chapter, the methodology relating to the current experiment has been described. Firstly, the requirements and recruitment of participants for the experiment were explained in section 5.3. The processing involved for the implementation of the

keystroke dynamics and the fingerprint recognition phases was described in sections 5.4 and 5.5 respectively, and included:

- The collection of raw data.
- The pre-processing of that data.
- The subsequent selection methods for extracting metrics from that data.

Importantly, it must be emphasised that the treatment of the keystroke dynamics and fingerprint feature data for the first two phases of the experiment were performed to meet the requirements of the third phase (i.e. feature level data fusion). That is, the data needed to be in a form that could be fused by a practical method. In particular, the pre-processing and metrics selection methods developed for both phases were (as far as is known) completely original and specifically designed for application in this study.

The treatment of data for either of the two initial phases could have been different if the requirement had been for a uni-modal biometric system, or if data was to be fused at a different level, or by a different method than that used in this experiment.

Finally, the final analysis was performed by ANNs to obtain recognition results for both phases (i.e. the two individual biometric characteristics). The results for phases 1 and 2 are presented in Chapter 6 sections 6.4.1 and 6.4.2 respectively, and discussed in detail in Chapter 7 sections 7.2.1 and 7.2.2 respectively.

Following the discussion of phases 1 and 2, phase 3 (feature level fusion of the two biometric characteristics) was discussed. The processing involved for the implementation of the two viable paradigms (complementary and cooperative) was described in sections 5.6.2 and 5.6.3 respectively, and included:

- The data source for both paradigms.
- The selection method for extracting metrics from those data. This was applicable for the cooperative paradigm only.
- The fusion methods for both paradigms (including the four stages involved for the cooperative fusion approach).

It is important to recognise that the methodology chosen for the selection of metrics (for the cooperative fusion approach), utilised an existing mathematical concept (i.e. the Approximate Relative Local Gain). However, its use for cooperative fusion of feature level data is (as far as is known) an original application. One issue with this methodology is the required existence of weights from trained ANNs (from the complementary data fusion approach). In some circumstances, the overheads associated with this aspect may make the application of this methodology inappropriate.

Finally, the final analysis was performed by ANNs to obtain recognition results for the combined data fused at the feature level. The results for the data fusion phase (for the complementary and cooperative paradigms) are presented in Chapter 6 sections 6.4.3.1 and 6.4.3.2 respectively, and discussed in detail in Chapter 7 sections 7.2.3.1 and 7.2.3.3 respectively.

During the performance of the experiment many choices were made in relation to treatment of data. Therefore, the reasons why particular decisions were taken were discussed, including those pertaining to experimental validity.

The next chapter discusses the findings from testing the trained ANNs with the data files obtained as described in this chapter.

# Chapter 6

# Research Results And Analysis Method

## 6.1 Overview

This chapter begins by introducing classification concepts for the analysis of authentication outcomes (section 6.2). A description of the specific methodology used to analyse the experimental results in the current study is then provided. This entailed the use of Receiver Operating Characteristics (ROC) graphs. An explanation of ROC analysis, and the reasons for utilising this methodology, are provided. Section 6.3 then discusses how ROC analysis was applied to the data in this study. The results for all phases of the experiment are then presented in section 6.4. Finally, the conclusion summarises the chapter (section 6.5).

## 6.2 Classification of Authentication Outcomes

### 6.2.1 Classification Measurement

Formally, the classification outcome for an authentication system is the likelihood that two samples belong to the same individual (Maltoni et al., 2003). That is, that both samples either belong to the same individual or they belong to different individuals.

In a traditional password authentication system, the likelihood that two samples belong to the same individual is either a perfect match (probability of 1) or a non-match (probability of 0). That is, the result of comparing two such samples is a discrete value in a binary domain.

However, biometric authentication systems cannot be expressed by such discrete values, because biometric samples of the same characteristic, taken at different times from the same individual, rarely match perfectly. That is, there is usually some degree of uncertainty. Typically, the likelihood that two samples belong to the same individual is expressed as a probability score over the continuous interval $[0, 1]$. That is, the result of comparing two such samples is a floating point value in a continuous domain.

In a binary domain, four possible classifications can be formally defined (Bradley, 1997; Fawcett, 2006; Flach, 2004):

1. True positive. This occurs when an instance/sample is a known member of a class, and a classifier correctly predicts membership of that class.

2. False positive. This occurs when an instance/sample is a known non-member of a class, but a classifier incorrectly predicts membership of that class—referred to as a Type I error.

3. True negative. This occurs when an instance/sample is a known non-member of a class, and a classifier correctly predicts non-membership of that class.

4. False negative. This occurs when an instance/sample is a known member of a class, but a classifier incorrectly predicts non-membership of that class—referred to as a Type II error.

These four classifications can be represented in a contingency table (also known as a confusion matrix) as demonstrated in Figure 6.1.

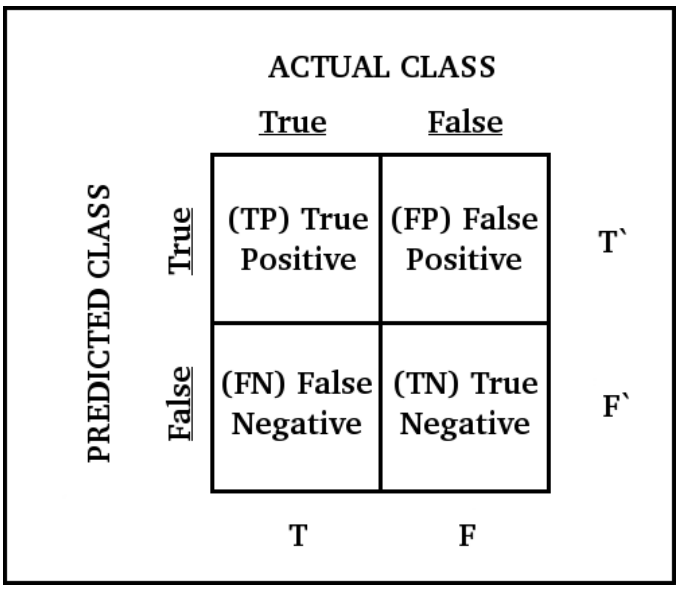


Figure 6.1: Contingency Table

The following explanations are provided to clarify the labels in Figure 6.1.

**Class Labels:**

`ACTUAL CLASS` refers to the class that an instance is known to be a member of. Its status can be True, if it is known to be a member of the class; False, if it is known to **not** be a member of the class.

`PREDICTED CLASS` refers to the class that a classifier predicts the instance to be a member of. Its status can be True, if the classifier determines that the instance is a member of the `ACTUAL CLASS`; False, if the classifier determines that the instance is **not** a member of the `ACTUAL CLASS`.

**Class Instances:**

**TP** is the total number of instances that are classified as true positives.

**FP** is the total number of instances that are classified as false positives—Type I errors.

**TN** is the total number of instances that are classified as true negatives.

**FN** is the total number of instances that are classified as false negatives—Type II errors.

**Column Totals:**

**T** is the sum of the `ACTUAL CLASS` instances with True status, calculated using Equation 6.1.

$$T = TP + FN \tag{6.1}$$

**F** is the sum of the `ACTUAL CLASS` instances with False status, calculated using Equation 6.2.

$$F = FP + TN \tag{6.2}$$

**Row Totals:**

**T′** (pronounced T prime) is the sum of the `PREDICTED CLASS` instances with True status, calculated using Equation 6.3.

$$T' = TP + FP \tag{6.3}$$

**F′** (pronounced F prime) is the sum of the `PREDICTED CLASS` instances with False status, calculated using Equation 6.4.

$$F' = FN + TN \tag{6.4}$$

The following measurements or rates can be determined from the class labels and instances, and the row and column totals indicated in Figure 6.1.

**Rates:**

- False positive rate ($fpr$) is the result of dividing the number of false positive instances (**FP**) by the number of `ACTUAL CLASS` instances with False status (**F**). It is calculated as demonstrated by Equation 6.5. This rate is also known as the false alarm rate.

$$fpr = FP/F \tag{6.5}$$

- True positive rate ($tpr$) is the result of dividing the number of true positive instances (**TP**) by the number of `ACTUAL CLASS` instances with True status (**T**). It is calculated as demonstrated by Equation 6.6. This rate is also known as the hit rate or recall.

$$tpr = TP/T \tag{6.6}$$

- Precision (also known as the positive prediction value) can be calculated as demonstrated by Equation 6.7.

$$Precision = TP/T' \tag{6.7}$$

- Accuracy is the weighted average of the true positive instances (**TP**) and the true negative instances (**TN**), and is calculated as according to Equation 6.8.

$$Accuracy = (TP + TN)/(T + F) \tag{6.8}$$

- Sensitivity is equivalent to the $tpr$, and is calculated using Equation 6.6.

$$Sensitivity = tpr = TP/T$$

- Specificity is the compliment of the $fpr$. It can be calculated by two methods as demonstrated by Equations 6.9 and 6.10.

$$Specificity = 1 - fpr \tag{6.9}$$

$$Specificity = TN/F \tag{6.10}$$

Findings in biometrics research are typically expressed using the performance metrics/variables known as the False Acceptance Rate (FAR) and the False Rejection Rate (FRR)[1]. These metrics/variables are related to the Type error definitions mentioned above, and are respectively the Type I error rate and Type II error rate.

[1]Some researchers use the terms False Match Rate (FMR) and the False Non-Match Rate (FNMR) respectively.

That is, the FAR is the ratio of Type I errors; the FRR is the ratio of Type II errors. The reason for using these particular metrics/variables or Type error rates is because in authentication the interest is in measuring mis-classification, rather than correct classification.

The two metrics/variables relate to the measurements listed above (derived from Figure 6.1) as follows:

- The False Acceptance Rate (FAR) is equivalent to the false positive rate ($fpr$), and is therefore calculated by Equation 6.5. That is,

$$FAR = fpr = FP/F$$

- The False Rejection Rate (FRR) is the compliment of the true positive rate ($tpr$). It can be calculated by two methods as demonstrated by Equations 6.11 and 6.12.

$$FRR = 1 - tpr \tag{6.11}$$

$$FRR = FN/T \tag{6.12}$$

As mentioned at the start of this section, authentication systems involving biometrics do not typically deal with classifier outcomes in the binary domain, but rather the continuous domain (because of the degree of uncertainty). However, the classification measurements discussed in this section can be adapted for applications where classifier outcomes are in the continuous domain.

Given a classifier outcome in the continuous domain, classification necessitates a subjective decision being made on whether or not the predicted outcome (from testing a sample) should be accepted or rejected as being a member of the `Actual Class`. This determination may be termed 'the final classification decision', and typically involves the use of a threshold or cut-off value that is applied to the classifier outcome. This threshold is often termed the 'decision threshold'. The decision threshold provides a cut-off value that can be adjusted to allow for uncertainty, and can be suitably determined for the particular application under consideration.

The method adopted for determining the decision threshold in the current study is described in section 6.3.4. The next section discusses an area of data analysis particularly suitable for classifier outcomes in the continuous domain, and facilitates the determination of the performance metrics/variables discussed in this section.

### 6.2.2 Receiver Operating Characteristics (ROC) Graphs

A Receiver Operating Characteristics (ROC) graph provides a means for analysing the performance of classifiers (Fawcett, 2006). That is, when faced with the final classification decision based on the performance of classifiers, ROC graphs provide a graphical representation of their performance to assist classification.

ROC graphs have historically been used to document the trade-off between hit rates ($tpr$) and false alarm rates ($fpr$) in signal detection theory (Swets, 1988). More recently, ROC graphs have increasingly been employed in the machine learning environment to provide a more accurate metric than previous classification measures (Provost et al., 1998). In machine learning, a classifier is an algorithm that has been trained on certain data; it is then tested on data it has not previously seen and its classification performance is assessed.

#### 6.2.2.1 ROC space

A classifier's performance can be represented in a two dimensional coordinate system, where the false positive instance is plotted along the x axis and the true positive instance is plotted along the y axis. The origin has coordinates $(0, 0)$, the maximum value on the x axis has coordinates $(1, 0)$, the maximum value on the y axis has coordinates $(0, 1)$, and therefore the maximum value at the termination of the major diagonal has coordinates $(1, 1)$.

The area encompassed within these coordinate boundaries is known as ROC space[2] (refer Figure 6.2). Each point plotted in ROC space (referred to as an operating point) represents the coordinate values of a false positive/true positive $(FP, TP)$ pair, and corresponds to the classifier's performance.

[2]Note, this area also denotes a unit square.

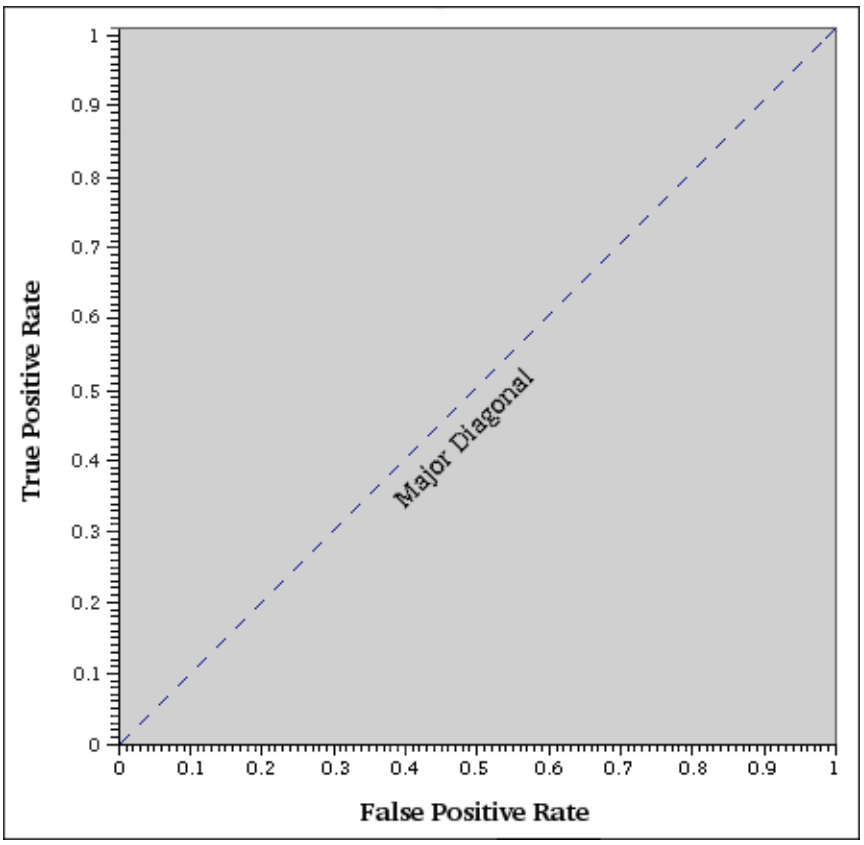


Figure 6.2: ROC Space

Figure 6.3 demonstrates the performance of four hypothetical classifiers A, B, C, and D, showing their corresponding coordinate values. Classifiers A, B, C, and D represent the outcome of binary classification because there is just one operating point for each classifier.

ROC graphs also allow for the analysis of the performance of classifiers that produce continuous output. This is achieved by applying an adjustable determinant value to the classifier output, to obtain operating points—i.e. $(FP, TP)$ pairs—whose coordinate values vary as the adjustable determinant value is incremented from 0.0 to 1.0 by an arbitrary amount.

This adjustable determinant value[3] is often referred to as a threshold but is not referred to as such here, so as not to confuse this value with the decision threshold (which is used when making the final classification decision). Hereafter in this document, the adjustable determinant value will be referred to as the AD value.

[3]Note that the adjustable determinant value is a variable whose successive values produce classifier outcomes (coordinates) corresponding to individual operating points—i.e. $(FP, TP)$ pairs.

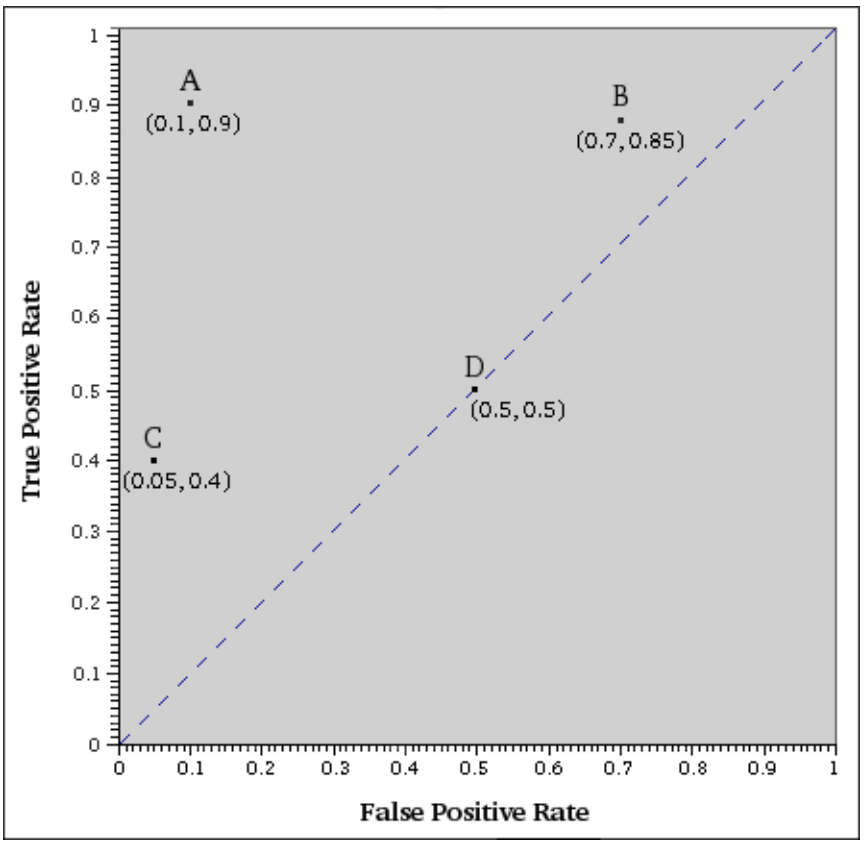


Figure 6.3: Binary Classifiers

There are a number of properties associated with ROC graphs in ROC space (Bradley, 1997; Fawcett, 2006):

- The operating point $(0, 0)$ means that the classifier has predicted zero false positives, and has also predicted zero true positives. That is, no instances (whether they are known members of the class or not) have been predicted to be members of the class. Typically, operating points that occur in the lower left region of an ROC graph (but above and left of the major diagonal) are considered to be 'conservative', because the number of acceptances (whether true or not) tend to decrease as the AD value increases (toward 1.0). Operating point C in Figure 6.3 could be considered an example of conservative classification.

- The operating point $(1, 1)$ means that the classifier has predicted 100% false positives, and also predicted 100% true positives. That is, all instances (whether they are actually members of the class or not) have been predicted to be members the class. Typically, operating points that occur in the upper right region

of an ROC graph (but above and left of the major diagonal) are considered to be 'liberal', because the number of acceptances (whether true or not) tend to increase as the AD value decreases (toward 0.0). Operating point B in Figure 6.3 could be considered an example of liberal classification.

- The operating point (0,1) means that the classifier has demonstrated perfect prediction. That is, the classifier has correctly predicted 100% True Positives and 0% False Positives. This is the best possible outcome, because there have been no instances of incorrect prediction. Typically, operating points that occur in the top left region of an ROC graph demonstrate 'good' prediction performance or classification. Operating point A in Figure 6.3 could be considered an example of good classification.

- Operating points whose predicted coordinate values lie along the major diagonal (from $(0, 0)$ to $(1, 1)$) represent random classifications. That is, they are equally likely to have been incorrectly predicted as they are to have been correctly predicted. Operating point D in Figure 6.3 could be considered an example of random classification.

- Operating points whose predicted coordinate values lie below the major diagonal represent worse than random classification.

Plotting numerous operating points in ROC space (by applying an AD value), can hypothetically be viewed as forming an ROC curve. Figure 6.4 illustrates an example of a ROC curve which demonstrates very good classification.

The granularity of an ROC curve will be dependent on the number of AD values used to produce the operating points. If the AD value is incremented at a fine resolution, an ROC curve can be accurately estimated. For example, at a resolution of 100 the AD value is incremented by 0.01, which will allow for a fine enough granularity to produce an accurately representative ROC curve.

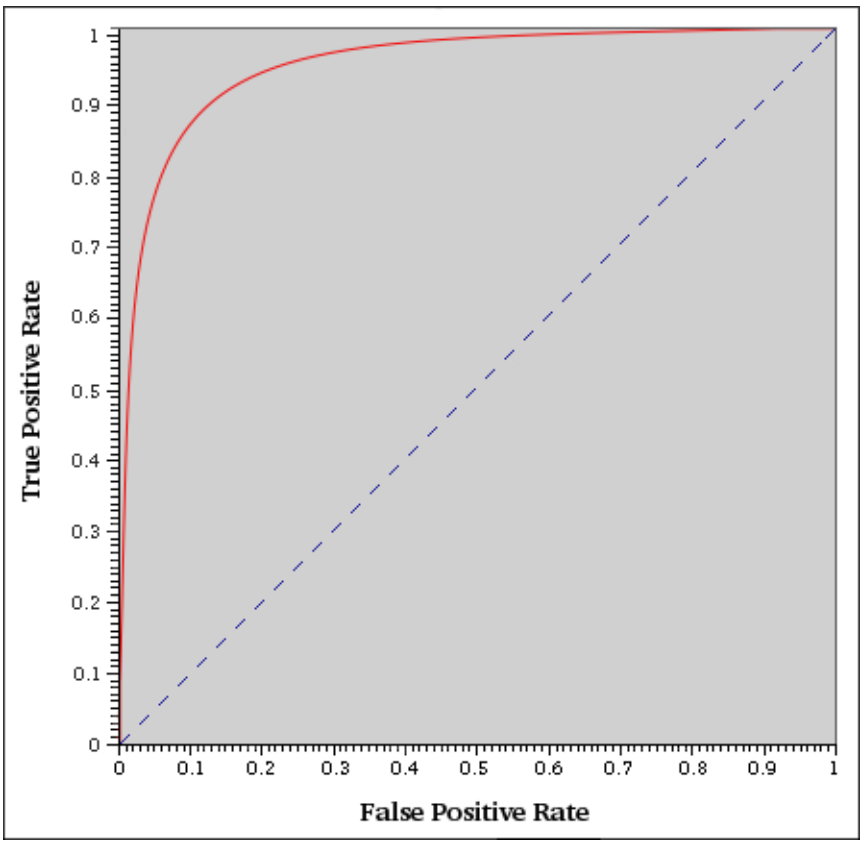


Figure 6.4: ROC Curve

The reasons for utilising ROC curves in the analysis of this study's experimental results are:

- Testing involved applying test samples to the trained ANNs for all training group members, which produced continuous data output[4]. By applying an AD value to each ANN output, an ROC curve could be obtained for each training group member.

- ROC curves are ideal for representing continuous data. By their very nature, ROC curves demonstrate the trade-off between false positive and true positive instances (Bradley, 1997; Fawcett, 2006; Greiner et al., 2000). This can be helpful in determining the 'best' decision threshold to utilise for final classification.

[4]Note that as described in sections 5.4.5.2 and 5.5.6.2, each training group member had one test file consisting of 100 of their own samples (tested for true positives) and 10,500 samples from the other 49 training group members and the 40 non-training group members (tested for false positives).

- The representation of data in ROC curves allows for the extraction of other information, which facilitates further analysis. In particular, there are two statistics (important to this study) that can be extracted. These are the area under the ROC curve and the optimal operating point (refer sections 6.2.2.2 and 6.2.2.3 respectively).

The next section discusses the area under the ROC curve, its possible uses, and presents a method for its calculation.

#### 6.2.2.2 Area Under The ROC Curve

When considering a classifier's performance, it can be useful to obtain a summary statistic that represents its overall classification performance (Greiner et al., 2000). This could facilitate the comparative performance of one classifier with that of another classifier. One such summary statistic, well documented in the literature, is termed the area under the ROC curve (abbreviated to AUC).

In section 6.2.2.1 a description of the boundaries of ROC space was presented; these boundaries denote the unit square. Therefore, any statistic calculated to represent the area under an ROC curve must be in the continuous interval $[0, 1]$.

As described in section 6.2.2.1, the area below and to the right of the major diagonal demonstrates random or less than random classification. This region equates to an area of 0.5, and so any classifier that produces an AUC value less than or equal to 0.5 could be considered non-informative (Greiner et al., 2000).

In fact, Swets (1988) proposed a possible scale to describe a classifier's performance utilising the AUC statistic (refer Table 6.1).

| Statistic | Description |
|---|---|
| $AUC = 1$ | Perfect |
| $0.9 < AUC < 1.0$ | Highly Accurate |
| $0.7 < AUC \leq 0.9$ | Moderately Accurate |
| $0.5 < AUC \leq 0.7$ | Less Accurate |
| $0.0 \leq AUC \leq 0.5$ | Non-Informative |

Table 6.1: AUC Statistic Decriptions

The ranges of the Statistic in Table 6.1 (column 1) conform with the concept of the area of a unit square being no greater than 1.0, and the area below the major diagonal being between 0.0 and 0.5.

Perfect classification occurs if the AUC equals 1.0. The other ranges nominated in Table 6.1 suggest possible descriptions for specifying the performance of a classifier from Highly Accurate to Non-Informative.

Equations 6.13, 6.14, and 6.15 show the formulae for calculating the AUC, using the trapezoidal integration method proposed by Bradley (1997).

$$AUC = \sum_i \left\{ (tpr_i \times \triangle fpr) + \frac{1}{2} (\triangle tpr \times \triangle fpr) \right\} \tag{6.13}$$

where

$$\triangle fpr = fpr_i - fpr_{i-1} \tag{6.14}$$

$$\triangle tpr = tpr_i - tpr_{i-1} \tag{6.15}$$

Figure 6.5 illustrates the ROC curves of two classifiers (A and B) for comparison. It can be seen that classifier A, which could be described as highly accurate (with an area of approximately 0.93), generally performed better than classifier B, which could be described as moderately accurate (with an area of approximately 0.86).

Note that this does not necessarily mean that classifier A performed better than classifier B under all circumstances. From Figure 6.5 it can discerned that classifier B in fact out performed classifier A when the False Positive Rate is greater than approximately 0.23. Of course, a False Positive Rate of 0.23 is unlikely to be practical for authentication purposes.

Nonetheless, Figure 6.5 highlights a limitation of the AUC statistic when comparing the performance of different classifiers. The statistic is a single value and attributes equal weighting to all parts under the curve. It is therefore possible that the comparative performance of two classifiers, using the AUC statistic, will be non-significant if they differ in an area of practical relevance (Greiner et al., 2000).

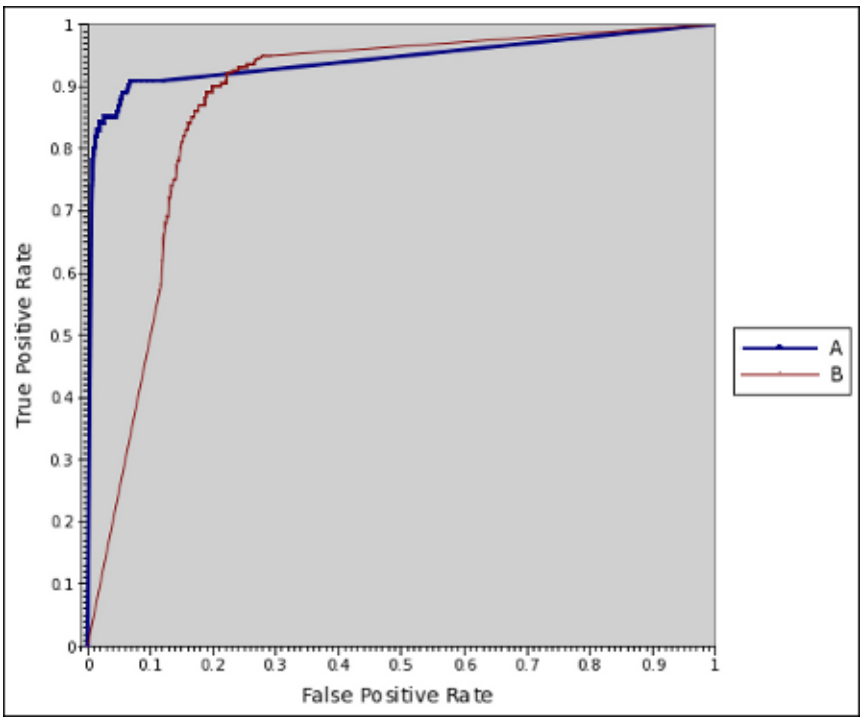


Figure 6.5: Comparison of AUC for Two Classifiers

This suggests it is doubtful that the AUC can be used to show significant evidence of comparative accuracy (or infer experimental proof) when comparing the performance of two or more classifiers (Provost et al., 1998). However, the statistic may be justifiably used as indicative, in the broader sense, of better performance. For example, classifier A generally performed better than classifier B.

Because of this limitation, the AUC values (as measures of accuracy) for the training group members in this study were used only to help determine the most appropriate decision threshold for presentation of results (refer section 6.3.1).

The next section discusses the concept of the optimal operating point, what it is typically used for, and possible methods for its calculation.

#### 6.2.2.3 Optimal Operating Point

The Optimal Operating Point (abbreviated to OOP) is that operating point on the ROC curve that indicates optimal performance. The question is, what constitutes optimal performance? The answer depends on the application to which the classifier is to be applied.

For profit-loss (or benefit-cost) applications, the OOP will be that operating point which provides the best trade-off between profit and loss (or benefit and cost). The task is to determine the point at which the *tpr* and the $fpr$ are both optimised. This typically occurs when the number of true positives (*tpr*) is maximised and the number of false positives ($fpr$) is minimised. The operating point (on the ROC curve) closest to the point (0,1) represents the best classification (refer section 6.2.2.1), and therefore is most likely to be determined as the OOP for these types of applications.

For authentication purposes (using biometrics), the task is not as straightforward. Theoretically, it is still the objective to find a point where the *tpr* is maximised and the $fpr$ is minimised. However, the primary objective is to reduce the $fpr$ (i.e. the number of false positives) as much as possible; this typically eventuates in a decrease in the *tpr* (i.e. the number of true positives). To achieve reduction in the $fpr$, means nominating an operating point (on the ROC curve) closer to the point (0,0)—that is, moving away from the point (0,1) toward the origin. The degree to which the $fpr$ is reduced (and consequently the *tpr* is decreased) will depend on the specific application.

As discussed in section 6.2.1, the performance metric $far$ is equivalent to the $fpr$ and the performance metric $frr$ is the compliment of the *tpr*. So relating the discussion in the previous paragraph to these performance metrics, reducing the $far$ is likely to increase the $frr$. The task then is to determine the operating point that reduces the $far$ as much as is practicable, whilst attempting to maintain the $frr$ within an acceptable limit. Again, the limit to which the $frr$ is permitted to increase will depend on the specific application.

Assuming an ROC curve with no discontinuities and shaped similar to those presented to this point in the discussion, there are mathematical methods for determining the turning point of a curve (Greiner et al., 2000). Let us further assume that this turning point will be the closest point (on the ROC curve) to the point (0,1). If a formula for the curve can be determined, then the first derivative can be used to find the turning point (given the previous assumptions). If only raw data

is available, numerical differentiation may prove successful in determining a turning point. However, there may be multiple turning points in a curve; also the above assumptions may not always be applicable (as is evident with classifiers A and B in Figure 6.5). In such cases, other steps would need to be taken to find the optimal operating point.

Another way to determine the OOP is to use the iso-performance lines of an ROC graph (Flach, 2004; Hong, 2009). These lines are essentially tangents to the ROC curve, and can be used to determine the OOP according to the slope of the tangent at a particular operating point. Hong (2009) proposed using iso-performance lines to refine the accuracy statistic (refer Equation 6.8) which can then be used to identify the OOP (for benefit-cost applications).

However, a much simpler method for determining the OOP is to use a distance measurement. In section 6.2.2.1 it was stated that the point (0,1) in ROC space demonstrates the best possible classifier performance. This infers that an operating point on the ROC curve closest to the upper left corner of the unit square (i.e. the point (0,1)) optimises performance (Greiner et al., 2000). That is, the operating point (on the ROC curve) closest to the point (0,1) determines the OOP, because the classifier performed better at this point than at any other operating point on the ROC curve. A simple distance measure can be used to calculate distances from all operating points on the ROC curve to the point (0,1); the smallest distance would indicate the OOP.

The next section describes the calculation of the operating points (section 6.3.2) and AUC (section 6.3.3) as implemented for this study. Also discussed is how the AUC and the OOP were utilised to help determine the decision threshold (section 6.3.4). These tasks were conducted for all participants, for each phase of the experiment—including keystroke dynamics, fingerprint recognition, and the two paradigms for data fusion (complimentary and cooperative). Once established, the decision threshold was used to obtain the results that are then presented in section 6.4 and discussed in Chapter 7.

## 6.3 Applying ROC In This Study

### 6.3.1 Introduction

Chapter 5 discussed in detail the experimental procedures used in this study for user authentication via keystroke dynamics (section 5.4), fingerprint recognition (section 5.5), and data fusion (section 5.6).

As discussed in sections 5.4.5.2 and 5.5.6.2, one testing file was created for each training group member, with each file consisting of:

- 100 test samples belonging to the training group member that the ANN had been trained to recognise.
- 10,500 test samples from the other 49 training group members and the 40 non-training group members.

Testing involved applying a training group member's test file (with all 10,600 samples) to the ANN that had been trained to recognise their samples. For this process, the saved weights from the training phase (for that training group member) were loaded by the ANN, and each test sample was then evaluated by the ANN (using the saved weights), and a probability score for each was produced. This score represented the likelihood that a test sample belonged to the training group member that the ANN had been trained to recognise.

So, the result was an ANN output file consisting of its predictions for each sample applied to that ANN. This was done for all training group members, so there were 50 such ANN output files.

The goal of testing was to determine the correct recognition or rejection of a test sample applied to a trained ANN. Ideally, if a test sample was known to belong to a training group member, then when that sample was applied to their trained ANN, it could be expected that the ANN output would indicate the sample did indeed belong to that training group member. Conversely, if a test sample was known to **not** belong to any training group member, then when that sample was applied to any of the trained ANNs, it could be expected that the ANN output would indicate the sample did **not** belong to any training group member.

There are two issues here that need clarification:

1. The ANN output is a prediction only. It does not necessarily mean that the ANN has predicted correctly.

2. The ANN output (prediction) is a probability score in the continuous interval $[0, 1]$, where 0 indicates the lowest probability of the sample matching that of a training group member and 1 indicates the highest probability of the sample matching that of a training group member.

Given the probability score for an ANN output, a method was required to make a final classification decision. That is, to definitively decide (based on the probability score) whether to accept or reject the sample as belonging to the training group member that the ANN had been trained to recognise.

An option is to use AUC and OOP calculations to assist in making this decision. However, as demonstrated in sections 6.2.2.2 and 6.2.2.3 the AUC and OOP had the following limitations:

- The AUC can only be justifiably used to compare whether one classifier performed 'better' than another classifier. It can **not** infer experimental proof when comparing the performance of two or more classifiers.

- For authentication purposes, the OOP will most probably **not** be the closest operating point (on the ROC curve) to the point $(0, 1)$.

Though these limitations exist for the specific circumstances discussed, and therefore do impose some restrictions on the use of the AUC and OOP for analysis of results in the current experiment, they may still have the following possible applications:

1. The AUC can be used as a confidence indicator of an individual classifier performance.

2. It is still possible to determine an operating point on a ROC curve (but possibly not the OOP) to meet the conditions required for authentication purposes.

Before describing how the AUC and OOP were utilised in this study (refer sections 6.3.3 and 6.3.4), the next section describes the method used to calculate the operating points to obtain an ROC curve for each training group member. This is described first because the operating points that form an ROC curve are required for the calculation of the AUC and the OOP.

### 6.3.2 Calculation of ROC Operating Points

Algorithm 6.1 demonstrates the method used to determine the operating points to represent each training group member's classifier (i.e. trained ANN) output. Recall that the number of samples tested for correct acceptance (i.e. true positives) was 100 per training group member, and the number of samples tested for false acceptance (i.e. false positives) was 10,500 per training group member (refer sections 5.4.5.2 and 5.5.6.2).

For each AD value[5], all 10,600 samples were applied to a trained ANN (for each training group member). Note that the first 100 samples, in a test file, were the samples tested for correct acceptance (refer Algorithm 6.1 Steps 2–8), and the next 10,500 samples were the samples tested for false acceptance (refer Algorithm 6.1 Steps 10–16).

If the ANN output was greater than or equal to the current AD value, then that test sample was considered a member of the class that it was being tested against (refer Algorithm 6.1 Steps 3–5 and Steps 11–13). From this prediction, an accumulation of such occurrences was obtained, and the resultant metric determined (refer Algorithm 6.1 Steps 8 and 16).

As an example of the representation provided by an ROC curve, Figure 6.6 presents a graph from the keystroke dynamics testing phase for participant 1. Because of the resolution of the AD value, which determined 101 operating points, an ROC curve can be accurately estimated. This is demonstrated in Figure 6.6 by the operating points being plotted, and a line joining the points to form the ROC curve.

[5]Note that the AD value is incremented by 0.01 (refer Algorithm 6.1 Step 17) from 0.0 to 1.0 inclusive (refer Algorithm 6.1 Step 1). This determined 101 different AD value, which resulted in 101 operating points.

**Algorithm 6.1** Calculation of ROC Operating Points

Let $netValues$ be an array containing the $10,600$ output values from testing the trained ANNs.
Let $truePositives \leftarrow 100$ corresponding to the first 100 test samples in the $netValues$ array for testing correct recognition.
Let $falsePositives \leftarrow 10,500$ corresponding to the remaining test samples in the $netValues$ array for testing false acceptances.
Let $ADV$ be a variable representing the AD value that increments from 0.0 to 1.0 in increments of 0.01.
Let $tpr$ be an array containing the True Positive instances as $ADV$ varies from 0.0 to 1.0 in increments of 0.01.
Let $fpr$ be an array containing the False Positive instances as $ADV$ varies from 0.0 to 1.0 in increments of 0.01.
Let $count \leftarrow 0$ be a counter variable for accumulating the number of $netValues$ instances that meet the threshold condition.
Let $i \leftarrow 0$ be a loop control variable for accessing the appropriate elements of $netValues$.

1: **For** $ADV \leftarrow 0.0$ **to** $1.0$ **do**
2: **For** $i \leftarrow 0$ **to** $truePositives$ **do**
3: **If** $(netValues[i] \geqslant ADV)$
4: $count \leftarrow count + 1$
5: **End If**
6: $i \leftarrow i + 1$
7: **End** $i$ **For**
8: $tpr[i] \leftarrow count/truePostivies$
9: $count \leftarrow 0$
10: **For** $i \leftarrow truePositives$ **to** $falsePositives$ **do**
11: **If** $(netValues[i] \geqslant ADV)$
12: $count \leftarrow count + 1$
13: **End If**
14: $i \leftarrow i + 1$
15: **End** $i$ **For**
16: $fpr[i] \leftarrow count/falsePostivies$
17: $ADV \leftarrow ADV + 0.01$
18: **End** $ADV$ **For**

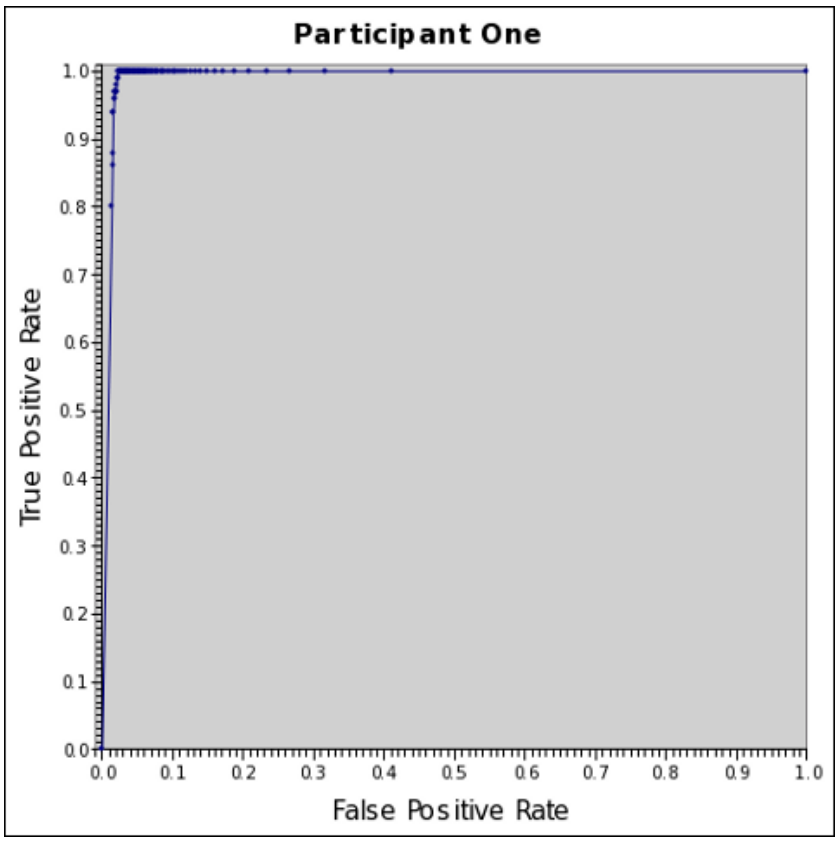


Figure 6.6: ROC Curve for Participant 1

### 6.3.3 Calculation of The Area Under The ROC Curve

Algorithm 6.2 shows the implementation of Equations 6.13, 6.14, and 6.15 for determining the AUC for each training group member. As mentioned in section 6.2.2.2, the method is that proposed by Bradley (1997).

The AUC values calculated via this method are presented in the tables of statistics in sections 6.4.1, 6.4.2 and 6.4.3.

In the next section, the determination of a decision threshold (for each participant, for each phase of the experiment) is discussed. The decision threshold allows for the nomination of an operating point at which the analysis of the performance of each classifier can be observed. For the determination of decision thresholds, points 1 and 2 highlighted in section 6.3.1 (at the bottom of page 334) were applied. As with the AUC values, the decision thresholds are presented in the tables of statistics in the sections 6.4.1, 6.4.2 and 6.4.3.

**Algorithm 6.2** Area Under The ROC Curve Calculation

Let $tpr$ be an array containing the stored True Postive values as the $ADV$ was varied from 0.0 to 1.0 (inclusive) in increments of 0.01.
Let $fpr$ be an array containing the stored False Postive values as the $ADV$ was varied from 0.0 to 1.0 (inclusive) in increments of 0.01.
Let $length \leftarrow 101$ be the length of the two arrays $tpr$ and $fpr$.
Let $deltaTP$ be a temporary variable for storing the calculated change between successive $tpr$ values.
Let $deltaFP$ be a temporary variable for storing the calculated change between successive $fpr$ values.
Let $AUC$ be a variable for storing the calculated area under the ROC curve.
Let $i$ be a loop control variable to access successive elements of $tpr$ and $fpr$.

1: **For** $i \leftarrow 1$ **to** $length$ **do**
2: $\quad deltaTP \leftarrow tpr[i] - tpr[i-1]$
3: $\quad deltaFP \leftarrow fpr[i] - fpr[i-1]$
4: $\quad AUC \leftarrow AUC + ((tpr[i] * deltaFP) + ((deltaTP * deltaFP)/2.0)))$
5: $\quad i \leftarrow i + 1$
6: **End** $i$ **for**

### 6.3.4 Calculation of Decision Threshold

According to Bradley (1997), the visual representation of classifier performance provided by ROC graphs facilitates the determination of a decision threshold. For many applications the decision threshold is the OOP. That is, the operating point that results in the 'best' performance of the classifier. Of course, best performance is subjective depending on the application area (as discussed in the section 6.2.2.3).

Given the previously stated provisions, a suitable decision threshold can be determined for authentication purposes. The decision threshold should return the lowest $fpr$, but with an acceptable $tpr$. This is a typical trade-off situation when presenting results in a continuous domain for authentication purposes (Wayman et al., 2005).

Note that when the $fpr$ decreases (as the AD value approaches 1.0), the $tpr$ also typically decreases. As explained in section 6.2.1, the $fpr$ is presented as the performance metric $far$, and the compliment of the $tpr$ is presented as the performance metric $frr$ (as demonstrated by Equation 6.11—i.e. $frr = 1 - tpr$). Thus, if the $tpr$ decreases, the $frr$ increases.

So the objective was to obtain the lowest $far$ whilst still achieving an acceptable $frr$. This entailed setting an acceptable upper limit on the $frr$, as it was more suitable to allow the $frr$ (rather than the $far$) to achieve higher values.

Thus it seemed appropriate to focus attention on the true positive instances when determining the decision threshold (because the true positive instances have a linear relationship with the $frr$). There were 100 samples available (per training group member) for testing true positive instances. Where possible it was preferable that the number of true positives was greater than 90. The compliment of this means that the number of false rejections would be no more than 10 (though this was unattainable in some cases).

By calculating the mean of the ANN outputs for all 100 true positive test samples, a true positive mean ($TPmean$) was obtained for each training group member. Next, a confidence level was determined (for each training group member) and applied to their $TPmean$. In determining a confidence level for each training group member, their $AUC$ value was utilised because it was a measure of confidence in the performance of the classifier when their unseen data was applied to it.

The decision to utilise the $AUC$ values (instead of the $TPmean$ values) as a measure of confidence, was taken for two reasons:

1. After comparing the differences between the $AUC$ values and the $TPmean$ values for the keystroke dynamics testing output (refer Table 6.3), it was evident that the differences between them were generally inconsiderable. With the average of the $AUC$ values being 0.9725 and the average of the $TPmean$ values being 0.9535, the difference of the two averages was 0.0189 (with the $AUC$ average being the larger). So there was every reason to lean toward using the $AUC$ values as confidence indicators.

2. From column four in Table 6.3, there are only 8 participants having the $AUC$ values less than the $TPmean$ values. This is an indication of the $AUC$ statistic being a more discriminative measurement than the $TPmean$. Logically, this holds true as well, since the $AUC$ is a statistic from testing both true positive and false positive samples (whereas the $TPmean$ is calculated from the true positive samples only).

Note, that all values in Table 6.3 have been rounded to four decimal places.

| Participant | Area Under Curve (AUC) | True Positive Mean (TPMean) | Difference |
|---|---|---|---|
| 1 | 0.9914 | 0.9893 | 0.0021 |
| 2 | 0.9566 | 0.9533 | 0.0033 |
| 3 | 0.8806 | 0.8589 | 0.0217 |
| 5 | 0.9979 | 0.9947 | 0.0031 |
| 7 | 0.9932 | 0.9881 | 0.0051 |
| 9 | 0.9616 | 0.9627 | -0.0012 |
| 12 | 0.9199 | 0.9398 | -0.0199 |
| 14 | 0.9952 | 0.9865 | 0.0087 |
| 16 | 0.9944 | 0.9712 | 0.0232 |
| 18 | 0.9997 | 0.9986 | 0.0011 |
| 20 | 0.9937 | 0.9766 | 0.0170 |
| 21 | 0.9827 | 0.9670 | 0.0157 |
| 23 | 0.9753 | 0.9554 | 0.0199 |
| 24 | 0.9528 | 0.9577 | -0.0049 |
| 25 | 0.9906 | 0.9537 | 0.0369 |
| 27 | 0.9995 | 0.9911 | 0.0084 |
| 29 | 0.9412 | 0.9324 | 0.0088 |
| 32 | 0.9795 | 0.9431 | 0.0365 |
| 34 | 0.9992 | 0.9800 | 0.0191 |
| 36 | 0.9921 | 0.9829 | 0.0092 |
| 38 | 0.9969 | 0.9903 | 0.0066 |
| 40 | 0.9983 | 0.9993 | -0.0009 |
| 41 | 0.9871 | 0.9565 | 0.0306 |
| 43 | 0.8626 | 0.8638 | -0.0012 |
| 45 | 0.9888 | 0.9986 | -0.0099 |
| 46 | 0.9962 | 0.9923 | 0.0039 |
| 47 | 0.9822 | 0.9541 | 0.0282 |
| 49 | 0.9454 | 0.9750 | -0.0296 |
| 52 | 0.9999 | 0.9983 | 0.0016 |
| 54 | 0.9856 | 0.9646 | 0.0210 |
| 56 | 0.9755 | 0.9171 | 0.0584 |
| 58 | 0.9518 | 0.9188 | 0.0330 |
| 60 | 0.9976 | 0.9784 | 0.0192 |
| 61 | 0.9425 | 0.8405 | 0.1019 |
| 63 | 0.9556 | 0.9330 | 0.0226 |
| 65 | 0.9797 | 0.8951 | 0.0846 |
| 67 | 0.9630 | 0.9051 | 0.0579 |
| 68 | 0.9723 | 0.9475 | 0.0248 |
| 69 | 0.9923 | 0.9708 | 0.0214 |
| 72 | 0.9985 | 0.9935 | 0.0050 |
| 74 | 0.8683 | 0.8454 | 0.0229 |
| 76 | 0.9876 | 0.9737 | 0.0139 |
| 78 | 0.9781 | 0.9446 | 0.0335 |
| 80 | 0.9726 | 0.9538 | 0.0188 |
| 81 | 0.9544 | 0.8973 | 0.0571 |
| 83 | 0.9838 | 0.9625 | 0.0213 |
| 85 | 1.0000 | 0.9966 | 0.0034 |
| 87 | 0.9554 | 0.9259 | 0.0295 |
| 89 | 0.9836 | 0.9842 | -0.0006 |
| 90 | 0.9703 | 0.9170 | 0.0533 |
| **Average** | 0.9725 | 0.9535 | 0.0189 |

Table 6.3: Comparison Between AUC and TPMean for Keystroke Dynamics

Given the figures in Table 6.3, it seemed appropriate to use the $AUC$ values to indicate a level of confidence when determining a decision threshold. For the current study, the confidence levels were varied because of the variability demonstrated by the $AUC$ values for each training group member (particularly for keystroke dynamics). Accordingly, the confidence levels were allocated as demonstrated in Table 6.4.

| **Area Under Curve** | **Confidence Level** |
|---|---|
| $AUC > 0.95$ | 0.0500 |
| $0.90 < AUC \leq 0.95$ | 0.0625 |
| $0.85 < AUC \leq 0.90$ | 0.0750 |
| $AUC \leq 0.85$ | 0.0875 |

Table 6.4: Confidence Levels

Note that no $AUC$ value in Table 6.3 fell below 0.85, and therefore the intervals specified in the first column of Table 6.4 were limited to operands greater than 0.85. It should be apparent that given a different application or experiment, the specified ranges could be adjusted and/or extended in a similar fashion depending on the results. Also, the confidence levels (in the second column of Table 6.4) could be allocated differently, according to the application to which the scheme was being applied.

Adopting a traditional statistical approach, a threshold can be determined by subtracting the confidence level from the mean—in our case, the $TPmean$—(Zikmund, 1997). With the $TPmean$ values and the confidence levels now available, a decision threshold could be determined for each training group member. In each case, this was a simple matter of subtracting their confidence level from their $TPmean$, and applying successive ($fpr$,$tpr$) pairs to a decision criteria at successive AD values. Recall that the AD value (adjustable determinant value) was used in determining each operating point in an ROC curve.

It is important to understand that if the confidence level increases in magnitude, a greater quantity is subtracted from the $TPmean$ (which lowers the AD value by a larger amount). That is, higher confidence levels actually indicate lower confidence[6].

[6]Though this may seem counter-intuitive, it is necessary because if the AUC is low—indicating poorer performance—then a greater quantity should be subtracted from the $TPmean$ to lower the AD value, thus indicating lower confidence.

As demonstrated in Algorithm 6.3, the task then was to determine which ($fpr$,$tpr$) pair met the criteria for selection of the decision threshold (i.e. OOP). This entailed successively applying all pairs of operating points to the decision criteria. If the decision criteria was met (Step 2 Algorithm 6.3), the current $fpr$ value was stored along with the index number of its occurrence in the array storing all $fpr$ values.

To meet the decision criteria, the current $fpr$ had to be smaller than the previously stored smallest $fpr$, and the current $tpr$ had to be greater than the previously determined confidence value. Thus at the end of iteration, the operating point with lowest $far$ (with an acceptable $frr$) was obtained.

**Algorithm 6.3** Calculation of Decision Threshold

Let $tpr$ be an array containing the stored True Positive values obtained as per Algorithm 6.1.
Let $fpr$ be an array containing the stored False Positive values obtained as per Algorithm 6.1.
Let $length \leftarrow 101$ be the length of the two arrays $tpr$ and $fpr$.
Let $ADV \leftarrow (TPmean - Confidence\ level)$ be the adjustable determinant value used in the selection criteria.
Let $smallestFpr \leftarrow 1.0$ be a variable for storing successive smallest $fpr$ value.
Let $count$ be a variable for storing the array index corresponding to the smallest $fpr$ value.
Let $i$ be a loop control variable to access successive elements of $tpr$ and $fpr$.

1: **For** $i \leftarrow 0$ **to** $length$ **do**
2: **If** $fpr[i] < smallestFpr$ **AND** $tpr[i] > ADV$
3: $smallestFpr \leftarrow fpr[i]$
4: $count \leftarrow i$
5: **End If**
6: **End** $i$ **for**

Once the decision threshold (OOP) was determined as demonstrated in Algorithm 6.3, the stored index number (for that OOP) was used to extract its corresponding $(fpr, tpr)$ coordinate pair. This coordinate pair was therefore the operating point that demonstrated best classifier performance for authentication purposes[7].

The following sections present tables with the relevant data for the $AUC$ values, the $TPmean$ values, the confidence levels, and the decision threshold for each participant, for all phases of the experiment. Also presented are the performance metrics ($far$ and $frr$) at the determined thresholds.

[7]With the $far$ being equivalent to the $fpr$ and the $frr$ being the compliment of the $tpr$, the performance metrics were thus obtained.

## 6.4 Results

In this section, the results are presented for the keystroke dynamics (section 6.4.1), fingerprint recognition (section 6.4.2), and feature level data fusion (section 6.4.3) phases of the experiment. Results presented for the data fusion phase cover the two paradigms—complimentary and cooperative—discussed in Chapter 5. For the cooperative paradigm, the results are presented for the four stages (i.e. at the four percentages) described in section 5.6.3.

Findings are demonstrated by the use of tables, with two tables being presented for each of the phases, paradigms, and stages of the experiment. Tables 6.5, 6.7, 6.9, 6.11, 6.13, 6.15, 6.17 are presented to demonstrate the statistics used in the determination of the decision threshold (at which results were determined and then presented); this is done for completeness. Each table presents the relevant data pertaining to each training group member (listed by row) and the following column format:

1. Participant number.
2. The area under the ROC curve—to 8 decimal places—refer sections 6.2.2.2 and 6.3.3.
3. The true positive mean—to 8 decimal places—refer section 6.3.4.
4. The confidence level—to 4 decimal places—refer section 6.3.4.
5. The adjustable determinant value—to 8 decimal places—refer sections 6.2.2.1 and 6.3.4.
6. The decision threshold—to 2 decimal places——refer sections 6.2.2.3 and 6.3.4.

Tables 6.6, 6.8, 6.10, 6.12, 6.14, 6.16, 6.18 are presented to demonstrate the results achieved at the determined decision threshold. Each table presents the relevant results pertaining to each training group member (listed by row) and the following column format:

1. Participant number.
2. The True Positive Rate—to 2 decimal places—refer section 6.2.1.

3. The False Rejection Rate—to 2 decimal places—refer section 6.2.1.

4. The Number of False Rejections.

5. The False Positive Rate—to 8 decimal places—refer section 6.2.1.

6. The Number of False Acceptances.

The last two rows of each table present the average and standard deviation for relevant columns (if applicable).

The following sections present the tabled statistics and results in the following order:

- Section 6.4.1 presents the keystroke dynamics statistics in Table 6.5 (page 345) and the results achieved in Table 6.6 (page 346).
- Section 6.4.2 presents the fingerprint recognition statistics in Table 6.7 (page 351) and the results achieved in Table 6.8 (page 352).
- Section 6.4.3 presents data fusion outcomes for both paradigms—complimentary and cooperative:
  - Section 6.4.3.1 presents the complimentary data fusion statistics in Table 6.9 (page 353) and the results achieved in Table 6.10 (page 354).
  - Section 6.4.3.2 presents cooperative data fusion outcomes for the following four stages:
    * Stage 1 statistics (i.e. 40% of the available data) in Table 6.11 (page 355) and the results achieved in Table 6.12 (page 356).
    * Stage 2 statistics (i.e. 50% of the available data) in Table 6.13 (page 357) and the results achieved in Table 6.14 (page 358).
    * Stage 3 statistics (i.e. 60% of the available data) in Table 6.15 (page 359) and the results achieved in Table 6.16 (page 360).
    * Stage 4 statistics (i.e. 70% of the available data) in Table 6.17 (page 361) and the results achieved in Table 6.18 (page 362).

### 6.4.1 Keystroke Dynamics (Phase 1)

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 0.99135190 | 0.98927652 | 0.0500 | 0.93927652 | 0.96 |
| 2 | 0.95659857 | 0.95334717 | 0.0500 | 0.90334717 | 0.97 |
| 3 | 0.88062905 | 0.85889453 | 0.0750 | 0.78389453 | 0.86 |
| 5 | 0.99785381 | 0.99474991 | 0.0500 | 0.94474991 | 0.98 |
| 7 | 0.99322429 | 0.98808178 | 0.0500 | 0.93808178 | 0.99 |
| 9 | 0.96157905 | 0.96274334 | 0.0500 | 0.91274334 | 0.95 |
| 12 | 0.91994667 | 0.93981623 | 0.0625 | 0.87731623 | 0.97 |
| 14 | 0.99516429 | 0.98649916 | 0.0500 | 0.93649916 | 0.93 |
| 16 | 0.99442571 | 0.97118196 | 0.0500 | 0.92118195 | 0.95 |
| 18 | 0.99973810 | 0.99862410 | 0.0500 | 0.94862410 | 0.99 |
| 20 | 0.99369381 | 0.97664731 | 0.0500 | 0.92664731 | 0.98 |
| 21 | 0.98268905 | 0.96702640 | 0.0500 | 0.91702640 | 0.97 |
| 23 | 0.97532000 | 0.95544168 | 0.0500 | 0.90544168 | 0.96 |
| 24 | 0.95277429 | 0.95765571 | 0.0500 | 0.90765571 | 0.94 |
| 25 | 0.99058143 | 0.95367530 | 0.0500 | 0.90367530 | 0.96 |
| 27 | 0.99949143 | 0.99109513 | 0.0500 | 0.94109513 | 0.96 |
| 29 | 0.94121762 | 0.93243909 | 0.0625 | 0.86993909 | 0.92 |
| 32 | 0.97951571 | 0.94305124 | 0.0500 | 0.89305124 | 0.98 |
| 34 | 0.99918000 | 0.98004077 | 0.0500 | 0.93004077 | 0.95 |
| 36 | 0.99211571 | 0.98287234 | 0.0500 | 0.93287234 | 0.96 |
| 38 | 0.99692810 | 0.99031712 | 0.0500 | 0.94031712 | 0.99 |
| 40 | 0.99834524 | 0.99927547 | 0.0500 | 0.94927547 | 0.99 |
| 41 | 0.98708762 | 0.95652380 | 0.0500 | 0.90652380 | 0.97 |
| 43 | 0.86263333 | 0.86382857 | 0.0750 | 0.78882857 | 0.99 |
| 45 | 0.98877381 | 0.99863071 | 0.0500 | 0.94863071 | 0.99 |
| 46 | 0.99621143 | 0.99234570 | 0.0500 | 0.94234570 | 0.96 |
| 47 | 0.98223762 | 0.95406353 | 0.0500 | 0.90406353 | 0.89 |
| 49 | 0.94541667 | 0.97502202 | 0.0625 | 0.91252202 | 0.94 |
| 52 | 0.99994952 | 0.99833596 | 0.0500 | 0.94833596 | 0.98 |
| 54 | 0.98558571 | 0.96455751 | 0.0500 | 0.91455750 | 0.94 |
| 56 | 0.97548333 | 0.91711252 | 0.0500 | 0.86711252 | 0.92 |
| 58 | 0.95181810 | 0.91879129 | 0.0500 | 0.86879129 | 0.93 |
| 60 | 0.99755619 | 0.97835570 | 0.0500 | 0.92835570 | 0.97 |
| 61 | 0.94248857 | 0.84054680 | 0.0625 | 0.77804680 | 0.95 |
| 63 | 0.95563619 | 0.93299954 | 0.0500 | 0.88299954 | 0.89 |
| 65 | 0.97971571 | 0.89510313 | 0.0500 | 0.84510313 | 0.88 |
| 67 | 0.96299048 | 0.90508670 | 0.0500 | 0.85508670 | 0.88 |
| 68 | 0.97225952 | 0.94748348 | 0.0500 | 0.89748348 | 0.98 |
| 69 | 0.99225190 | 0.97082952 | 0.0500 | 0.92082952 | 0.95 |
| 72 | 0.99848952 | 0.99350247 | 0.0500 | 0.94350247 | 0.98 |
| 74 | 0.86826095 | 0.84537021 | 0.0750 | 0.77037021 | 0.82 |
| 76 | 0.98762333 | 0.97372189 | 0.0500 | 0.92372189 | 0.94 |
| 78 | 0.97809667 | 0.94457460 | 0.0500 | 0.89457460 | 0.95 |
| 80 | 0.97258619 | 0.95375459 | 0.0500 | 0.90375459 | 0.97 |
| 81 | 0.95439667 | 0.89733494 | 0.0500 | 0.84733494 | 0.98 |
| 83 | 0.98380381 | 0.96253737 | 0.0500 | 0.91253737 | 0.97 |
| 85 | 1.00000000 | 0.99663067 | 0.0500 | 0.94663067 | 0.88 |
| 87 | 0.95541905 | 0.92590986 | 0.0500 | 0.87590986 | 0.94 |
| 89 | 0.98358381 | 0.98422130 | 0.0500 | 0.93422130 | 0.97 |
| 90 | 0.97026905 | 0.91696189 | 0.0500 | 0.86696189 | 0.91 |
| **Average** | 0.97245977 | 0.95353777 | – | 0.90103777 | 0.9486 |
| **SD** | 0.03232844 | 0.04117834 | – | 0.04586111 | 0.03811958 |

Table 6.5: Keystroke Dynamics Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 0.94 | 0.06 | 6 | 0.01561905 | 164 |
| 2 | 0.91 | 0.09 | 9 | 0.06533333 | 686 |
| 3 | 0.79 | 0.21 | 21 | 0.12600000 | 1323 |
| 5 | 0.95 | 0.05 | 5 | 0.00466667 | 49 |
| 7 | 0.96 | 0.04 | 4 | 0.00247619 | 26 |
| 9 | 0.93 | 0.07 | 7 | 0.05200000 | 546 |
| 12 | 0.88 | 0.12 | 12 | 0.10628571 | 1116 |
| 14 | 0.94 | 0.06 | 6 | 0.01342857 | 141 |
| 16 | 0.95 | 0.05 | 5 | 0.00009524 | 1 |
| 18 | 0.97 | 0.03 | 3 | 0.00047619 | 5 |
| 20 | 0.94 | 0.06 | 6 | 0.01009524 | 106 |
| 21 | 0.92 | 0.08 | 8 | 0.01742857 | 183 |
| 23 | 0.91 | 0.09 | 9 | 0.01409524 | 148 |
| 24 | 0.91 | 0.09 | 9 | 0.06952381 | 730 |
| 25 | 0.91 | 0.09 | 9 | 0.00142857 | 15 |
| 27 | 0.95 | 0.05 | 5 | 0.00085714 | 9 |
| 29 | 0.87 | 0.13 | 13 | 0.08104762 | 851 |
| 32 | 0.90 | 0.10 | 10 | 0.00104762 | 11 |
| 34 | 0.96 | 0.04 | 4 | 0.00123810 | 13 |
| 36 | 0.95 | 0.05 | 5 | 0.01171429 | 123 |
| 38 | 0.96 | 0.04 | 4 | 0.00542857 | 57 |
| 40 | 0.99 | 0.01 | 1 | 0.00323810 | 34 |
| 41 | 0.91 | 0.09 | 9 | 0.01028571 | 108 |
| 43 | 0.81 | 0.19 | 19 | 0.15019048 | 1577 |
| 45 | 0.98 | 0.02 | 2 | 0.02190476 | 230 |
| 46 | 0.95 | 0.05 | 5 | 0.01200000 | 126 |
| 47 | 0.92 | 0.08 | 8 | 0.02171429 | 228 |
| 49 | 0.94 | 0.06 | 6 | 0.08647619 | 908 |
| 52 | 0.99 | 0.01 | 1 | 0.00009524 | 1 |
| 54 | 0.92 | 0.08 | 8 | 0.01152381 | 121 |
| 56 | 0.87 | 0.13 | 13 | 0.00552381 | 58 |
| 58 | 0.87 | 0.13 | 13 | 0.05095238 | 535 |
| 60 | 0.93 | 0.07 | 7 | 0.00114286 | 12 |
| 61 | 0.78 | 0.22 | 22 | 0.00866667 | 91 |
| 63 | 0.89 | 0.11 | 11 | 0.06076190 | 638 |
| 65 | 0.85 | 0.15 | 15 | 0.00400000 | 42 |
| 67 | 0.86 | 0.14 | 14 | 0.02923810 | 307 |
| 68 | 0.90 | 0.10 | 10 | 0.01838095 | 193 |
| 69 | 0.94 | 0.06 | 6 | 0.01190476 | 125 |
| 72 | 0.95 | 0.05 | 5 | 0.00266667 | 28 |
| 74 | 0.78 | 0.22 | 22 | 0.14400000 | 1512 |
| 76 | 0.94 | 0.06 | 6 | 0.01380952 | 145 |
| 78 | 0.92 | 0.08 | 8 | 0.01104762 | 116 |
| 80 | 0.91 | 0.09 | 9 | 0.02304762 | 242 |
| 81 | 0.86 | 0.14 | 14 | 0.01342857 | 141 |
| 83 | 0.93 | 0.07 | 7 | 0.00438095 | 46 |
| 85 | 1.00 | 0.00 | 0 | 0.00000000 | 0 |
| 87 | 0.88 | 0.12 | 12 | 0.02695238 | 283 |
| 89 | 0.95 | 0.05 | 5 | 0.02800000 | 294 |
| 90 | 0.87 | 0.13 | 13 | 0.00742857 | 78 |
| **Average** | 0.9138 | 0.0862 | 8.62 | 0.02766095 | 290.44 |
| **SD** | 0.051504 | 0.051504 | – | 0.03806474 | – |

Table 6.6: Keystroke Dynamics Results

As there were 50 training group members, there were 50 ROC graphs attainable. Figures 6.7, 6.8, and 6.9 are presented here to illustrate an example of the best, average, and worst performances for the keystroke dynamics phase of the experiment. More examples are provided in Appendix D.

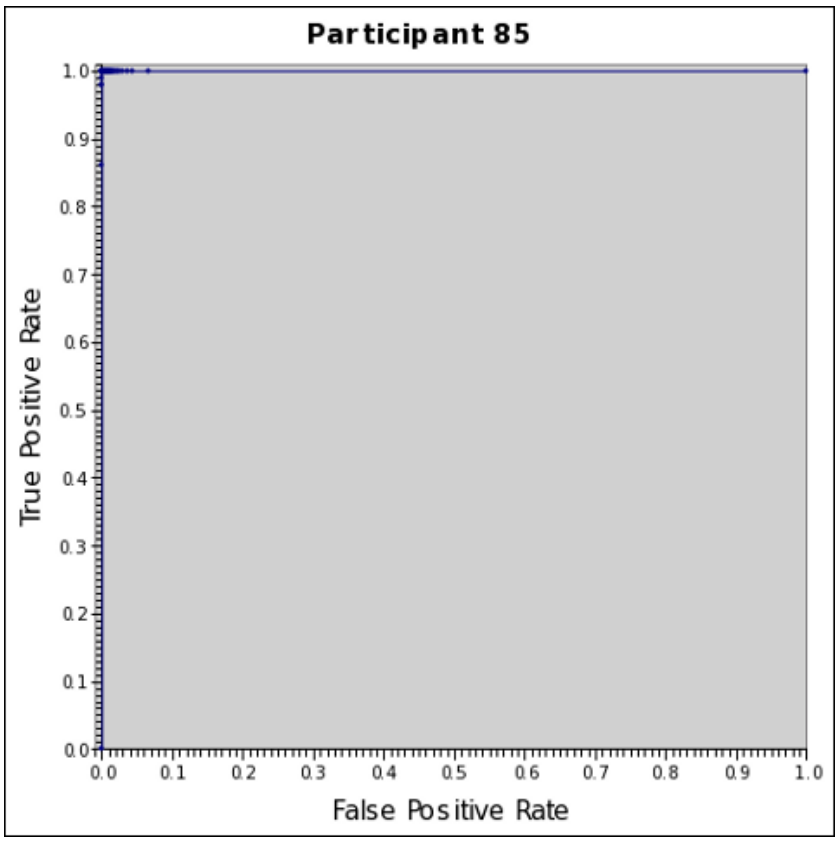


Figure 6.7: Example Demonstrating Best Classification

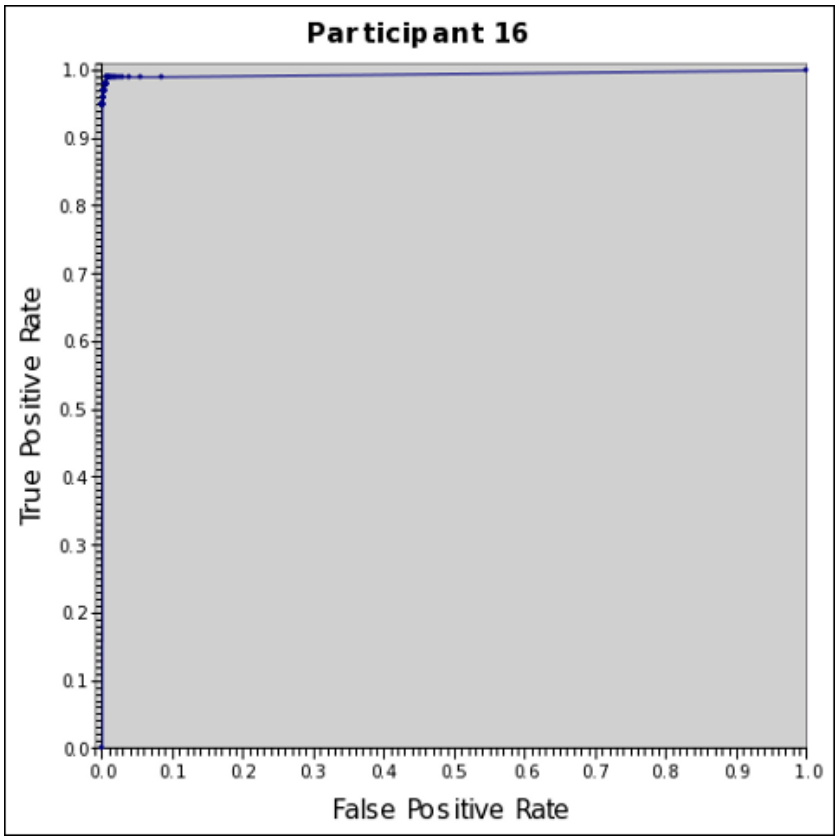


Figure 6.8: Example Demonstrating Average Classification

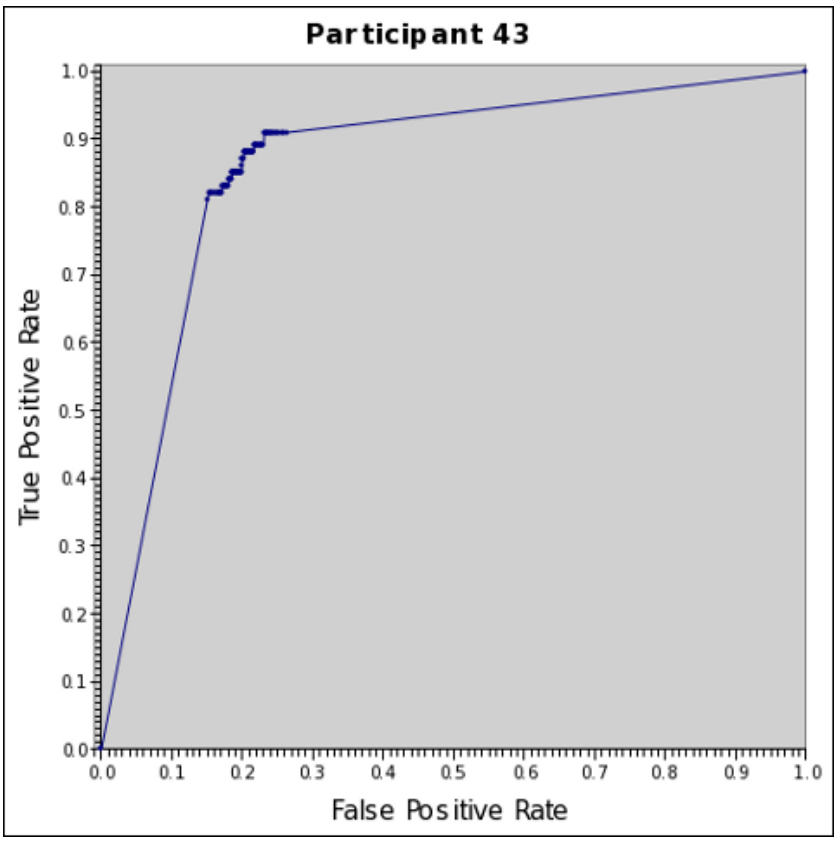


Figure 6.9: Example Demonstrating Worst Classification

Figure 6.10 presents the $far$ and $frr$ results (coupled together, with the $frr$ on the left of the $far$) for the keystroke dynamics phase, for all training group members. Participant numbers are listed along the x axis, and the performance measurement or rate on the y axis. Note, the performance measurement ranges from (0.0) to (0.25). This provides a indication of the accuracy obtained for the keystroke dynamics experiment.

When reviewing Figure 6.10, it should be remembered that the results achieved for the $frr$ (for each training group member) were obtained by applying 100 genuine test samples to their trained ANN. Thus the $frr$ results demonstrate coarse granularity. The $far$ results were obtained by applying 10,500 impostor test samples to a training group member's trained ANN, resulting in a much finer granularity.

Figure 6.10 demonstrates that 39 of the 50 training group members achieved a $far$ less than 0.05, with the remaining 11 greater than or equal to 0.05 (the largest being approximately 0.15); only 6 members achieved a $frr$ less than 0.05. Figure 6.10 also shows that the $frr$ results were higher than the $far$ results in all cases.

It seems appropriate to note here, that a figure (similar to Figure 6.10) is **not** presented for any other phase of the experiment. This is because the results achieved in all other phases did not warrant such a figure. The $frr$ and $far$ for all other phases of the experiment were so low that they would not convey any meaningful information, if presented in a similar figure.

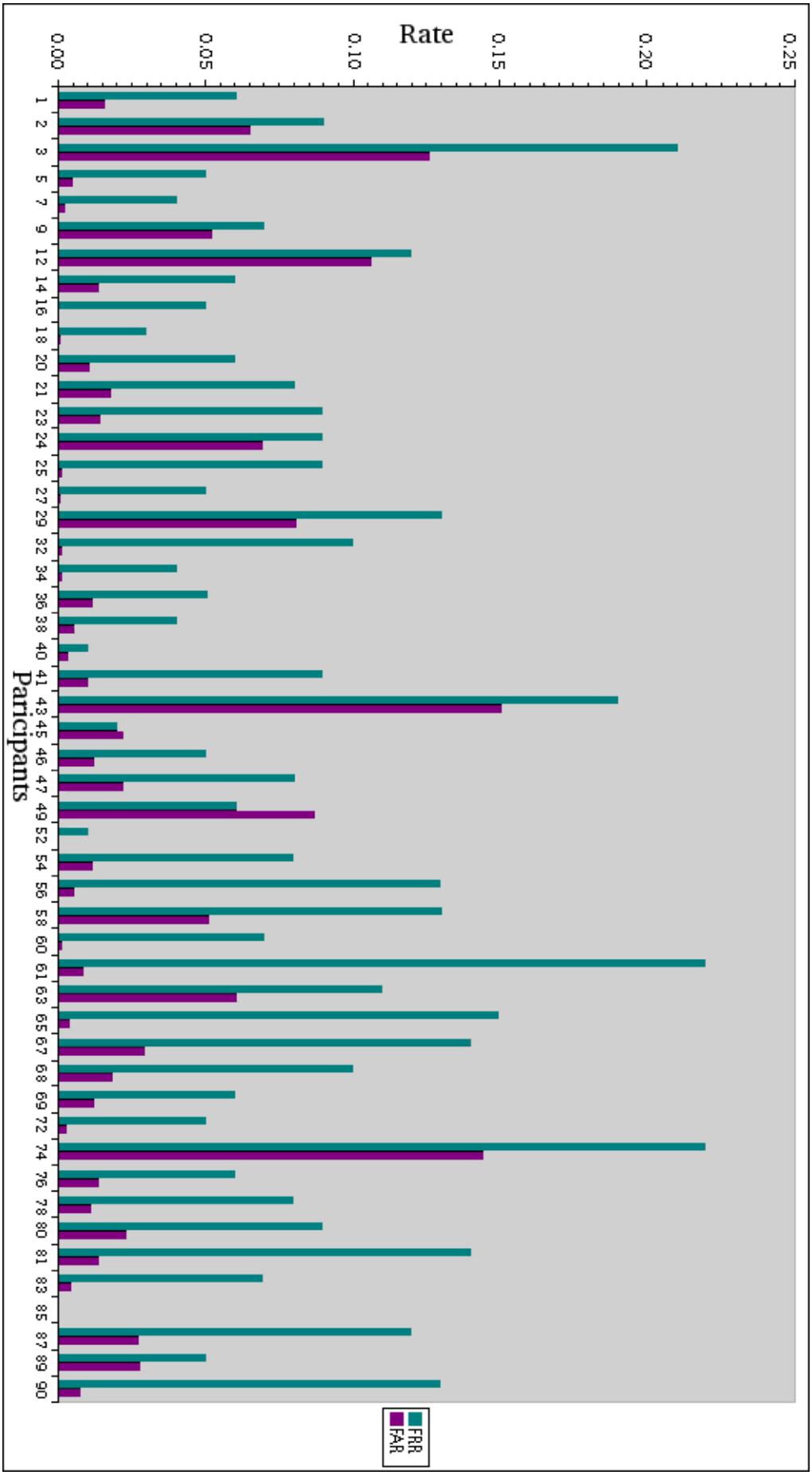


Figure 6.10: Keystroke Dynamics

### 6.4.2 Fingerprint Recognition (Phase 2)

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99996294 | 0.0500 | 0.94996294 | 0.84 |
| 2 | 1 | 0.99996188 | 0.0500 | 0.94996188 | 0.95 |
| 3 | 1 | 0.99802990 | 0.0500 | 0.94802990 | 0.61 |
| 5 | 1 | 0.98105692 | 0.0500 | 0.93105692 | 0.02 |
| 7 | 1 | 0.99997253 | 0.0500 | 0.94997252 | 0.22 |
| 9 | 1 | 0.99998120 | 0.0500 | 0.94998120 | 0.20 |
| 12 | 1 | 0.99997238 | 0.0500 | 0.94997238 | 0.16 |
| 14 | 1 | 0.99976722 | 0.0500 | 0.94976722 | 0.39 |
| 16 | 1 | 0.99887867 | 0.0500 | 0.94887867 | 0.09 |
| 18 | 1 | 0.99969411 | 0.0500 | 0.94969411 | 0.64 |
| 20 | 1 | 0.99995251 | 0.0500 | 0.94995251 | 0.94 |
| 21 | 1 | 0.99995280 | 0.0500 | 0.94995279 | 0.20 |
| 23 | 0.99999200 | 0.99681243 | 0.0500 | 0.94681243 | 0.97 |
| 24 | 1 | 0.99949265 | 0.0500 | 0.94949264 | 0.81 |
| 25 | 1 | 0.99968960 | 0.0500 | 0.94968960 | 0.94 |
| 27 | 1 | 0.99989653 | 0.0500 | 0.94989653 | 0.85 |
| 29 | 1 | 0.99997006 | 0.0500 | 0.94997006 | 0.04 |
| 32 | 1 | 0.99993341 | 0.0500 | 0.94993341 | 0.03 |
| 34 | 1 | 0.99980600 | 0.0500 | 0.94980600 | 0.59 |
| 36 | 1 | 0.99996920 | 0.0500 | 0.94996920 | 0.67 |
| 38 | 1 | 0.99799319 | 0.0500 | 0.94799319 | 0.35 |
| 40 | 1 | 0.99994016 | 0.0500 | 0.94994016 | 0.05 |
| 41 | 1 | 0.99996204 | 0.0500 | 0.94996204 | 0.05 |
| 43 | 1 | 0.99993914 | 0.0500 | 0.94993914 | 0.03 |
| 45 | 1 | 0.99988545 | 0.0500 | 0.94988545 | 0.08 |
| 46 | 1 | 0.99994543 | 0.0500 | 0.94994543 | 0.90 |
| 47 | 1 | 0.99988648 | 0.0500 | 0.94988648 | 0.11 |
| 49 | 1 | 0.99985759 | 0.0500 | 0.94985759 | 0.95 |
| 52 | 1 | 0.99988577 | 0.0500 | 0.94988577 | 0.94 |
| 54 | 1 | 0.99997126 | 0.0500 | 0.94997126 | 0.16 |
| 56 | 1 | 0.99995320 | 0.0500 | 0.94995320 | 0.66 |
| 58 | 1 | 0.99990624 | 0.0500 | 0.94990624 | 0.93 |
| 60 | 1 | 0.99997203 | 0.0500 | 0.94997203 | 0.10 |
| 61 | 1 | 0.99997752 | 0.0500 | 0.94997752 | 0.02 |
| 63 | 1 | 0.99993195 | 0.0500 | 0.94993195 | 0.93 |
| 65 | 1 | 0.99995170 | 0.0500 | 0.94995170 | 0.30 |
| 67 | 1 | 0.99994074 | 0.0500 | 0.94994074 | 0.22 |
| 68 | 1 | 0.99997243 | 0.0500 | 0.94997243 | 0.87 |
| 69 | 1 | 0.99995608 | 0.0500 | 0.94995608 | 0.04 |
| 72 | 1 | 0.99986472 | 0.0500 | 0.94986472 | 0.01 |
| 74 | 0.99999600 | 0.99571402 | 0.0500 | 0.94571402 | 0.93 |
| 76 | 0.99999900 | 0.99910068 | 0.0500 | 0.94910068 | 0.98 |
| 78 | 1 | 0.99997427 | 0.0500 | 0.94997427 | 0.75 |
| 80 | 1 | 0.98823433 | 0.0500 | 0.93823433 | 0.02 |
| 81 | 1 | 0.95769181 | 0.0500 | 0.90769181 | 0.41 |
| 83 | 1 | 0.99989038 | 0.0500 | 0.94989038 | 0.41 |
| 85 | 1 | 0.99995190 | 0.0500 | 0.94995190 | 0.19 |
| 87 | 1 | 0.99995996 | 0.0500 | 0.94995996 | 0.93 |
| 89 | 1 | 0.99995232 | 0.0500 | 0.94995232 | 0.03 |
| 90 | 1 | 0.99069618 | 0.0500 | 0.94069618 | 0.02 |
| **Average** | 0.99999974 | 0.99801224 | – | 0.94801224 | 0.4506 |
| **SD** | 0.00000123 | 0.00672410 | – | 0.00672410 | 0.37543667 |

Table 6.7: Fingerprint Recognition Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.95 | 0.05 | 5 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 0.95 | 0.05 | 5 | 0 | 0 |
| 76 | 0.99 | 0.01 | 1 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| **Average** | 0.9978 | 0.0022 | 0.22 | 0 | 0 |
| **SD** | 0.009957 | 0.009957 | – | 0 | – |

Table 6.8: Fingerprint Recognition Results

### 6.4.3 Data Fusion (Phase 3)

#### 6.4.3.1 Complementary Data Fusion

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99993349 | 0.0500 | 0.94993349 | 0.78 |
| 2 | 1 | 0.99993526 | 0.0500 | 0.94993526 | 0.88 |
| 3 | 1 | 0.99777526 | 0.0500 | 0.94777526 | 0.28 |
| 5 | 1 | 0.99968306 | 0.0500 | 0.94968306 | 0.08 |
| 7 | 1 | 0.99981626 | 0.0500 | 0.94981626 | 0.08 |
| 9 | 1 | 0.99996180 | 0.0500 | 0.94996180 | 0.25 |
| 12 | 1 | 0.99992290 | 0.0500 | 0.94992290 | 0.12 |
| 14 | 1 | 0.99961334 | 0.0500 | 0.94961334 | 0.36 |
| 16 | 1 | 0.99975571 | 0.0500 | 0.94975571 | 0.10 |
| 18 | 1 | 0.99979385 | 0.0500 | 0.94979385 | 0.92 |
| 20 | 1 | 0.99980667 | 0.0500 | 0.94980667 | 0.93 |
| 21 | 1 | 0.99997503 | 0.0500 | 0.94997503 | 0.53 |
| 23 | 0.99999429 | 0.99489361 | 0.0500 | 0.94489361 | 0.91 |
| 24 | 1 | 0.99843420 | 0.0500 | 0.94843420 | 0.11 |
| 25 | 1 | 0.99962849 | 0.0500 | 0.94962849 | 0.88 |
| 27 | 1 | 0.99986136 | 0.0500 | 0.94986136 | 0.94 |
| 29 | 1 | 0.99999791 | 0.0500 | 0.94999791 | 0.29 |
| 32 | 1 | 0.99987604 | 0.0500 | 0.94987604 | 0.05 |
| 34 | 1 | 0.99965469 | 0.0500 | 0.94965469 | 0.45 |
| 36 | 1 | 0.99997427 | 0.0500 | 0.94997427 | 0.95 |
| 38 | 1 | 0.99908305 | 0.0500 | 0.94908304 | 0.13 |
| 40 | 1 | 0.99996581 | 0.0500 | 0.94996581 | 0.29 |
| 41 | 1 | 0.99977032 | 0.0500 | 0.94977032 | 0.12 |
| 43 | 1 | 0.99995303 | 0.0500 | 0.94995303 | 0.04 |
| 45 | 1 | 0.99994619 | 0.0500 | 0.94994619 | 0.04 |
| 46 | 1 | 0.99995954 | 0.0500 | 0.94995954 | 0.69 |
| 47 | 1 | 0.99984224 | 0.0500 | 0.94984224 | 0.11 |
| 49 | 1 | 0.99988235 | 0.0500 | 0.94988235 | 0.79 |
| 52 | 1 | 0.99974277 | 0.0500 | 0.94974276 | 0.94 |
| 54 | 1 | 0.99996927 | 0.0500 | 0.94996927 | 0.09 |
| 56 | 1 | 0.99997403 | 0.0500 | 0.94997403 | 0.66 |
| 58 | 1 | 0.99969609 | 0.0500 | 0.94969609 | 0.92 |
| 60 | 1 | 0.99996602 | 0.0500 | 0.94996602 | 0.23 |
| 61 | 1 | 0.99967599 | 0.0500 | 0.94967599 | 0.03 |
| 63 | 1 | 0.99990192 | 0.0500 | 0.94990192 | 0.77 |
| 65 | 1 | 0.99990294 | 0.0500 | 0.94990294 | 0.05 |
| 67 | 1 | 0.99976500 | 0.0500 | 0.94976500 | 0.24 |
| 68 | 1 | 0.99986011 | 0.0500 | 0.94986011 | 0.89 |
| 69 | 1 | 0.99990128 | 0.0500 | 0.94990128 | 0.02 |
| 72 | 1 | 0.99992978 | 0.0500 | 0.94992978 | 0.02 |
| 74 | 1 | 0.99427257 | 0.0500 | 0.94427257 | 0.85 |
| 76 | 1 | 0.99594715 | 0.0500 | 0.94594715 | 0.10 |
| 78 | 1 | 0.99997000 | 0.0500 | 0.94997000 | 0.55 |
| 80 | 1 | 0.99866941 | 0.0500 | 0.94866941 | 0.06 |
| 81 | 1 | 0.94646377 | 0.0500 | 0.89646377 | 0.17 |
| 83 | 1 | 0.99981049 | 0.0500 | 0.94981049 | 0.27 |
| 85 | 1 | 0.99996275 | 0.0500 | 0.94996275 | 0.11 |
| 87 | 1 | 0.99978510 | 0.0500 | 0.94978510 | 0.95 |
| 89 | 1 | 0.99992310 | 0.0500 | 0.94992310 | 0.02 |
| 90 | 1 | 0.99125975 | 0.0500 | 0.94125975 | 0.02 |
| **Average** | 0.99999989 | 0.99822090 | – | 0.94822090 | 0.4012 |
| **SD** | 0.00000081 | 0.00765436 | – | 0.00765436 | 0.35812801 |

Table 6.9: Complementary Data Fusion Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.98 | 0.02 | 2 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| **Average** | 0.9996 | 0.0004 | 0.04 | 0 | 0 |
| **SD** | 0.002828 | 0.002828 | – | 0 | – |

Table 6.10: Complementary Data Fusion Results

#### 6.4.3.2 Cooperative Data Fusion

### Stage 1 (40%)

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99991754 | 0.0500 | 0.94991754 | 0.78 |
| 2 | 1 | 0.99988399 | 0.0500 | 0.94988399 | 0.94 |
| 3 | 1 | 0.99436450 | 0.0500 | 0.94436450 | 0.04 |
| 5 | 1 | 0.99858813 | 0.0500 | 0.94858813 | 0.08 |
| 7 | 1 | 0.99995643 | 0.0500 | 0.94995643 | 0.27 |
| 9 | 1 | 0.99999629 | 0.0500 | 0.94999629 | 0.20 |
| 12 | 1 | 0.99997473 | 0.0500 | 0.94997473 | 0.18 |
| 14 | 1 | 0.99998118 | 0.0500 | 0.94998118 | 0.94 |
| 16 | 1 | 0.99956927 | 0.0500 | 0.94956927 | 0.03 |
| 18 | 1 | 0.99987230 | 0.0500 | 0.94987230 | 0.93 |
| 20 | 1 | 0.99997892 | 0.0500 | 0.94997892 | 0.43 |
| 21 | 1 | 0.99997063 | 0.0500 | 0.94997063 | 0.65 |
| 23 | 0.99998095 | 0.98141660 | 0.0500 | 0.93141659 | 0.77 |
| 24 | 1 | 0.99906967 | 0.0500 | 0.94906967 | 0.40 |
| 25 | 0.99995238 | 0.99994703 | 0.0500 | 0.94994703 | 0.80 |
| 27 | 1 | 0.99996674 | 0.0500 | 0.94996674 | 0.72 |
| 29 | 1 | 0.99999984 | 0.0500 | 0.94999984 | 0.48 |
| 32 | 1 | 0.99990204 | 0.0500 | 0.94990204 | 0.07 |
| 34 | 1 | 0.99987635 | 0.0500 | 0.94987635 | 0.42 |
| 36 | 1 | 0.99997806 | 0.0500 | 0.94997806 | 0.51 |
| 38 | 1 | 0.99997937 | 0.0500 | 0.94997937 | 0.90 |
| 40 | 1 | 0.99996609 | 0.0500 | 0.94996609 | 0.11 |
| 41 | 1 | 0.99990079 | 0.0500 | 0.94990079 | 0.03 |
| 43 | 1 | 0.99994124 | 0.0500 | 0.94994124 | 0.05 |
| 45 | 1 | 0.99991208 | 0.0500 | 0.94991207 | 0.29 |
| 46 | 1 | 0.99998315 | 0.0500 | 0.94998315 | 0.98 |
| 47 | 1 | 0.99996134 | 0.0500 | 0.94996134 | 0.50 |
| 49 | 1 | 0.99992628 | 0.0500 | 0.94992628 | 0.87 |
| 52 | 1 | 0.99993853 | 0.0500 | 0.94993853 | 0.92 |
| 54 | 1 | 0.99996554 | 0.0500 | 0.94996554 | 0.04 |
| 56 | 1 | 0.99995322 | 0.0500 | 0.94995322 | 0.57 |
| 58 | 1 | 0.99993582 | 0.0500 | 0.94993582 | 0.94 |
| 60 | 1 | 0.99997868 | 0.0500 | 0.94997868 | 0.31 |
| 61 | 1 | 0.99996597 | 0.0500 | 0.94996597 | 0.02 |
| 63 | 1 | 0.99994824 | 0.0500 | 0.94994824 | 0.86 |
| 65 | 1 | 0.99996813 | 0.0500 | 0.94996813 | 0.07 |
| 67 | 1 | 0.99995870 | 0.0500 | 0.94995870 | 0.11 |
| 68 | 1 | 0.99996510 | 0.0500 | 0.94996510 | 0.95 |
| 69 | 1 | 0.99996257 | 0.0500 | 0.94996257 | 0.01 |
| 72 | 1 | 0.99997178 | 0.0500 | 0.94997178 | 0.02 |
| 74 | 1 | 0.99398993 | 0.0500 | 0.94398993 | 0.74 |
| 76 | 1 | 0.99813954 | 0.0500 | 0.94813954 | 0.18 |
| 78 | 1 | 0.99997119 | 0.0500 | 0.94997119 | 0.61 |
| 80 | 1 | 0.97866954 | 0.0500 | 0.92866954 | 0.29 |
| 81 | 1 | 0.95296646 | 0.0500 | 0.90296646 | 0.21 |
| 83 | 1 | 0.99994713 | 0.0500 | 0.94994713 | 0.25 |
| 85 | 1 | 0.99992489 | 0.0500 | 0.94992489 | 0.03 |
| 87 | 0.99995238 | 0.99996281 | 0.0500 | 0.94996281 | 0.49 |
| 89 | 1 | 0.99997073 | 0.0500 | 0.94997073 | 0.04 |
| 90 | 1 | 0.99238965 | 0.0500 | 0.94238965 | 0.14 |
| **Average** | 0.99999867 | 0.99774449 | – | 0.94774449 | 0.4234 |
| **SD** | 0.00000973 | 0.00768056 | – | 0.00768056 | 0.34155473 |

Table 6.11: Cooperative Data Fusion (40%) Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.98 | 0.02 | 2 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0.00009524 | 1 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0.00009524 | 1 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| Average | 0.9996 | 0.0004 | 0.04 | 0.00000381 | 0.04 |
| SD | 0.002828 | 0.002828 | – | 0.00001885 | – |

Table 6.12: Cooperative Data Fusion (40%) Results

### Stage 2 (50%)

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99995281 | 0.0500 | 0.94995281 | 0.82 |
| 2 | 1 | 0.99993966 | 0.0500 | 0.94993966 | 0.88 |
| 3 | 1 | 0.99786012 | 0.0500 | 0.94786012 | 0.70 |
| 5 | 1 | 0.99999326 | 0.0500 | 0.94999326 | 0.67 |
| 7 | 1 | 0.99988564 | 0.0500 | 0.94988564 | 0.15 |
| 9 | 1 | 0.99997344 | 0.0500 | 0.94997344 | 0.09 |
| 12 | 1 | 0.99999695 | 0.0500 | 0.94999695 | 0.28 |
| 14 | 1 | 0.99998740 | 0.0500 | 0.94998740 | 0.96 |
| 16 | 1 | 0.99978117 | 0.0500 | 0.94978117 | 0.11 |
| 18 | 1 | 0.99990018 | 0.0500 | 0.94990018 | 0.95 |
| 20 | 1 | 0.99991915 | 0.0500 | 0.94991915 | 0.95 |
| 21 | 1 | 0.99998011 | 0.0500 | 0.94998011 | 0.24 |
| 23 | 1 | 0.99796202 | 0.0500 | 0.94796202 | 0.93 |
| 24 | 1 | 0.99897221 | 0.0500 | 0.94897221 | 0.02 |
| 25 | 1 | 0.99994968 | 0.0500 | 0.94994968 | 0.98 |
| 27 | 1 | 0.99993967 | 0.0500 | 0.94993967 | 0.68 |
| 29 | 1 | 0.99997148 | 0.0500 | 0.94997148 | 0.03 |
| 32 | 1 | 0.99991157 | 0.0500 | 0.94991157 | 0.06 |
| 34 | 1 | 0.99991700 | 0.0500 | 0.94991700 | 0.64 |
| 36 | 1 | 0.99997909 | 0.0500 | 0.94997909 | 0.47 |
| 38 | 1 | 0.99986791 | 0.0500 | 0.94986791 | 0.87 |
| 40 | 1 | 0.99998716 | 0.0500 | 0.94998716 | 0.08 |
| 41 | 1 | 0.99993588 | 0.0500 | 0.94993588 | 0.02 |
| 43 | 1 | 0.99995880 | 0.0500 | 0.94995880 | 0.06 |
| 45 | 1 | 0.99993733 | 0.0500 | 0.94993733 | 0.19 |
| 46 | 1 | 0.99997161 | 0.0500 | 0.94997161 | 0.93 |
| 47 | 1 | 0.99989028 | 0.0500 | 0.94989028 | 0.12 |
| 49 | 1 | 0.99994911 | 0.0500 | 0.94994911 | 0.88 |
| 52 | 1 | 0.99993010 | 0.0500 | 0.94993010 | 0.93 |
| 54 | 1 | 0.99997683 | 0.0500 | 0.94997683 | 0.04 |
| 56 | 1 | 0.99995208 | 0.0500 | 0.94995208 | 0.79 |
| 58 | 1 | 0.99990854 | 0.0500 | 0.94990854 | 0.85 |
| 60 | 1 | 0.99997807 | 0.0500 | 0.94997807 | 0.15 |
| 61 | 1 | 0.99996351 | 0.0500 | 0.94996351 | 0.03 |
| 63 | 1 | 0.9999384 | 0.0500 | 0.9499384 | 0.66 |
| 65 | 1 | 0.99997876 | 0.0500 | 0.94997876 | 0.14 |
| 67 | 1 | 0.99993852 | 0.0500 | 0.94993852 | 0.09 |
| 68 | 1 | 0.99996217 | 0.0500 | 0.94996216 | 0.95 |
| 69 | 1 | 0.99994770 | 0.0500 | 0.94994770 | 0.03 |
| 72 | 1 | 0.99996497 | 0.0500 | 0.94996497 | 0.02 |
| 74 | 1 | 0.99671857 | 0.0500 | 0.94671857 | 0.82 |
| 76 | 1 | 0.99903137 | 0.0500 | 0.94903136 | 0.26 |
| 78 | 1 | 0.99999157 | 0.0500 | 0.94999157 | 0.20 |
| 80 | 1 | 0.99675004 | 0.0500 | 0.94675004 | 0.05 |
| 81 | 0.99999714 | 0.99820160 | 0.0500 | 0.94820160 | 0.98 |
| 83 | 1 | 0.99995360 | 0.0500 | 0.94995360 | 0.14 |
| 85 | 1 | 0.99996115 | 0.0500 | 0.94996115 | 0.12 |
| 87 | 1 | 0.99998683 | 0.0500 | 0.94998683 | 0.92 |
| 89 | 1 | 0.99996138 | 0.0500 | 0.94996138 | 0.04 |
| 90 | 1 | 0.99462889 | 0.0500 | 0.94462889 | 0.13 |
| Average | 0.99999994 | 0.99955791 | – | 0.94955791 | 0.442 |
| SD | 0.00000040 | 0.00105515 | – | 0.00105515 | 0.38194988 |

Table 6.13: Cooperative Data Fusion (50%) Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 1 | 0 | 0 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 0.97 | 0.03 | 3 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| Average | 0.9994 | 0.0006 | 0.06 | 0 | 0 |
| SD | 0.004243 | 0.004243 | – | 0 | – |

Table 6.14: Cooperative Data Fusion (50%) Results

### Stage 3 (60%)

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99992793 | 0.0500 | 0.94992793 | 0.85 |
| 2 | 1 | 0.99996556 | 0.0500 | 0.94996556 | 0.89 |
| 3 | 1 | 0.99843240 | 0.0500 | 0.94843240 | 0.38 |
| 5 | 1 | 0.98928223 | 0.0500 | 0.93928223 | 0.03 |
| 7 | 1 | 0.99998868 | 0.0500 | 0.94998868 | 0.16 |
| 9 | 1 | 0.99996960 | 0.0500 | 0.94996960 | 0.10 |
| 12 | 1 | 0.99998107 | 0.0500 | 0.94998107 | 0.18 |
| 14 | 1 | 0.99982194 | 0.0500 | 0.94982194 | 0.69 |
| 16 | 1 | 0.99985554 | 0.0500 | 0.94985554 | 0.58 |
| 18 | 1 | 0.99995425 | 0.0500 | 0.94995425 | 0.93 |
| 20 | 1 | 0.99992648 | 0.0500 | 0.94992648 | 0.94 |
| 21 | 1 | 0.99993923 | 0.0500 | 0.94993923 | 0.20 |
| 23 | 0.99998571 | 0.99222820 | 0.0500 | 0.94222819 | 0.95 |
| 24 | 1 | 0.99828443 | 0.0500 | 0.94828443 | 0.25 |
| 25 | 1 | 0.99990365 | 0.0500 | 0.94990365 | 0.92 |
| 27 | 1 | 0.99993836 | 0.0500 | 0.94993836 | 0.81 |
| 29 | 1 | 0.99996302 | 0.0500 | 0.94996302 | 0.03 |
| 32 | 1 | 0.99990327 | 0.0500 | 0.94990327 | 0.03 |
| 34 | 1 | 0.99985921 | 0.0500 | 0.94985921 | 0.73 |
| 36 | 1 | 0.99995089 | 0.0500 | 0.94995089 | 0.64 |
| 38 | 1 | 0.99764964 | 0.0500 | 0.94764963 | 0.13 |
| 40 | 1 | 0.99994105 | 0.0500 | 0.94994104 | 0.20 |
| 41 | 1 | 0.99992359 | 0.0500 | 0.94992359 | 0.03 |
| 43 | 1 | 0.99995974 | 0.0500 | 0.94995974 | 0.05 |
| 45 | 1 | 0.99993968 | 0.0500 | 0.94993968 | 0.04 |
| 46 | 1 | 0.99997368 | 0.0500 | 0.94997368 | 0.90 |
| 47 | 1 | 0.99993063 | 0.0500 | 0.94993063 | 0.15 |
| 49 | 1 | 0.99995690 | 0.0500 | 0.94995690 | 0.90 |
| 52 | 1 | 0.99995330 | 0.0500 | 0.94995330 | 0.92 |
| 54 | 1 | 0.99995585 | 0.0500 | 0.94995585 | 0.07 |
| 56 | 1 | 0.99995054 | 0.0500 | 0.94995054 | 0.80 |
| 58 | 1 | 0.99994746 | 0.0500 | 0.94994746 | 0.95 |
| 60 | 1 | 0.99998110 | 0.0500 | 0.94998110 | 0.15 |
| 61 | 1 | 0.99994356 | 0.0500 | 0.94994356 | 0.02 |
| 63 | 1 | 0.99995022 | 0.0500 | 0.94995022 | 0.90 |
| 65 | 1 | 0.99995826 | 0.0500 | 0.94995826 | 0.12 |
| 67 | 1 | 0.99997349 | 0.0500 | 0.94997349 | 0.22 |
| 68 | 1 | 0.99994444 | 0.0500 | 0.94994444 | 0.95 |
| 69 | 1 | 0.99998234 | 0.0500 | 0.94998234 | 0.01 |
| 72 | 1 | 0.99996257 | 0.0500 | 0.94996257 | 0.02 |
| 74 | 1 | 0.99440462 | 0.0500 | 0.94440462 | 0.57 |
| 76 | 1 | 0.99914280 | 0.0500 | 0.94914280 | 0.64 |
| 78 | 1 | 0.99997657 | 0.0500 | 0.94997657 | 0.73 |
| 80 | 1 | 0.99899996 | 0.0500 | 0.94899996 | 0.03 |
| 81 | 1 | 0.98798671 | 0.0500 | 0.93798671 | 0.77 |
| 83 | 1 | 0.99985162 | 0.0500 | 0.94985162 | 0.59 |
| 85 | 1 | 0.99998320 | 0.0500 | 0.94998320 | 0.25 |
| 87 | 1 | 0.99995217 | 0.0500 | 0.94995217 | 0.80 |
| 89 | 1 | 0.99995057 | 0.0500 | 0.94995057 | 0.04 |
| 90 | 1 | 0.9947447 | 0.0500 | 0.94474474 | 0.09 |
| Average | 0.99999971 | 0.99897694 | – | 0.94897694 | 0.4466 |
| SD | 0.00000202 | 0.00263041 | – | 0.00263041 | 0.36663172 |

Table 6.15: Cooperative Data Fusion (60%) Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.95 | 0.05 | 5 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| Average | 0.9990 | 0.0010 | 0.1 | 0 | 0 |
| SD | 0.007071 | 0.007071 | – | 0 | – |

Table 6.16: Cooperative Data Fusion (60%) Results

### Stage 4 (70%)

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99991163 | 0.0500 | 0.94991163 | 0.67 |
| 2 | 1 | 0.99995292 | 0.0500 | 0.94995292 | 0.93 |
| 3 | 1 | 0.99709036 | 0.0500 | 0.94709036 | 0.41 |
| 5 | 1 | 0.98615842 | 0.0500 | 0.93615841 | 0.10 |
| 7 | 1 | 0.99997614 | 0.0500 | 0.94997614 | 0.44 |
| 9 | 1 | 0.99997112 | 0.0500 | 0.94997112 | 0.25 |
| 12 | 1 | 0.99995742 | 0.0500 | 0.94995742 | 0.24 |
| 14 | 1 | 0.99979208 | 0.0500 | 0.94979208 | 0.51 |
| 16 | 1 | 0.99988488 | 0.0500 | 0.94988488 | 0.35 |
| 18 | 1 | 0.99996285 | 0.0500 | 0.94996285 | 0.94 |
| 20 | 1 | 0.99995091 | 0.0500 | 0.94995091 | 0.95 |
| 21 | 1 | 0.99996662 | 0.0500 | 0.94996662 | 0.47 |
| 23 | 0.99998238 | 0.99344748 | 0.0500 | 0.94344748 | 0.95 |
| 24 | 1 | 0.99873973 | 0.0500 | 0.94873973 | 0.71 |
| 25 | 1 | 0.99987783 | 0.0500 | 0.94987783 | 0.95 |
| 27 | 1 | 0.99989272 | 0.0500 | 0.94989272 | 0.76 |
| 29 | 1 | 0.99997935 | 0.0500 | 0.94997935 | 0.03 |
| 32 | 1 | 0.99995041 | 0.0500 | 0.94995041 | 0.03 |
| 34 | 1 | 0.99988808 | 0.0500 | 0.94988808 | 0.68 |
| 36 | 1 | 0.99998034 | 0.0500 | 0.94998034 | 0.70 |
| 38 | 1 | 0.99767674 | 0.0500 | 0.94767674 | 0.14 |
| 40 | 1 | 0.99997277 | 0.0500 | 0.94997277 | 0.25 |
| 41 | 1 | 0.99994357 | 0.0500 | 0.94994357 | 0.19 |
| 43 | 1 | 0.99992265 | 0.0500 | 0.94992265 | 0.02 |
| 45 | 1 | 0.99992186 | 0.0500 | 0.94992186 | 0.03 |
| 46 | 1 | 0.99994753 | 0.0500 | 0.94994753 | 0.84 |
| 47 | 1 | 0.99987963 | 0.0500 | 0.94987963 | 0.14 |
| 49 | 1 | 0.99989966 | 0.0500 | 0.94989966 | 0.95 |
| 52 | 1 | 0.99990184 | 0.0500 | 0.94990184 | 0.99 |
| 54 | 1 | 0.99998099 | 0.0500 | 0.94998099 | 0.15 |
| 56 | 1 | 0.99989832 | 0.0500 | 0.94989831 | 0.52 |
| 58 | 1 | 0.99987557 | 0.0500 | 0.94987557 | 0.95 |
| 60 | 1 | 0.99997128 | 0.0500 | 0.94997128 | 0.17 |
| 61 | 1 | 0.99981937 | 0.0500 | 0.94981937 | 0.02 |
| 63 | 1 | 0.99997335 | 0.0500 | 0.94997335 | 0.91 |
| 65 | 1 | 0.99994637 | 0.0500 | 0.94994637 | 0.09 |
| 67 | 1 | 0.99993493 | 0.0500 | 0.94993493 | 0.20 |
| 68 | 1 | 0.99996165 | 0.0500 | 0.94996165 | 0.90 |
| 69 | 1 | 0.99992784 | 0.0500 | 0.94992784 | 0.04 |
| 72 | 1 | 0.99993243 | 0.0500 | 0.94993243 | 0.03 |
| 74 | 1 | 0.99493857 | 0.0500 | 0.94493857 | 0.82 |
| 76 | 1 | 0.99817754 | 0.0500 | 0.94817754 | 0.17 |
| 78 | 1 | 0.99995635 | 0.0500 | 0.94995635 | 0.64 |
| 80 | 1 | 0.99315162 | 0.0500 | 0.94315162 | 0.14 |
| 81 | 1 | 0.95857320 | 0.0500 | 0.90857320 | 0.47 |
| 83 | 1 | 0.99985074 | 0.0500 | 0.94985074 | 0.50 |
| 85 | 1 | 0.99995945 | 0.0500 | 0.94995945 | 0.01 |
| 87 | 1 | 0.99992721 | 0.0500 | 0.94992721 | 0.92 |
| 89 | 1 | 0.99996624 | 0.0500 | 0.94996624 | 0.03 |
| 90 | 1 | 0.99305809 | 0.0500 | 0.94305809 | 0.03 |
| Average | 0.99999965 | 0.99816357 | – | 0.94816357 | 0.4466 |
| SD | 0.00000249 | 0.00626269 | – | 0.00626269 | 0.35438714 |

Table 6.17: Cooperative Data Fusion (70%) Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Positive Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.95 | 0.05 | 5 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| Average | 0.9990 | 0.0010 | 0.1 | 0 | 0 |
| SD | 0.007071 | 0.007071 | – | 0 | – |

Table 6.18: Cooperative Data Fusion (70%) Results

## 6.5 Conclusion

In this chapter, a discussion of the classification outcomes for authentication purposes has been presented in section 6.2; the measurements retrievable from these outcomes are discussed in section 6.2.1.

As Receiver Operating Characteristics graphs were utilised in this study to assist classification, a description of their operations and properties were outlined in section 6.2.2, along with the reasons for their use.

The implementation of ROC analysis in this study was outlined in section 6.3. This included a description of how the decision threshold was determined. This threshold was the basis for determination of performance metrics used to present results (for each training group member, for each phase of the experiment).

Finally, the experimental results were presented in table form in section 6.4. The following sections presented the tabled results for all phases of the experiment:

1. Keystoke Dynamics—section 6.4.1.

2. Fingerprint Recognition—section 6.4.2.

3. Feature Level Data Fusion:

    - Complimentary Data Fusion—section 6.4.3.1.
    - Cooperative Data Fusion—section 6.4.3.2:
        (a) Stage 1 (40%).
        (b) Stage 2 (50%).
        (c) Stage 3 (60%).
        (d) Stage 4 (70%).

Given the summary statistics presented in Tables 6.6, 6.8, 6.10, 6.12, 6.14, 6.16, 6.18, Table 6.19 presents the overall performance—as averages and standard deviations—achieved in each phase of the experiment for the $frr$ and $far$ metrics.

| Phase | False Rejection Rate Average | False Rejection Rate Standard Deviation | False Acceptance Rate Average | False Acceptance Rate Standard Deviation |
|---|---|---|---|---|
| Keystoke Dynamics | 0.0862 | 0.051504 | 0.02766095 | 0.03806474 |
| Fingerprint Recognition | 0.0022 | 0.009957 | 0 | 0 |
| Complimentary Data Fusion | 0.0004 | 0.002828 | 0 | 0 |
| Cooperative Data Fusion–Stage 1 | 0.0004 | 0.002828 | 0.0000381 | 0.00001885 |
| Cooperative Data Fusion–Stage 2 | 0.0006 | 0.004243 | 0 | 0 |
| Cooperative Data Fusion–Stage 3 | 0.0010 | 0.007071 | 0 | 0 |
| Cooperative Data Fusion–Stage 4 | 0.0010 | 0.007071 | 0 | 0 |

Table 6.19: Summary Statistics of Experimental Results

The results presented in this chapter reflect the performance of classifiers at the nominated decision thresholds only. The study makes no claim that these results would be reflected at any other decision threshold or cut-off value. However, the method for determining the decision threshold was scientifically sound, and therefore the results should be repeatable.

The next chapter discusses the results in detail and compares the findings with other research in the respective fields.

# Chapter 7

# Discussion Of Results

## 7.1 Introduction

This chapter presents a discussion of the results achieved in the current experiment, that were presented in Chapter 6, and also compares the results with other research efforts in the respective fields. Results for the keystroke dynamics phase of the experiment are discussed in section 7.2.1, and results for the fingerprint recognition phase of the experiment are discussed in section 7.2.2.

Section 7.2.3 provides a discussion of results for the feature level data fusion phase of the experiment, including both the complementary paradigm (section 7.2.3.1) and the cooperative paradigm (section 7.2.3.3). In section 7.2.3.3, a discussion is provided for results in relation to the four experimental stages conducted for the cooperative data fusion paradigm; that is, the four stages where different proportions of available data were used for data fusion[1].

Finally, section 7.3 provides a conclusion to the chapter.the results

## 7.2 Discussion

This section presents a detailed discussion of the results for the three phases of the experiment. Statistics pertaining to the determination of the decision thresholds, and the corresponding experimental findings, for all training group members were presented via the tables in Chapter 6 section 6.4.

[1]Described in Chapter 5 section 5.6.3.

For convenience, these tables have been duplicated and re-numbered in this chapter. Table 7.1 illustrates the correspondence between the tables as they were numbered in Chapter 6 and as they are numbered for this Chapter. Thus the following discussion will henceforth make reference to the figures in the tables as they are numbered for this chapter, according to Table 7.1.

| Table number in Chapter 6 | Table number in this Chapter |
|---|---|
| Table 6.5 | Table 7.2 |
| Table 6.6 | Table 7.3 |
| Table 6.7 | Table 7.7 |
| Table 6.8 | Table 7.8 |
| Table 6.9 | Table 7.10 |
| Table 6.10 | Table 7.11 |
| Table 6.11 | Table 7.13 |
| Table 6.12 | Table 7.14 |
| Table 6.13 | Table 7.15 |
| Table 6.14 | Table 7.16 |
| Table 6.15 | Table 7.17 |
| Table 6.16 | Table 7.18 |
| Table 6.17 | Table 7.19 |
| Table 6.18 | Table 7.20 |

Table 7.1: Corresponding Table Numbers

For all phases of the experiment, the decision threshold was determined as that operating point, on each training group member's ROC curve, which returned the best results in terms of the performance variables; the false acceptance rate (FAR) and the false rejection rate (FRR)[2].

After discussing the results for each phase of the experiment (by referring to the tables listed in Table 7.1), a comparison will be made between the results achieved in the current experiment and the results presented in the publications reviewed for keystroke dynamics (Chapter 3 section 3.4.1), fingerprint recognition (Chapter 4 section 4.5), and multi-modal biometrics (Chapter 2 section 2.3.2.2).

Whilst making these comparisons, attention will be drawn to similarities and differences between the current study and those reviewed. This is done to highlight what is different about the methodology used in the current study (in relation to those papers reviewed), and to demonstrate that the results achieved—utilising the stated methodology—were comparable to those achieved in other research efforts in their respective fields.

[2]Refer to the discussion in Chapter 6 section 6.2.1.

### 7.2.1 Discussion Of Keystroke Dynamics Results

The statistics in Table 7.2 provide information regarding the calculation of the decision threshold, for each of the 50 training group members, for the keystroke dynamics phase of the experiment[3].

Figures in column 2 of Table 7.2 (the Area Under the ROC Curve) show that member 85 achieved an area of 1[4]. With a FAR of 0.0 and FRR of 0.0, member 85 indeed achieved the best possible results (as demonstrated in Table 7.3). Training group members 16, 18, and 52 also achieved some of the best AUC values (0.99442571, 0.99973810, and 0.99994952 respectively), which resulted in very good results for both performance variables.

It can also be seen that training group members 43, 74, 3, and 12 had the four lowest AUC values (0.83263333, 0.86826095, 0.88062905, and 0.91994667 respectively). This meant that the confidence levels (Table 7.2 column 4) for those four members were higher than 0.05[5] (0.075, 0.075, 0.075, and 0.0625 respectively). Figures in Table 7.3 demonstrate that these training group members indeed achieved the worst results for both performance variables.

Training group members 29, 49, and 61 also had higher confidence levels (0.0625 each), resulting from lower AUC values of 0.94121762, 0.94541667, and 0.94248857 respectively. For member 29 the lower AUC and higher confidence level resulted in high error rates for both performance variables (a FAR of 0.08104762 and a FRR of 0.13). For members 49 and 61, they resulted in a high rate for one performance variable and a low rate for the other. That is, member 49 scored a FAR of 0.08647619 and a FRR of 0.06, and member 61 scored a FAR of 0.00866667 and a FRR of 0.22.

Note that the decision thresholds for keystroke dynamics were generally above 0.9; although 7 were less than 0.9, they were quite close. In fact, the average for the decision thresholds was 0.9486 with a standard deviation of 0.03811958. This was most likely a result of the high variability associated with keystroke dynamics data.

[3]The processes for calculating the area under the ROC curve (AUC) and the decision threshold were described in Chapter 6 sections 6.3.3 and 6.3.4 respectively.

[4]An AUC value of 1 demonstrates perfect recognition at the nominated decision threshold.

[5]As explained in Chapter 6 section 6.3.4, a higher value for the confidence level decreased the adjustable determinant (AD) value, and actually indicated lower confidence in the performance of the classifier (typically producing poorer results).

As will be seen in the following sections, the decision thresholds for the keystroke dynamics phase of the experiment were typically much higher than those in the remaining phases of the experiment.

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 0.99135190 | 0.98927652 | 0.0500 | 0.93927652 | 0.96 |
| 2 | 0.95659857 | 0.95334717 | 0.0500 | 0.90334717 | 0.97 |
| 3 | 0.88062905 | 0.85889453 | 0.0750 | 0.78389453 | 0.86 |
| 5 | 0.99785381 | 0.99474991 | 0.0500 | 0.94474991 | 0.98 |
| 7 | 0.99322429 | 0.98808178 | 0.0500 | 0.93808178 | 0.99 |
| 9 | 0.96157905 | 0.96274334 | 0.0500 | 0.91274334 | 0.95 |
| 12 | 0.91994667 | 0.93981623 | 0.0625 | 0.87731623 | 0.97 |
| 14 | 0.99516429 | 0.98649916 | 0.0500 | 0.93649916 | 0.93 |
| 16 | 0.99442571 | 0.97118196 | 0.0500 | 0.92118195 | 0.95 |
| 18 | 0.99973810 | 0.99862410 | 0.0500 | 0.94862410 | 0.99 |
| 20 | 0.99369381 | 0.97664731 | 0.0500 | 0.92664731 | 0.98 |
| 21 | 0.98268905 | 0.96702640 | 0.0500 | 0.91702640 | 0.97 |
| 23 | 0.97532000 | 0.95544168 | 0.0500 | 0.90544168 | 0.96 |
| 24 | 0.95277429 | 0.95765571 | 0.0500 | 0.90765571 | 0.94 |
| 25 | 0.99058143 | 0.95367530 | 0.0500 | 0.90367530 | 0.96 |
| 27 | 0.99949143 | 0.99109513 | 0.0500 | 0.94109513 | 0.96 |
| 29 | 0.94121762 | 0.93243909 | 0.0625 | 0.86993909 | 0.92 |
| 32 | 0.97951571 | 0.94305124 | 0.0500 | 0.89305124 | 0.98 |
| 34 | 0.99918000 | 0.98004077 | 0.0500 | 0.93004077 | 0.95 |
| 36 | 0.99211571 | 0.98287234 | 0.0500 | 0.93287234 | 0.96 |
| 38 | 0.99692810 | 0.99031712 | 0.0500 | 0.94031712 | 0.99 |
| 40 | 0.99834524 | 0.99927547 | 0.0500 | 0.94927547 | 0.99 |
| 41 | 0.98708762 | 0.95652380 | 0.0500 | 0.90652380 | 0.97 |
| 43 | 0.86263333 | 0.86382857 | 0.0750 | 0.78882857 | 0.99 |
| 45 | 0.98877381 | 0.99863071 | 0.0500 | 0.94863071 | 0.99 |
| 46 | 0.99621143 | 0.99234570 | 0.0500 | 0.94234570 | 0.96 |
| 47 | 0.98223762 | 0.95406353 | 0.0500 | 0.90406353 | 0.89 |
| 49 | 0.94541667 | 0.97502202 | 0.0625 | 0.91252202 | 0.94 |
| 52 | 0.99994952 | 0.99833596 | 0.0500 | 0.94833596 | 0.98 |
| 54 | 0.98558571 | 0.96455751 | 0.0500 | 0.91455750 | 0.94 |
| 56 | 0.97548333 | 0.91711252 | 0.0500 | 0.86711252 | 0.92 |
| 58 | 0.95181810 | 0.91879129 | 0.0500 | 0.86879129 | 0.93 |
| 60 | 0.99755619 | 0.97835570 | 0.0500 | 0.92835570 | 0.97 |
| 61 | 0.94248857 | 0.84054680 | 0.0625 | 0.77804680 | 0.95 |
| 63 | 0.95563619 | 0.93299954 | 0.0500 | 0.88299954 | 0.89 |
| 65 | 0.97971571 | 0.89510313 | 0.0500 | 0.84510313 | 0.88 |
| 67 | 0.96299048 | 0.90508670 | 0.0500 | 0.85508670 | 0.88 |
| 68 | 0.97225952 | 0.94748348 | 0.0500 | 0.89748348 | 0.98 |
| 69 | 0.99225190 | 0.97082952 | 0.0500 | 0.92082952 | 0.95 |
| 72 | 0.99848952 | 0.99350247 | 0.0500 | 0.94350247 | 0.98 |
| 74 | 0.86826095 | 0.84537021 | 0.0750 | 0.77037021 | 0.82 |
| 76 | 0.98762333 | 0.97372189 | 0.0500 | 0.92372189 | 0.94 |
| 78 | 0.97809667 | 0.94457460 | 0.0500 | 0.89457460 | 0.95 |
| 80 | 0.97258619 | 0.95375459 | 0.0500 | 0.90375459 | 0.97 |
| 81 | 0.95439667 | 0.89733494 | 0.0500 | 0.84733494 | 0.98 |
| 83 | 0.98380381 | 0.96253737 | 0.0500 | 0.91253737 | 0.97 |
| 85 | 1.00000000 | 0.99663067 | 0.0500 | 0.94663067 | 0.88 |
| 87 | 0.95541905 | 0.92590986 | 0.0500 | 0.87590986 | 0.94 |
| 89 | 0.98358381 | 0.98422130 | 0.0500 | 0.93422130 | 0.97 |
| 90 | 0.97026905 | 0.91696189 | 0.0500 | 0.86696189 | 0.91 |
| **Average** | 0.97245977 | 0.95353777 | – | 0.90103777 | 0.9486 |
| **SD** | 0.03232844 | 0.04117834 | – | 0.04586111 | 0.03811958 |

Table 7.2: Duplication of Keystroke Dynamics Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 0.94 | 0.06 | 6 | 0.01561905 | 164 |
| 2 | 0.91 | 0.09 | 9 | 0.06533333 | 686 |
| 3 | 0.79 | 0.21 | 21 | 0.12600000 | 1323 |
| 5 | 0.95 | 0.05 | 5 | 0.00466667 | 49 |
| 7 | 0.96 | 0.04 | 4 | 0.00247619 | 26 |
| 9 | 0.93 | 0.07 | 7 | 0.05200000 | 546 |
| 12 | 0.88 | 0.12 | 12 | 0.10628571 | 1116 |
| 14 | 0.94 | 0.06 | 6 | 0.01342857 | 141 |
| 16 | 0.95 | 0.05 | 5 | 0.00009524 | 1 |
| 18 | 0.97 | 0.03 | 3 | 0.00047619 | 5 |
| 20 | 0.94 | 0.06 | 6 | 0.01009524 | 106 |
| 21 | 0.92 | 0.08 | 8 | 0.01742857 | 183 |
| 23 | 0.91 | 0.09 | 9 | 0.01409524 | 148 |
| 24 | 0.91 | 0.09 | 9 | 0.06952381 | 730 |
| 25 | 0.91 | 0.09 | 9 | 0.00142857 | 15 |
| 27 | 0.95 | 0.05 | 5 | 0.00085714 | 9 |
| 29 | 0.87 | 0.13 | 13 | 0.08104762 | 851 |
| 32 | 0.90 | 0.10 | 10 | 0.00104762 | 11 |
| 34 | 0.96 | 0.04 | 4 | 0.00123810 | 13 |
| 36 | 0.95 | 0.05 | 5 | 0.01171429 | 123 |
| 38 | 0.96 | 0.04 | 4 | 0.00542857 | 57 |
| 40 | 0.99 | 0.01 | 1 | 0.00323810 | 34 |
| 41 | 0.91 | 0.09 | 9 | 0.01028571 | 108 |
| 43 | 0.81 | 0.19 | 19 | 0.15019048 | 1577 |
| 45 | 0.98 | 0.02 | 2 | 0.02190476 | 230 |
| 46 | 0.95 | 0.05 | 5 | 0.01200000 | 126 |
| 47 | 0.92 | 0.08 | 8 | 0.02171429 | 228 |
| 49 | 0.94 | 0.06 | 6 | 0.08647619 | 908 |
| 52 | 0.99 | 0.01 | 1 | 0.00009524 | 1 |
| 54 | 0.92 | 0.08 | 8 | 0.01152381 | 121 |
| 56 | 0.87 | 0.13 | 13 | 0.00552381 | 58 |
| 58 | 0.87 | 0.13 | 13 | 0.05095238 | 535 |
| 60 | 0.93 | 0.07 | 7 | 0.00114286 | 12 |
| 61 | 0.78 | 0.22 | 22 | 0.00866667 | 91 |
| 63 | 0.89 | 0.11 | 11 | 0.06076190 | 638 |
| 65 | 0.85 | 0.15 | 15 | 0.00400000 | 42 |
| 67 | 0.86 | 0.14 | 14 | 0.02923810 | 307 |
| 68 | 0.90 | 0.10 | 10 | 0.01838095 | 193 |
| 69 | 0.94 | 0.06 | 6 | 0.01190476 | 125 |
| 72 | 0.95 | 0.05 | 5 | 0.00266667 | 28 |
| 74 | 0.78 | 0.22 | 22 | 0.14400000 | 1512 |
| 76 | 0.94 | 0.06 | 6 | 0.01380952 | 145 |
| 78 | 0.92 | 0.08 | 8 | 0.01104762 | 116 |
| 80 | 0.91 | 0.09 | 9 | 0.02304762 | 242 |
| 81 | 0.86 | 0.14 | 14 | 0.01342857 | 141 |
| 83 | 0.93 | 0.07 | 7 | 0.00438095 | 46 |
| 85 | 1.00 | 0.00 | 0 | 0.00000000 | 0 |
| 87 | 0.88 | 0.12 | 12 | 0.02695238 | 283 |
| 89 | 0.95 | 0.05 | 5 | 0.02800000 | 294 |
| 90 | 0.87 | 0.13 | 13 | 0.00742857 | 78 |
| **Average** | 0.9138 | 0.0862 | 8.62 | 0.02766095 | 290.44 |
| **SD** | 0.051504 | 0.051504 | – | 0.03806474 | – |

Table 7.3: Duplication of Keystroke Dynamics Results

The individual training group members' results, demonstrated in Table 7.3, indicate that the lowest FAR (column 5) was 0.0 attained by member 85, whilst members 16 and 52 attained a FAR of 0.00009524. This meant that no impostor samples were incorrectly accepted as belonging to member 85, and that only 1 impostor sample each was incorrectly accepted as belonging to members 16 and 52[6]. Other low false

[6]Recall: member 85 had an AUC of 1, and members 16 and 52 had an AUC very close to 1.

acceptance rates were 0.00047619 (5 false acceptances out of 10,500) and 0.00085714 (9 false acceptances out of 10,500), attained by members 18 and 27 respectively. The above rates would typically be considered very good rates for the FAR performance variable.

False acceptance rates of 0.15019048, 0.144, 0.126, and 0.10628571 (attained by training group members 43, 74, 3, and 12 respectively) were the highest of all those attained by any training group member. This meant that 1,577, 1,512, 1,323, and 1,116 impostor samples (out of 10,500) were incorrectly accepted as belonging to members 43, 74, 3, and 12 respectively. Other high false acceptance rates were 0.08647619 (908 false acceptances out of 10,500) and 0.08104762 (851 false acceptances out of 10,500), attained by training group members 49 and 29 respectively. The above rates would definitely be considered unacceptably high rates for the FAR performance variable.

In general, if an acceptable upper limit for the FAR was set to 0.001, it would mean that no more than 1 impostor sample in 1,000 could be incorrectly accepted. In the current experiment, that would equate to approximately 10 impostor samples or less (out of 10,500), which could be incorrectly accepted for each of the training group members.

According to Table 7.3 only 5 members achieved a FAR less than or equal to 0.001. However in keystroke dynamics research, a FAR of 0.001 is not typically achieved often (refer Table 3.2 in Chapter 3 section 3.4.1). If the upper limit was to be relaxed to 0.0025 for example, then 10 members achieved a FAR less than or equal to this value (that is, 26 impostor samples or less were falsely accepted).

The lowest FRR of 0.0 (refer Table 7.3 column 3) was attained by training group member 85, whilst members 40 and 52 shared a rate of 0.01. This meant that for member 85 no genuine samples (out of 100) were incorrectly rejected, and for members 40 and 52 only 1 each of their 100 genuine samples were incorrectly rejected. Other low false rejection rates were 0.02 and 0.03, attained by members 45 and 18 respectively. The above rates would typically be considered acceptable for the FRR performance variable.

The highest FRR of 0.22 was attained by training group members 61 and 74. This meant that 22 of their 100 genuine samples were incorrectly rejected. Other high false rejection rates were 0.21 and 0.19, attained by members 3 and 43 respectively. The above rates would definitely be considered unacceptably high rates for the FRR performance variable.

In total there were 14 training group members whose FRR was above 0.1 (i.e. 10 out of 100). In some circumstances (though probably not many), 0.1 may be considered an acceptable upper limit for the FRR. For example in a mission critical application, the FAR would be required to be as low as possible; this restriction applied to the FAR would very likely result in a higher FRR (so the accuracy required for this rate may be relaxed).

However, most researchers would obviously prefer a lower FRR. For example 0.05 may be considered a more preferred upper limit. From Table 7.3, it can be seen that 14 training group members achieved a FRR less than or equal to 0.05. Therefore, 22 members had rates greater than 0.05 and less than or equal to 0.1.

As demonstrated in Table 7.3, the training group members who attained the four best FAR scores also achieved some of the best FRR scores. That is, member 85 (with a FAR of 0.0 and a FRR of 0.0), member 52 (with a FAR of 0.00009524 and a FRR of 0.01), member 16 (with a FAR of 0.00009524 and a FRR of 0.05), and member 18 (with a FAR of 0.00047619 and a FRR of 0.03).

Also, the training group members who attained the four worst FAR scores achieved some of the worst FRR scores. That is, member 43 (with a FAR of 0.15019048 and a FRR of 0.19), member 74 (with a FAR of 0.144 and a FRR of 0.22), member 3 (with a FAR of 0.126 and a FRR of 0.21), member 12 (with a FAR of 0.10628571 and a FRR of 0.12).

Contrary to the observations just highlighted, member 61 had the equal highest FRR of 0.22 (22 in 100 genuine samples falsely rejected), yet attained a FAR of only 0.008667 (91 in 10,500 impostor samples incorrectly accepted). Member 49 had a rather high FAR of 0.08647619 (908 in 10,500 impostor samples incorrectly accepted), yet attained a FRR of only 0.06 (6 in 100 genuine samples falsely rejected).

The last two rows of Table 7.3 provide the average and standard deviation FAR and FRR figures, for all training group members, for the keystroke dynamics phase of the experiment. The average FAR was 0.02766095 (2.766%), with a standard deviation of 0.03806474; the average FRR was 0.0862 (8.62%), with a standard deviation of 0.0515. These figures demonstrated that the keystroke dynamics phase of the experiment performed reasonably well (compared to the research efforts reviewed in Chapter 3 section 3.4).

It meant that on average, there was approximately a 9 in 100 chance that any of the 50 training group members would have one of their own genuine samples incorrectly rejected. Also on average, there was approximately a 290 in 10,500 chance that any training group member would have an impostor sample incorrectly accepted as their own.

### Comparison with Reviewed Keystroke Dynamics Research

The average FAR and average FRR—for the 50 training group members—provide figures that permit a comparison with the results of the research efforts reviewed in Chapter 3 section 3.4.1. Though the majority of those research efforts designed their experiments based on a uni-modal approach—as opposed to the current study, which was designed to facilitate a multi-modal approach—it is still feasible to compare other research results with those of the current study, as in most cases the same performance variables (that is, the FAR and the FRR) were used[7].

#### *Comparison to Past Studies Using Statistical Analysis Methods*

Firstly, a comparison between the results achieved in the current study and those achieved by research efforts that utilised deterministic statistical analysis methods of keystroke dynamics data will be considered. Table 7.4 presents a summary of the results achieved by research efforts that adopted such analysis methods.

[7]As noted in Chapter 3 section 3.4.1, numerous authors expressed these performance variables as a percentage. That is, the percentage of the FAR and the percentage of the FRR. However in Table 3.2 columns 9 and 10, the figures were denoted as their actual rates (i.e. the percentage divided by 100). Therefore, any discussion comparing the performance variables will denote both the actual rate and the percentage rate (in parentheses).

The information in Table 7.4 is duplicated from Table 3.2 in Chapter 3 section 3.4.1, and is provided for convenient comparison between the results of the reviewed research and the results of the current study[8].

| Reviewed Paper | FAR | FRR |
|---|---|---|
| Gaines et al., 1980 | 0.0 | 0.04 |
| Umphress and Williams, 1985 | 0.0588 | 0.1176 |
| Leggett and Williams, 1988 | 0.05 | 0.055 |
| Joyce and Gupta, 1990 | 0.0025 | 0.1636 |
| Bergadano et al., 2002 | 0.0001 | 0.04 |
| **Current Study, 2010** | **0.02766** | **0.0862** |

Table 7.4: Summary of Reviewed Papers Using Statistical Analysis Methods

The earliest reviewed research paper achieved a FAR of 0.0 and FRR of 0.04 (4.0%) (Gaines et al., 1980). The results achieved in the current study did not achieve the same accuracy as those by Gaines et al., (1980). The FAR of 0.02766095 achieved in the current study was less accurate than the FAR of 0.0 achieved by Gaines et al., (1980). Also, the FRR of 0.0862 was less accurate than the 0.04 achieved by Gaines et al., (1980).

However, although the results achieved by Gaines et al., (1980) appeared very impressive, the authors stated that methodological issues implied some uncertainty in their results. These concerns were the small number of participants (6 only), and the small number of samples collected from those 6 participants (3 samples from each participant).

This small number of samples per participant would not normally capture the intra-class variance, so variability in the biometric data for the same individual may not be visible. Therefore, the intra-class variance would most likely be very low compared to the inter-class variance, leading to an apparent improvement in performance of the classifier.

For the current study, 90 participants provided 140 samples each. This provided more samples for training and testing purposes; each participant was tested for false acceptance with 10,500 impostor samples and for false rejection with 100 genuine

[8]Note that a direct comparison between the experimental results of the publications listed in Table 7.4 (and also between them and the results of the current experiment) would be misleading. All of these results can only be viewed as an informal indicator of algorithmic performance because different data sets were used in the experiments.

samples. Therefore, a much finer granularity can be attributed to both performance variables in the current study.

Other differences between the two studies were:

- Data filtering. Gaines et al., (1980) included only time values associated with digraphs that occurred 10 times or more. According to the authors, this was necessary because of the small amount of data collected. The current study utilised 4 statistics (associated with an assumed normal distribution), to filter noisy data that could have impeded ANN pattern recognition capabilities (refer Chapter 5 section 5.4.4).

- Metrics used. Gaines et al., (1980) used one metric only, the keystroke latency discussed in Chapter 3 section 3.3. As explained in that section, the use of two metrics—the keystroke duration and digraph latency—has become the accepted 'norm'. These two metrics were used in the current study.

- Analysis method. In their experiment, Gaines et al., (1980) utilised a traditional T-test for classification of typing patterns, whilst the current study utilised ANNs for this task. As discussed in the review in Chapter 3 section 3.4.1, ANNs have demonstrated more discriminative pattern recognition capabilities than deterministic statistical methods of analysis.

All of the issues just discussed, could imply uncertainty in the results achieved by Gaines et al., (1980), whereas the methodology adopted in the current study should inspire more confidence in the achieved results.

Gaines et al., (1980) recommended the use of certain character combinations to provide better discrimination between typing patterns (refer Chapter 3 section 3.4.1). These character combinations are all typed by the right hand in the standard typing method. By incorporating the recommended combinations into the character string that participants were required to type—and also incorporating the corresponding combinations typed by the left hand—the results of the current research indicated a more distinctive signature for each participants' typing pattern.

The results achieved in the current study demonstrated better accuracy than the results presented by Umphress and Williams (1985), with a FAR of 0.0588 (5.88%) and a FRR of 0.1176 (11.76%), and Leggett and Williams (1988), with a FAR of 0.05 (5.0%) and a FRR of 0.055 (5.5%). The study by Joyce and Gupta (1990) achieved a FAR of 0.0025 (0.25%) and a FRR of 0.1636 (16.36%). Whilst the FAR was more accurate than that achieved in the current study (0.0025 compared to 0.02766095), the FRR was much less accurate (0.1636 compared to 0.0862).

The above authors all used deterministic statistical analysis methods. Their experiments also exhibited similar methodological concerns to those of Gaines et al., (1980); that is, a small number of participants were recruited, a small number of samples was provided by each participant, and only one metric was used. Again, these concerns could imply uncertainty in the results achieved by those authors.

The experiment by Bergadano et al., (2002) achieved a FAR of 0.0001 (0.01%) and FRR of 0.04 (4%). Both the FAR and the FRR achieved better accuracy than those achieved in the current study. The number of participants was comparable to that used in the current study, however, the number of samples provided by participants was much lower (5 compared to 140). Even though the authors generated more test samples from the initial inputs, that procedure—as discussed in Chapter 3 section 3.4.1—raises experimental validity concerns which could give rise to some doubt concerning the reported performance.

The Bergadano et al., (2002) study also differed from the current study in the following points:

- The authors used trigraphs to calculate metrics and other measurements. In the current study digraphs were used to determine metrics.
- Only one metric was used in the their study (the degree of disorder), whereas the current study used two (the keystroke duration and digraph latency).
- The authors used a resolution of 10 milliseconds to measure keystroke events, whereas the current study used a 1 millisecond resolution.
- The authors used deterministic statistical methods for classification, whereas in the current study ANNs were used for classification.

The degree of disorder of trigraphs, and the 10 millisecond keystroke event capture resolution, are courser measurements than those typically used by other research efforts. These facts could raise concerns about whether an individuals typing pattern remains uniquely identifiable (using the stated measurements).

Before comparing the current results with the research efforts that employed machine learning techniques for data analysis, the point should be clarified that the current study did utilise some statistical methods in the experiment. Statistical methods were used for feature selection; that is, selection of the most appropriate features—by discarding the most noisy features—to assist the ANNs in performing their pattern recognition task. The process is explained in detail in Chapter 5 section 5.4.4. However, the analysis of the keystroke dynamics data—for the purpose of recognising typing patterns—was performed only by ANNs.

### *Comparison to Past Studies Using Machine Learning Techniques*

A comparison between the results achieved in the current study and those achieved by research efforts that utilised machine learning techniques (other than ANNs) for analysis of keystroke dynamics data will now be considered. Table 7.5 presents a summary of the results achieved by research efforts that adopted machine learning techniques for analysis. The information in Table 7.5 is duplicated from Table 3.2 in Chapter 3 section 3.4.1, and is provided for convenient comparison between the results of the reviewed research and the results of the current study[9].

| Reviewed Paper | FAR | FRR |
|---|---|---|
| Peacock, 2000 | 0.042 | 0.08 |
| Yu and Cho, 2004 | 0.0 | 0.0369 |
| Jiang et al., 2007 | 0.0254 | 0.0254 |
| Hu et al., 2008 | 0.00045 | 0.0 |
| **Current Study, 2010** | **0.02766** | **0.0862** |

Table 7.5: Summary of Reviewed Papers Using Machine Learning Techniques

[9]Note that a direct comparison between the experimental results of the publications listed in Table 7.5 (and also between them and the results of the current experiment) would be misleading. All of these results can only be viewed as an informal indicator of algorithmic performance because different data sets were used in the experiments.

In comparison with Peacock (2000), the current study achieved a similar FRR result, with Peacock (2000) achieving 0.08 (8.0%) compared to the current study with 0.0862 (8.62%). However, the current study achieved a more accurate result for the FAR; 0.02766095 (2.766%) compared to 0.042 (4.2%).

Some of the methodological concerns exhibited in the experiments discussed previously (that is, for those that used deterministic statistical analysis methods) were also evident in the experiment conducted by Peacock (2000).

The number of participants was 11 only, and the number of samples provided by each of those participants was 20 only. Also, Peacock (2000) used only one metric; the sum of all 6 possible metrics calculable from any digraph (refer Chapter 3 sections 3.3 and 3.4.1).

The experiment by Peacock (2000) was the first to utilise the k-Nearest Neighbour (KNN) algorithm for typing pattern classification. The results were reasonably comparable to other research of the time, though as stated in the previous paragraph, some uncertainty could be attributed to the outcomes.

Hu et al., (2008) also used the KNN algorithm as a classifier. However, like Bergadano et al. (2002), the authors used trigraphs instead of digraphs to calculate metrics; the one metric used was the generalised degree of disorder (which included the similarity measure $A$ described in Chapter 3 section 3.4.1).

The results were impressive, with a FAR of 0.00045 (0.045%) and a FRR of 0.0. Because similar metrics were calculated from trigraphs, the results could rightly be compared with those achieved by Bergadano et al., (2002). Hu et al., (2008) could justifiably assert that the different method of classification, and the similarity measure used, achieved improved results compared to those achieved by Bergadano et al., (2002).

The experiment conducted by Hu et al., (2008) also achieved more accurate results than those achieved in the current study. Differences between the Hu et al., (2008) study and the current study were:

1. The number of legitimate users (19) was smaller than the 50 (training group members) used in the current study.

2. The number of samples per legitimate user was very small. That is, 5 compared to the 140 provided by all participants in the current study.

3. The use of trigraphs instead of digraphs to calculate the generalised degree of disorder and the similarity measure $A$.

Points 1 and 2 could raise some concerns. Would a similar accuracy be achieved if a larger population and quantity of samples (per legitimate user) were used? However, the results were very encouraging; it appears that the use of the generalised degree of disorder (incorporating the similarity measure $A$) may help to overcome some variability issues inherent in keystroke dynamics data.

Yu and Cho, (2004) achieved a FAR of 0.0 and a FRR of 0.0369 (3.69%) using a genetic algorithm approach for data analysis—and subsequent feature selection—and support vector machines for classification. The results by Yu and Cho, (2004) were obviously more accurate than those achieved in the current study (with a FAR of 0.02766095 and a FRR of 0.0862).

Yu and Cho, (2004) achieved a FAR of 0.0 by forcing that performance variable to that value (by manipulating the decision threshold), at the expense of increasing the FRR (to 0.0369). The results of the current study were not forced to attain a FAR of 0.0. It would have been possible to force the FAR to 0.0, however this was considered contrary to the purpose of the experiment and could have led to the results being considered contrived.

The number of participants and the number of samples used for testing purposes in the Yu and Cho, (2004) experiment, were considerably smaller than those in the current experiment. In particular, there were only 75 samples used to test false acceptance, which determines the value of the FAR performance variable. Thus, less confidence could be attributed to the results achieved by Yu and Cho, (2004).

Overall, the results the Yu and Cho, (2004) experiment did not suggest that any definitive advantage was gained by using support vector machines for classification (compared to using ANNs for that purpose). The good results achieved in their experiment did demonstrate that the use of feature selection techniques—to reduce the noise in keystroke dynamics data—was beneficial. The results also re-affirm that

machine learning techniques (in general) perform better, than deterministic analysis methods, for pattern recognition tasks involving keystroke dynamics.

Jiang et al, (2007) used Gaussian modeling and Hidden Markov Models to perform data analysis for their experiment. The number of participants for their experiment was larger than for the current experiment, though the number of samples provided by each participant was much smaller.

As discussed in Chapter 3 section 3.4.1, Jiang et al, (2007) independently reduced the FAR and FRR by adjusting the decision threshold. Recall that adjusting one performance variable typically impacts (in the opposite direction) on the other variable. Their results were reasonably acceptable[10], with a FAR and FRR of approximately 0.0254 (2.54%). The current study achieved similar accuracy for the FAR as that attained by Jiang et al., (2007) (0.02766095 compared with 0.0254), but was slightly higher for the FRR (0.0862 compared with 0.0254).

Considering the methodological issues (the small number of samples per participant and the use of only one metric), and the results achieved by Jiang et al. (2007), the use of Gaussian Modeling and Hidden Markov Models to analyse and classify keystroke dynamics data did not appear to have achieved any definitive advantage over other methods. The current study did not suffer from the same methodological issues, and yet performed as well by using normality statistics for data filtering and ANNs for classification.

***Comparison to Past Studies Using Artificial Neural Networks***

A comparison between the results achieved in the current study and those achieved by research efforts that utilised Artificial Neural Networks (ANNs) for analysis of keystroke dynamics data will now be considered. Table 7.6 presents a summary of the results achieved by research efforts that adopted ANNs for analysis. The information in Table 7.6 is duplicated from Table 3.2 in Chapter 3 section 3.4.1, and is provided for convenient comparison between the results of the reviewed research and the results of the current study[11].

[10] As explained in Chapter 3 section 3.4.1, the results were attained from the average of the reported results, at the threshold value of 1.5.

[11] Note that a direct comparison between the experimental results of the publications listed in Table 7.6 (and also between them and the results of the current experiment) would be misleading. All of these results can only be viewed as an informal indicator of algorithmic performance because different data sets were used in the experiments.

| Reviewed Paper | FAR | FRR |
|---|---|---|
| Brown and Rodgers, 1993 | 0.0 | 0.115 |
| Obaidat and Sadoun, 2004 | 0.0 | 0.0005 |
| Cho et al., 2000 | 0.0 | 0.01 |
| Abernethy et al., 2004 | 0.019 | 0.108 |
| Revett et al., 2007 | 0.0195 | 0.0195 |
| **Current Study, 2010** | **0.02766** | **0.0862** |

Table 7.6: Summary of Reviewed Papers Using Artificial Neural Networks

Brown and Rogers (1993) were the first to conduct experimentation using ANNs for recognition of typing patterns. They used a Kohonen network to identify outlier data in input samples, and used the Single Layer Perceptron (SLP), the Multi-Layer Perceptron (MLP), and a distance measure to recognise typing patterns.

The reported results were a FAR of 0.0 for all three classifiers, with a FRR of 0.149 (14.9%) for the distance measure, 0.174 (17.4%) for the SLP, and 0.115 (11.5%) for the MLP.

As discussed in Chapter 3 section 3.4.1 the possible reasons for the less than impressive FRR scores were:

- Forcing the FAR to 0.0 would negatively impact on the FRR scores, by driving that rate higher. This practice was avoided in the current study.
- SLPs do not employ error back propagation when updating network weight values. Because of this, the SLPs are limited in their ability to recognise complex patterns in high dimensional data.
- Only 10 samples were available when testing for false rejection—measured by the FRR—which resulted in a course granularity for that performance variable.
- Approximately 8 characters only were existent in the sample string entered by participants, which may have had an impact on the classifiers performance.

In the current study, the FAR of 0.02766095 (whilst not ideal) could be acceptable in a general authentication setting. This computes to approximately 290 samples falsely classified (out of 10,500 samples available for testing false acceptance). Though not as good a result when compared to the Brown and Rogers (1993) FAR of 0.0, it was still a reasonable result (given that the performance variable was not forced to a particular rate).

The FRR of 0.0862 achieved in the current study, was more accurate than those achieved by Brown and Rogers (1993). Even given the above conditions which could have negatively impacted on the FRR scores achieved by Brown and Rogers (1993), the FRR of 0.0862 achieved in the current study (which computes to approximately 9 genuine samples out of 100 falsely rejected) was a comparatively good result.

Obaidat and Sadoun (1997) achieved excellent results in their experiment. As reported in Chapter 3 section 3.4.1, the best performing classifiers were the Multi-Layer Perceptron with back-propagation (MLP-BP), the Fuzzy ARTMAP (FA), the Radial Based Function Network (RBFN), and the Learning Vector Quantisation network (LVQ); their respective FAR and FRR results were (0.0, 0.0005), (0.0, 0.0), (0.0, 0.0), and (0.0, 0.0).

As the current study utilised the MLP-BP, a comparison between the results achieved in both studies—for that classifier—will be made. Obviously, the FAR of 0.0 and the FRR of 0.0005 (0.05%) achieved by Obaidat and Sadoun (1997) were better numerical results than those achieved in the current study (with a FAR of 0.02766095 and a FRR of 0.0862).

In the Obaidat and Sadoun (1997) experiment, each of 15 participants provided 15 samples only to try to impersonate the other 14 participants; that is, there were only 210 (14x15) samples available (for each participant) when testing for false acceptance. Given this, and the fact that there were only 15 participants who provided samples, it raises the question of whether the same results could be achieved given a larger number of participants and given a larger number of samples for testing false acceptance. In the current study, there were 10,500 samples (for each of the 50 training group members) available for testing false acceptance.

Also though not stated by the authors, it could be assumed that with only 15 participants, the ANN would have been exposed to the data (during the training process) provided by all 15 participants. This means that the trained ANN would have been tested on data it had already seen; generalising on the results of such a training and testing regime may bring into question the achieved accuracy.

For the current study, 5,600 (40x140) of the 10,500 samples were supplied by non-training group members. This meant the ANNs had not been exposed to these data prior to testing, and thus some level of generalisability can legitimately be claimed. Though it is unwise to generalise too much with only 50 training group members, the current study should offer reasonable confidence in its FAR result.

According to their report, Obaidat and Sadoun (1997) had each of the 15 participants provide 9,000 genuine samples; they used 4,500 samples per participant in the training process. This meant that they had 4,500 samples available to test positive recognition. This number of samples would exhibit a very fine granularity for the FRR performance variable, and so the result could be accepted with confidence.

The current study used 100 samples when testing false rejection, which is a small number in comparison to 4,500. Whilst the FRR performance variable in the current study did not achieve a similar granularity to that of Obaidat and Sadoun (1997), it did achieve reasonable level of accuracy.

Cho et al., (2000) used the Multi-Layer Perceptron as an auto-associator (MLP-AA) rather than in the typical fashion. Their results—a FAR of 0.0 and a FRR of 0.01 (1.0%)—again were very impressive. However as discussed in Chapter 3 section 3.4.1, there were validity issues with the experiment. As with many experiments in this field, a small number of participants were recruited. Of more concern was that data collection was unsupervised. In the current study, all data collection was supervised so no opportunity existed for subversion of the input process, and thus input data could not be compromised.

For their experiment, Cho et al., (2000) had each of the 21 participants provide an average of 275 samples to train the ANN (the last 75 were used to test for positive recognition). They also recruited 15 impostors to try to impersonate each of the 21 original participants (by providing 75 samples of each password); that is, there were 1,125 (15x75) samples available when testing for false acceptance. Again, it may be questionable if the same results could be achieved given a larger number of participants and given a larger number of samples for testing false acceptance. In the current study, there were 10,500 samples (for each of the 50 training group members) available for testing false acceptance.

Thus the granularity of the FAR performance variable in the current study was ten times finer than that of the Cho et al., (2000) study (i.e. 10,500 impostor tests per participant compared to 1,125 impostor tests per participant).

The finer granularity of the FAR in the current study could be a reason for the higher rate achieved when compared to both the Obaidat and Sadoun (1997) and the Cho et al., (2000) studies. Even so, the rate achieved in the current study could still be considered a reasonably acceptable rate in some circumstances.

The author and his collaborators, in an earlier study, also used the Multi-Layer Perceptron with back-propagation (MLP-BP) as a classifier (Abernethy et al., 2004). For the experiment, 50 participants were recruited; 25 were randomly assigned to a training group with the remaining 25 assigned to a non-training group. Each participant provided 40 samples of the required text; 30 were randomly selected for training an ANN, and the remaining 10 used for testing purposes.

For the current study, 90 participants were recruited; 50 were randomly assigned to a training group with the remaining 40 assigned to a non-training group. Each participant provided 140 samples of the required text; 30 were randomly selected for training an ANN, 10 were randomly selected for validation during training, and the remaining 100 used for testing purposes.

The number of samples available for testing false acceptance (per participant) in the previous experiment was 1,240. The number of samples available for testing false rejection (per participant) was 10. Whilst the quantity for testing false acceptance may be acceptable in some circumstances, the 10 for testing false rejection resulted in a very course granularity for the FRR performance variable.

In the current study, there were 10,500 samples available for testing false acceptance (per participant) and 100 samples available for testing false rejection (per participant). These quantities should invoke more confidence in the results achieved in the current study.

A primary purpose for the previous experiment was to determine the optimal character string length for use in keystroke dynamics authentication. It was determined that the optimal character string length was fifteen; the results at that length were a FAR of 0.0119 (1.19%) and a FRR of 0.108 (10.8%).

However because of limitations noted in the previous experiment, some uncertainty could be attributed to the results achieved. Even though the FAR of 0.02766095 achieved in the current study was less accurate than that achieved in the previous experiment, the author has much more confidence in the current results.

Apart from the greater number of samples available for testing, the current study also performed pre-treatment of the raw typing data before exposing the input to the ANNs. The pre-treatment took the form of feature selection to reduce the noise in raw data. As was evident, from the previous experiments reviewed, feature selection has demonstrated a positive impact on results.

Revett et al., (2007) used primary and secondary metrics to represent typing patterns, and the Probability Neural Network (PNN) to recognise patterns. As explained in Chapter 3 section 3.4.1, no discrete values were provided—for the two performance variables (FAR/FRR)—to report the results; rather the results were reported as a combined average of 0.039 (3.9%).

However, to compare results with other experiments, it was deemed necessary to be in possession of two discrete values for these performance variables. So for discussion purposes, the reported average has been evenly divided into two to yield a FAR of 0.0195 (1.95%) and a FRR of 0.0195 (1.95%)[12]. With a FAR of 0.02766095 and a FRR of 0.0862, the current study did not achieve the same level of accuracy as the Revett et al., (2007) experiment.

Though the number of participants (legitimate users) for the Revett et al., (2007) experiment was smaller than that for the current study, the number of samples supplied did provide a reasonable number for testing. Given the available data, and the reported results, the experiment did not meet similar accuracy to those of other experiments that used ANNs for classification (excluding the current study). It is unlikely that the PNN would be responsible for the loss of accuracy; as stated by the authors, PNNs have proved to be a reliable classifier. A possible cause may be the primary and secondary metrics used to represent the typing patterns.

[12]Though there may be other methods to isolate discrete values from the combined average, the simplest has been adopted here.

#### 7.2.1.1 Summary of Keystroke Dynamics Results

From the previous discussions, the following can be observed in relation to the experimental results achieved in the current study for the keystroke dynamics phase:

- The average FAR of 0.02766095 (2.766%) meant that on average approximately 290 impostor samples (out of 10,500) were incorrectly accepted. Whilst this average result was not as accurate as some other research efforts (particularly those that used ANNs for classification), in non-mission critical applications it may be acceptable. The results demonstrated that a reasonable level of accuracy is achievable for keystroke dynamics, which could allow for its application in a multi-modal authentication system.

- The average FRR of 0.0862 (8.62%) meant that on average approximately 9 genuine samples (out of 100) were incorrectly rejected. This was a less than ideal achievement, even for non-mission critical applications.

- In comparison to the reviewed research efforts that utilised deterministic statistical analysis methods, the current study generally achieved better results (if methodological concerns with those reviewed research efforts are taken into account). The exception could be the very good results achieved by Bergadano et al. (2002), although some areas of concern were still apparent.

- In comparison to the reviewed research efforts that utilised machine learning techniques (other than ANNs), the current study generally achieved similar accuracy (again, if methodological concerns with those reviewed research efforts are taken into account). Exceptions could be the impressive results achieved by Yu and Cho (2004) and Hu et al. (2008), although some areas of concern were apparent in the Yu and Cho (2004) experiment.

- In comparison to the reviewed research efforts that utilised Artificial Neural Networks, the current study generally achieved less accuracy. The reason for the less than accurate results is unknown; it is possible that the feature selection method adopted in the current study was not as successful as anticipated.

When considering the studies involving ANNs, there are two points that need clarification and emphasis:

1. The accepted approach when testing ANNs is to maintain some data sets consisting of participants' samples that the ANNs has not been exposed to during training. This provides an indication of the ability to generalise based on the results attained. Brown and Rogers (1993), Obaidat and Sadoun (1997), Cho et al. (2000), and Revett et al., (2007) used data for testing that the ANN had previously been exposed to during training. In the current study, data from the 40 non-training group members were set aside for testing purposes only. Thus the trained ANNs had no prior exposure to this data, and so confidence in the results (to be generalisable) should be apparent.

2. In the previously mentioned research efforts, one ANN was trained to distinguish between all legitimate users. As pointed out by Cho et al., (2000) and Yu and Cho (2004), this approach has its limitations. If a new user needs to be added to the system, the whole ANN needs to be trained again (as the number of users increases, so does the time taken to train the ANN). Also, the more users that the ANN needs to differentiate between, the larger the problem space becomes and so maintaining accuracy becomes increasingly difficult. In the current study, each training group member had one ANN trained exclusively to recognise their typing pattern. Whilst some data from the other training group members was used to create negative case samples for the training process, each ANN was trained only to recognise one member. This occurred for all training group members.

Whilst the results for the keystroke dynamics phase of the current experiment were less accurate than other experiments that used ANNs to recognise typing patterns, they were generally as accurate—or more accurate—than other experiments that used non-ANN methods to recognise typing patterns.

The next section discusses the results of the fingerprint recognition phase of the experiment, and compares them with other research in that field.

### 7.2.2 Discussion Of Fingerprint Recognition Results

The statistics in Table 7.7 provide information regarding the calculation of the decision threshold, for all 50 training group members, for the fingerprint recognition phase of the experiment.

Figures in column 2 (the Area Under the ROC Curve) of Table 7.7 show that 47 of the 50 training group members achieved an area of 1. This was a very positive result, as an area of 1 demonstrates perfect recognition (at the nominated decision threshold). Those members with an area less than 1 were 23, 74, and 76, with areas of 0.999992, 0.999996, and 0.999999 respectively.

Other points of interest in Table 7.7 are:

- The confidence level for all 50 training group members was 0.5, which indicated that no member required an adjustment to their confidence level because of poor classifier performance.

- Members 23 and 76 had the highest decision threshold values of 0.97 and 0.98 respectively. However, it should be noted that there does not seem to be any evidence of a correspondence between the adjustable determinant value and the decision threshold for any member[13].

- The decision thresholds for the fingerprint recognition phase of the experiment were generally lower than those in the keystroke dynamics phase. The average for the decision thresholds was 0.4506 with a standard deviation of 0.37543667. Note that the average for this phase was more than half that of the keystroke dynamics phase (0.4506 compared to 0.9486), and the standard deviation for this phase was approximately 10 times that of the keystroke dynamics phase (0.37543667 compared to 0.03811958). Though the larger standard deviation—for this phase—suggests a greater variance in the distribution of the decision thresholds, there seems no reason to suspect any statistical implications about the distribution because the average of the decision thresholds was so much lower than that of the keystroke dynamics phase.

[13]Given the method used to calculate the adjustable determinant value (described in Chapter 6 section 6.3.4), no correspondence should be expected.

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99996294 | 0.0500 | 0.94996294 | 0.84 |
| 2 | 1 | 0.99996188 | 0.0500 | 0.94996188 | 0.95 |
| 3 | 1 | 0.99802990 | 0.0500 | 0.94802990 | 0.61 |
| 5 | 1 | 0.98105692 | 0.0500 | 0.93105692 | 0.02 |
| 7 | 1 | 0.99997253 | 0.0500 | 0.94997252 | 0.22 |
| 9 | 1 | 0.99998120 | 0.0500 | 0.94998120 | 0.20 |
| 12 | 1 | 0.99997238 | 0.0500 | 0.94997238 | 0.16 |
| 14 | 1 | 0.99976722 | 0.0500 | 0.94976722 | 0.39 |
| 16 | 1 | 0.99887867 | 0.0500 | 0.94887867 | 0.09 |
| 18 | 1 | 0.99969411 | 0.0500 | 0.94969411 | 0.64 |
| 20 | 1 | 0.99995251 | 0.0500 | 0.94995251 | 0.94 |
| 21 | 1 | 0.99995280 | 0.0500 | 0.94995279 | 0.20 |
| 23 | 0.99999200 | 0.99681243 | 0.0500 | 0.94681243 | 0.97 |
| 24 | 1 | 0.99949265 | 0.0500 | 0.94949264 | 0.81 |
| 25 | 1 | 0.99968960 | 0.0500 | 0.94968960 | 0.94 |
| 27 | 1 | 0.99989653 | 0.0500 | 0.94989653 | 0.85 |
| 29 | 1 | 0.99997006 | 0.0500 | 0.94997006 | 0.04 |
| 32 | 1 | 0.99993341 | 0.0500 | 0.94993341 | 0.03 |
| 34 | 1 | 0.99980600 | 0.0500 | 0.94980600 | 0.59 |
| 36 | 1 | 0.99996920 | 0.0500 | 0.94996920 | 0.67 |
| 38 | 1 | 0.99799319 | 0.0500 | 0.94799319 | 0.35 |
| 40 | 1 | 0.99994016 | 0.0500 | 0.94994016 | 0.05 |
| 41 | 1 | 0.99996204 | 0.0500 | 0.94996204 | 0.05 |
| 43 | 1 | 0.99993914 | 0.0500 | 0.94993914 | 0.03 |
| 45 | 1 | 0.99988545 | 0.0500 | 0.94988545 | 0.08 |
| 46 | 1 | 0.99994543 | 0.0500 | 0.94994543 | 0.90 |
| 47 | 1 | 0.99988648 | 0.0500 | 0.94988648 | 0.11 |
| 49 | 1 | 0.99985759 | 0.0500 | 0.94985759 | 0.95 |
| 52 | 1 | 0.99988577 | 0.0500 | 0.94988577 | 0.94 |
| 54 | 1 | 0.99997126 | 0.0500 | 0.94997126 | 0.16 |
| 56 | 1 | 0.99995320 | 0.0500 | 0.94995320 | 0.66 |
| 58 | 1 | 0.99990624 | 0.0500 | 0.94990624 | 0.93 |
| 60 | 1 | 0.99997203 | 0.0500 | 0.94997203 | 0.10 |
| 61 | 1 | 0.99997752 | 0.0500 | 0.94997752 | 0.02 |
| 63 | 1 | 0.99993195 | 0.0500 | 0.94993195 | 0.93 |
| 65 | 1 | 0.99995170 | 0.0500 | 0.94995170 | 0.30 |
| 67 | 1 | 0.99994074 | 0.0500 | 0.94994074 | 0.22 |
| 68 | 1 | 0.99997243 | 0.0500 | 0.94997243 | 0.87 |
| 69 | 1 | 0.99995608 | 0.0500 | 0.94995608 | 0.04 |
| 72 | 1 | 0.99986472 | 0.0500 | 0.94986472 | 0.01 |
| 74 | 0.99999600 | 0.99571402 | 0.0500 | 0.94571402 | 0.93 |
| 76 | 0.99999900 | 0.99910068 | 0.0500 | 0.94910068 | 0.98 |
| 78 | 1 | 0.99997427 | 0.0500 | 0.94997427 | 0.75 |
| 80 | 1 | 0.98823433 | 0.0500 | 0.93823433 | 0.02 |
| 81 | 1 | 0.95769181 | 0.0500 | 0.90769181 | 0.41 |
| 83 | 1 | 0.99989038 | 0.0500 | 0.94989038 | 0.41 |
| 85 | 1 | 0.99995190 | 0.0500 | 0.94995190 | 0.19 |
| 87 | 1 | 0.99995996 | 0.0500 | 0.94995996 | 0.93 |
| 89 | 1 | 0.99995232 | 0.0500 | 0.94995232 | 0.03 |
| 90 | 1 | 0.99069618 | 0.0500 | 0.94069618 | 0.02 |
| **Average** | 0.99999974 | 0.99801224 | – | 0.94801224 | 0.4506 |
| **SD** | 0.00000123 | 0.00672410 | – | 0.00672410 | 0.37543667 |

Table 7.7: Duplication of Fingerprint Recognition Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.95 | 0.05 | 5 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 0.95 | 0.05 | 5 | 0 | 0 |
| 76 | 0.99 | 0.01 | 1 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| **Average** | 0.9978 | 0.0022 | 0.22 | 0 | 0 |
| **SD** | 0.009957 | 0.009957 | – | 0 | – |

Table 7.8: Duplication of Fingerprint Recognition Results

The individual training group members' results, demonstrated in Table 7.8, indicate that all training group members achieved a FAR of 0.0. This meant that all training group members had no impostor samples—out of 10,500—accepted as their own. This was an excellent result, and reflective of the AUC figures presented in Table 7.7.

Table 7.8 also showed that only three training group members registered a non-zero FRR. Members 23 and 74 had the highest FRR of 0.05 and member 76 had a FRR of 0.01. This meant that members 23 and 74 had 5 samples each (of their 100 genuine samples) incorrectly rejected, and member 76 had only 1 genuine sample incorrectly rejected. Note that members 23, 74 and 76 were the only members to register AUC values less than 1. As an area of 1 reflects perfect recognition (at the nominated decision threshold), the non-zero FRR for these members verifies the less than perfect performance.

The last two rows of Table 7.8 provide the average and standard deviation FAR and FRR figures, for all training group members, for the fingerprint recognition phase of the experiment. The average FAR was 0.0 with a standard deviation of 0.0, and the average FRR was 0.0022 with a standard deviation of 0.009957. These figures demonstrated that the fingerprint recognition phase of the experiment performed extremely well.

It meant that on average, there was approximately a 2 in 1,000 chance that any of the 50 training group members would have one of their own genuine samples incorrectly rejected. Also, there was no chance that any training group member would have an impostor sample (out of 10,500) incorrectly accepted as their own.

### Comparison with Reviewed Fingerprint Recognition Research

The average FAR and average FRR, for the 50 training group members, provide figures that permit a comparison with the results of the research efforts reviewed in Chapter 4 section 4.5. Even though the majority of those research efforts designed experiments based on a uni-modal approach—as opposed to the current study which was designed to facilitate a multi-modal approach—it is still feasible to compare other research results with those of the current study, as in most cases the same performance variables (that is, the FAR and the FRR) were used[14].

[14] As noted in 4 section 4.5, numerous authors expressed the performance variables as a percentage. That is, the percentage of the FAR and the percentage of the FRR. However in Table 4.5 columns 8 and 9 (Chapter 4 section 4.5), the figures for the performance variables are denoted as their actual rates (i.e. the percentage divided by 100). Therefore, any discussion comparing the performance variables will denote both the actual rate and the percentage rate (in parentheses).

Table 7.9 presents a summary of the fingerprint recognition results achieved by the reviewed research efforts. The information is duplicated from Table 4.5 in Chapter 4 section 4.5, and is provided for convenient comparison between the results of the reviewed research and the results of the current study[15].

| Reviewed Paper | FAR | FRR |
|---|---|---|
| Jain et al., 1997 | na | 0.16 |
| Luo et al., 2000 | na | 0.133 |
| Jiang and Yau, 2000 | 0.0 | 0.0997 |
| Lee et al., 2002 | 0.0002 | 0.1666 |
| He et al., 2003 | 0.0001 | 0.045 |
| Tong et al., 2005 | 0.00001 | 0.07 |
| Qi and Wang, 2005 | 0.0325 | 0.0605 |
| Jie et al., 2006 | 0.00001 | 0.001 |
| Ozkaya et al., 2006 | 0.03158 | 0.015 |
| Kumar and Deva Vikram, 2010 | 0.0113 | 0.015 |
| **Current Study, 2010** | **0.0** | **0.0022** |

Table 7.9: Summary of Fingerprint Recognition Results For Reviewed Papers

To begin with, Jain et al., (1997) and Luo et al., (2000) did not use both of the above mentioned performance variables, but used what they termed the verification rate and the reject rate. As explained in Chapter 4 section 4.5, the verification rate appears to be analogous to the precision measurement (also known as the positive prediction value) that was defined by Equation 6.7 in Chapter 6 section 6.2.1. As such, no comparative inference between the verification rate and the FAR can be drawn, and therefore the verification rates have not been reported in Table 7.9 column 2. The reject rate appears to be analogous to the false rejection rate (FRR), and therefore the reject rates reported by the authors have been registered in Table 7.9 column 3.

As the findings of the current experiment were calculated as the FAR and the FRR performance variables, only the FRR results can be compared with those of Jain et al., (1997) and Luo et al., (2000). The authors achieved FRR results of 0.16 (16.0%) and 0.133 (13.3%) respectively. The FRR of 0.0022 achieved in the current experiment demonstrated a more accurate performance than that achieved by Jain et al., (1997) and Luo et al., (2000).

[15]Note that a direct comparison between the experimental results of the publications listed in Table 7.9 (and also between them and the results of the current experiment) would be misleading. All of these results can only be viewed as an informal indicator of algorithmic performance because different data sets were used in the experiments.

Some of the notable differences between the Jain et al., (1997) and Luo et al., (2000) experiments and the current experiment were:

- The participants in the experiments conducted by the other researchers provided only 10 samples each. This meant that there were less samples upon which to base the pattern recognition task. It also meant that there were only 10 samples available to test for false rejection. If their methodologies were applied to a larger population, it may raise concerns about the achievements. In the current experiment, 140 samples were provided by each participant with 40 samples used for the pattern recognition task and 100 samples used to test for false rejection.

- The other researchers used 3 and 4 attributes respectively to represent local fingerprint features. In the current experiment, 6 local feature attributes were utilised. As discussed in Chapter 5 section 5.5.5, to the author's knowledge no other researchers have used this number of local feature attributes.

- The other researchers incorporated feature set alignment into the overall minutiae matching process, but prior to actual matching. This is a typical approach for a uni-modal system involving fingerprint recognition. For their experiments, they used additional ridge information to help with the alignment. In the current experiment, feature set alignment was treated as a separate process because prior transformation of the data sets was required for phase 3 of the experiment (that is, the data fusion phase).

- In the other researchers' experiments, analysis was performed using string matching algorithms, and a matching score was determined; this was used as a basis for the final verification decision. The use of a matching score is a typical approach for uni-modal systems or for data fusion systems operating at the confidence score level. For the current experiment, analysis was performed by ANNs; as the ANNs output probability scores, applying final decision threshold values to this score determined classification (thus no matching score was required).

Jiang and Yau (2000) used local and global fingerprint structures for the alignment of fingerprint feature sets. Though the actual alignment method used by these authors was different to that used in the current experiment, the two methods had a similar approach in common: local alignment was used to find candidate transformation factors that were then applied to a global alignment process for refinement. Again, for the current experiment this process was performed as a pre-processing step.

Jiang and Yau (2000) performed analysis by calculating a matching certainty level, based on the comparison of features that had been converted to a polar coordinate system. As a result of the comparison, a normalised matching score was determined.

The authors presented their results as a FAR of 0.0 and a FRR of 0.0997 (9.97%). The FAR result achieved in the current study was the same as that achieved by Jiang and Yau (2000), and the FRR of 0.0022 (in the current study) was more accurate than their FRR result of 0.0997.

Even though Jiang and Yau (2000) recruited 188 participants, they were required to provide only 8 samples each. This meant that their were only 8 samples available for testing false rejection, and 1,496 (187x8) samples (per participant) for testing false acceptance. Again these are much reduced numbers compared to the current study—where 100 samples (per participant) were used to test for false rejection and 10,500 samples (per participant) were used to test for false acceptance—and could raise concerns about the results achieved if their methodology was applied to a larger population.

Lee et al., (2002) used the average ridge frequency to determine distances between minutiae in a fingerprint. The normalised average distance was used in the alignment process (based on local feature structures), as well as to determine an adaptive bounding box used in the matching process. The bounding box (applied to the reference sample minutiae) was used to determine minutiae correspondences between the reference and query fingerprints.

Results derived from ROC graphs presented by the authors were a FAR of 0.0002 (0.02%) and a FRR of 0.1666 (16.66%). The FAR of 0.0 and the FRR of 0.0022 achieved in the current study, demonstrated better accuracy than that achieved by Lee et al., (2002).

In their experiment, Lee et al., (2002) recruited 100 participants who provided 10 samples each. Thus, there were only 10 samples (per participant) available for testing false rejection, and 990 (99x10) samples (per participant) for testing false acceptance. As previously discussed, this may be an inadequate number of classification tests to demonstrate full confidence in the findings.

He et al., (2003) utilised a similar alignment method to that proposed by Luo et al., (2000). The authors included the minutia type with the other three local feature attributes (x and y coordinates and orientation).

For their matching process, He et al., (2003) incorporated bounding box calculations into the converted polar coordinates of the registered feature set. By comparing these to other possible transformation sets (converted to the polar coordinate system), a matching result for each set was determined. The maximum matching result that also satisfied a threshold condition determined whether the query fingerprint was accepted as a match to the registered fingerprint.

The authors tested their methodology on four sections of the FVC2000 Fingerprint database, but provided no information about the number of participants or the number samples (per participant) used for their experiment.

Results were presented (via ROC graphs) for the four database sections tested. From the ROC graphs, the best results (recorded for database DB2_a) were a FAR 0.0001 and FRR 0.045. The results of the current study demonstrated better accuracy than the He et al., (2003) experiment, but a proper determination can't be made as the number of participants and the number of samples per participant in their study is unknown.

Tong et al., (2005) proposed an innovative method for aligning and matching fingerprints, by defining adjacent feature vectors (explained in Chapter 4 section 4.5). The method incorporated four local feature attributes (x and y coordinates,

orientation, and minutia type). Once a query sample was aligned with the reference sample, a similarity level[16] was used to determine a normalised matching score.

The authors tested their methodology on three sections of the FVC2000 Fingerprint database, but provided no information about the number of participants or the number samples (per participant) used for their experiment.

From the ROC graphs used to present the findings, the best results (recorded for database DB2_a) were a FAR 0.00001 and FRR 0.07. The results of the current study demonstrated better accuracy than the Tong et al., (2005) experiment, though again a proper determination can't be made as the number of participants and the number of samples per participant in their study is unknown.

Qi and Wang (2005) used three local feature attributes to represent a minutia. The authors proposed a method of alignment that incorporated relative orientation based on angle differences (refer Chapter 4 section 4.5).

Matching involved firstly using a similarity level to identify reference minutia in both registered and query feature sets. Then utilising three minutiae in close proximity to form a triangle (in both feature sets), a triangular similarity level (based on the distance and orientation differences of the vertices) was calculated. If these similarity levels met a pre-defined threshold, the minutiae were designated corresponding minutiae.

The orientation fields (a global structure) of both fingerprints were compared, and a similarity level determined. A normalised matching score was then determined, which incorporated the triangular similarity level and the orientation field similarity level.

For their experiment the authors used the FVC2002 Fingerprint database (section DB_3) comprising 100 participants, who each provided 8 sample fingerprints. For the purpose of comparing results from the current study to those achieved by Qi and Wang (2005), approximate values have been derived from the ROC curve from Figure 4 in their report. These were a FAR of 0.0325 (3.25%) and a FRR of 0.0605 (6.05%). The results from the current study achieved better accuracy than those achieved by Qi and Wang (2005).

[16]Based on Equation 4.10 Chapter 4 section 4.5

Jie et al., (2006) used the core point of a fingerprint as the centre of a circle containing potential reference minutiae, which were represented by four local feature attributes; the x and y coordinates, the orientation, and the minutia type. The following processing was applied to both registered and query feature sets. The features were converted to the polar coordinate system, and the radial angles were sorted in ascending order.

Matching involved the use of a 'circular bounding box'. If a minutia in the query feature set fell within the bounded region of a minutia in the registered feature set and met a threshold condition, a correspondence was recorded (via a matching score). After attempting all combinations of minutiae in the bounded region, the largest matching score that exceeded the decision threshold condition indicated a match.

For their experiment Jie et al., (2006) recruited 100 participants, who each provided 11 sample fingerprints. Each sample was tested for correct recognition against the other ten samples provided by the same participant. This meant that the FRR had a course granularity. Each sample was tested for false acceptance against the 1,089 (i.e 1,100-11) provided by the other participants. These test numbers are approximately one tenth of the number used in the current study, when the respective performance variables were determined.

Results presented by Jie et al., (2006) were a FAR of 0.00001 (0.001%) and a FRR of 0.001 (0.1%). The FAR result achieved in the current study was only minutely more accurate than that achieved by Jie et al., (2006); 0.0 compared to 0.00001. However, Jie et al., (2006) achieved a marginally more accurate FRR than the current study; 0.001 compared to 0.0022.

Like Jie et al. (2006), Ozkaya et al. (2006) used the core point as the centre for a circular bounded region for the alignment process. However, their method did not utilise the polar coordinate system, but rather the 2D coordinate locations of minutiae within the bounded region.

A similarity level was determined based on minutiae correspondences between the query feature set and the bounded regions of the registered feature set. The largest

similarity level from testing possible transformations (as above) was compared to a decision threshold to determine a match.

For their experiment Ozkaya et al. (2006) recruited 20 participants, who each provided 5 sample fingerprints. Each sample was tested for correct recognition against the other four samples provided by the same participant. This meant that the FRR had a very course granularity. Each sample was tested for false acceptance against all other 95 samples (provided by the other participants). Again, this meant a very course granularity for the FAR. These test numbers are well below the number used in the current study, when the respective performance variables were determined.

Results presented by Ozkaya et al., (2006) were a FAR of 0.03158 (3.158%) and a FRR of 0.015 (1.5%). These results were less accurate than those achieved in the current study, with a FAR of 0.0 (compared to 0.03158) and a FRR of 0.0022 (compared to 0.015).

Kumar and Deva Vikram (2010) developed an alternative feature representation for the matching purposes. This method did not require alignment, and used only two local feature attributes: the x and y coordinates.

The method involved locating minutiae in the gray-scale fingerprint image, reducing the pixilated image to a 15 x 15 matrix, and summing the intensity values of the minutiae in each matrix cell (if there were minutiae present). The values for each matrix cell were normalised to the continuous interval $[0, 1]$.

Analysis was performed by the multi-layer back propagation neural network. Input to the ANN was the result of the above matrix representation (i.e. 15x15 or 225 input values per sample).

For the experiment, 3,500 individuals provided three fingerprint scans each of the same finger (i.e. 10,500 samples). Two of the three samples provided by each of 3,000 individuals were used to train the ANN (i.e. 6,000 samples). Thus there were 4,500 samples remaining to test the trained ANN (only 500 of these patterns had not been exposed to the ANN during training).

Results presented by Kumar and Deva Vikram (2010) were a FAR 0.0113 (1.13%) and a FRR 0.015 (1.5%). These results were less than satisfactory, and were less accurate than those achieved in the current study.

Some differences between the Kumar and Deva Vikram (2010) experiment and the current study were:

- The other researchers used one ANN to authenticate fingerprints from 3,000 different individuals. This approach has obvious scalability ramifications, which could also lead to inaccuracy as the number of individuals becomes larger (because the ANN must accurately distinguish between a larger number of different patterns). In the current study, each individual had an ANN trained to recognise their pattern only. This approach seems more manageable and scalable. It also stands to reason that confidence could be higher if one ANN had to distinguish only one pattern (from numerous others) rather than 3,000 or more (from among each other).

- Only 2 samples per individual were used when training the ANN in the Kumar and Deva Vikram (2010) experiment. This seems a very low number of positive case samples for training purposes, particularly in that 3,000 different patterns needed to be distinguished. In the current study, 30 positive case samples were used to train an ANN to recognise one individual's fingerprint pattern.

- Because each participant provided only 3 samples and 2 of these were used for training purposes (in the Kumar and Deva Vikram (2010) experiment), only 1 sample per individual was available for testing false rejection. This results in the most course granularity possible for the FRR performance variable. The current study had 100 samples (per participant) available for testing false rejection.

- Most significant of all, the representation method used by Kumar and Deva Vikram (2010) may not have preserved the uniqueness provided by the local feature configuration. The authors method resulted in loss of data due to the processing required to obtain the representation, and the uniqueness (of the

fingerprint features) may have been compromised; it was entirely possible that two fingerprints under comparison may have ended up with the same or a very similar representation.The representation method utilised in the current study (as described in Chapter 5 section 5.5.5) preserved the local feature configuration, and in fact used more attributes to denote a local feature than any known research effort.

#### 7.2.2.1 Summary of Fingerprint Recognition Results

From the previous discussions, the following can be observed in relation to the experimental results achieved in the current study for the fingerprint recognition phase:

- The average FAR of 0.0 meant that no impostor samples (out of 10,500) were incorrectly accepted for any of the training group members. This average result demonstrated perfect recognition. Importantly, it demonstrated that the representation method used in the fingerprint phase of the experiment—described in Chapter 5 sections 5.5.4 and 5.5.5—which was developed to facilitate feature level data fusion, achieved results better than many other research efforts. Thus this method could used for mission critical applications.

- The average FRR of 0.0022 (0.22%) meant that on average their was approximately a 2 in 1,000 chance that a training group member would have one of their genuine samples incorrectly rejected. This average result was comparable to those achieved by other research efforts, and again demonstrated the suitability of the fingerprint representation method for mission critical applications.

- In comparison to the reviewed research efforts that utilised minutiae matching techniques (i.e. those that did not use ANNs), the current study achieved better accuracy. The two reviewed studies that achieved the best FAR (of 0.00001) were those by Tong et al., (2005) and Jie et al., (2006); the FAR of 0.0 achieved in the current study was marginally more accurate than these research efforts. The reviewed study that achieved the best FRR (of 0.001)

was that by Jie et al., (2006); the FRR of 0.0022 achieved in the current study was only marginally less accurate than this result. Though it seems feasible to compare performance variables with these studies, it should be kept in mind that they developed minutiae matching techniques (for uni-modal authentication) that extended previous research in the area. The method developed in the current study deviated from the typical minutiae matching techniques to facilitate data fusion at the feature level. The results demonstrated that the developed method would be appropriate for either uni-modal or multi-modal authentication.

- In comparison to the reviewed research effort that utilised Artificial Neural Networks (for the analysis of fingerprints), the current study achieved more accuracy. Kumar and Deva Vikram (2010) achieved a FAR of 0.0113 and a FRR of 0.015; the current study achieved a FAR of 0.0 and a FRR of 0.0022. As discussed in the previous section, the Kumar and Deva Vikram (2010) study used a very different fingerprint feature representation method, which may have impacted on the accuracy they achieved (because of data loss). The representation method used in the current study was developed to maintain local feature information; as demonstrated by the results, this method returned better accuracy.

When comparing the current experiment with other research efforts in the field of fingerprint recognition, the differences in approach and intended purpose impact on the methodology utilised. As was discussed in the previous section, the majority of the reviewed studies adopted the minutiae matching approach.

This approach typically incorporates:

- Fingerprint feature alignment (based on minutia location), prior to attempted matching. The possible alignment methods vary widely, as was evident by the discussion in the previous section.
- The matching process, which involves identifying all correspondences between minutia pairs (in both registered and query feature sets). The possible matching methods vary widely, as was evident by the discussion in the previous section.

The minutiae matching approach adopted by the reviewed studies were intended for use in either a uni-modal authentication system or a multi-modal authentication system (where data fusion occurred at the confidence score or decision levels).

In the current study, the alignment process was performed as a pre-processing step. As described in Chapter 5 sections 5.5.4 and 5.5.5, the approach was then to select 8 local fingerprint features (incorporating 6 attributes per feature) which were used as inputs to an ANN.

The intended purpose of the experiment was to fuse (at the feature level) a fingerprint feature input vector with a keystroke dynamics input vector, with the resultant combined feature vector used as input to an ANN.

So, this phase of the experiment and the keystroke dynamics phase were preliminary phases to facilitate the data fusion phase. As the development of a new representation method for fingerprint features was required for feature level data fusion, this preliminary phase was conducted to demonstrate the accuracy and applicability of that method. The results demonstrated that the method could indeed be used in either a uni-modal authentication system or a multi-modal authentication system (and take advantage of the rich feature level data).

The next section discusses the results of the data fusion phase of the experiment, and compares them with other research in that field.

### 7.2.3 Discussion Of Data Fusion Results

In this section, a discussion of results for the data fusion phase of the experiment is presented. The results for the complementary data fusion paradigm are discussed in section 7.2.3.1, and the results for the cooperative data fusion paradigm are discussed in section 7.2.3.3. In section 7.2.3.3, The discussion includes the results of the four experimental stages conducted for the cooperative data fusion paradigm (that is, the four stages where different proportions of available data were used for data fusion[17]).

To recapitulate, their are three paradigms for data fusion (refer Chapter 2 section 2.3.1.1). Only the complementary and cooperative data fusion paradigms were investigated in this study. The competitive paradigm was not investigated because it authenticates based on a single (the most accurate) data source, which in effect models uni-modal biometric authentication. This was considered at variance with the intention of the current study, which was to investigate multi-modal biometric authentication.

#### 7.2.3.1 Complementary Data Fusion

In this section, a discussion is provided for the results of the complementary data fusion phase of the experiment (explained in Chapter 5 section 5.6.2). Recall that the complementary data fusion paradigm fuses all available data (i.e. 100%) from all sources. In the field of biometric authentication, this has typically been achieved by simply concatenating the data from all sources.

Firstly, the figures in Tables 7.10 and 7.11 are discussed in detail. Then a comparison is provided between the complementary data fusion results achieved in the current study and the results achieved by the research efforts discussed in Chapter 2 section 2.3.2.2. Note, only those research efforts that investigated the complementary data fusion paradigm will be discussed in this section.

The statistics in Table 7.10 provide information regarding the calculation of the decision threshold, for all 50 training group members, for the complementary data fusion phase of the experiment.

[17] Described in Chapter 5 section 5.6.3.

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99993349 | 0.0500 | 0.94993349 | 0.78 |
| 2 | 1 | 0.99993526 | 0.0500 | 0.94993526 | 0.88 |
| 3 | 1 | 0.99777526 | 0.0500 | 0.94777526 | 0.28 |
| 5 | 1 | 0.99968306 | 0.0500 | 0.94968306 | 0.08 |
| 7 | 1 | 0.99981626 | 0.0500 | 0.94981626 | 0.08 |
| 9 | 1 | 0.99996180 | 0.0500 | 0.94996180 | 0.25 |
| 12 | 1 | 0.99992290 | 0.0500 | 0.94992290 | 0.12 |
| 14 | 1 | 0.99961334 | 0.0500 | 0.94961334 | 0.36 |
| 16 | 1 | 0.99975571 | 0.0500 | 0.94975571 | 0.10 |
| 18 | 1 | 0.99979385 | 0.0500 | 0.94979385 | 0.92 |
| 20 | 1 | 0.99980667 | 0.0500 | 0.94980667 | 0.93 |
| 21 | 1 | 0.99997503 | 0.0500 | 0.94997503 | 0.53 |
| 23 | 0.99999429 | 0.99489361 | 0.0500 | 0.94489361 | 0.91 |
| 24 | 1 | 0.99843420 | 0.0500 | 0.94843420 | 0.11 |
| 25 | 1 | 0.99962849 | 0.0500 | 0.94962849 | 0.88 |
| 27 | 1 | 0.99986136 | 0.0500 | 0.94986136 | 0.94 |
| 29 | 1 | 0.99999791 | 0.0500 | 0.94999791 | 0.29 |
| 32 | 1 | 0.99987604 | 0.0500 | 0.94987604 | 0.05 |
| 34 | 1 | 0.99965469 | 0.0500 | 0.94965469 | 0.45 |
| 36 | 1 | 0.99997427 | 0.0500 | 0.94997427 | 0.95 |
| 38 | 1 | 0.99908305 | 0.0500 | 0.94908304 | 0.13 |
| 40 | 1 | 0.99996581 | 0.0500 | 0.94996581 | 0.29 |
| 41 | 1 | 0.99977032 | 0.0500 | 0.94977032 | 0.12 |
| 43 | 1 | 0.99995303 | 0.0500 | 0.94995303 | 0.04 |
| 45 | 1 | 0.99994619 | 0.0500 | 0.94994619 | 0.04 |
| 46 | 1 | 0.99995954 | 0.0500 | 0.94995954 | 0.69 |
| 47 | 1 | 0.99984224 | 0.0500 | 0.94984224 | 0.11 |
| 49 | 1 | 0.99988235 | 0.0500 | 0.94988235 | 0.79 |
| 52 | 1 | 0.99974277 | 0.0500 | 0.94974276 | 0.94 |
| 54 | 1 | 0.99996927 | 0.0500 | 0.94996927 | 0.09 |
| 56 | 1 | 0.99997403 | 0.0500 | 0.94997403 | 0.66 |
| 58 | 1 | 0.99969609 | 0.0500 | 0.94969609 | 0.92 |
| 60 | 1 | 0.99996602 | 0.0500 | 0.94996602 | 0.23 |
| 61 | 1 | 0.99967599 | 0.0500 | 0.94967599 | 0.03 |
| 63 | 1 | 0.99990192 | 0.0500 | 0.94990192 | 0.77 |
| 65 | 1 | 0.99990294 | 0.0500 | 0.94990294 | 0.05 |
| 67 | 1 | 0.99976500 | 0.0500 | 0.94976500 | 0.24 |
| 68 | 1 | 0.99986011 | 0.0500 | 0.94986011 | 0.89 |
| 69 | 1 | 0.99990128 | 0.0500 | 0.94990128 | 0.02 |
| 72 | 1 | 0.99992978 | 0.0500 | 0.94992978 | 0.02 |
| 74 | 1 | 0.99427257 | 0.0500 | 0.94427257 | 0.85 |
| 76 | 1 | 0.99594715 | 0.0500 | 0.94594715 | 0.10 |
| 78 | 1 | 0.99997000 | 0.0500 | 0.94997000 | 0.55 |
| 80 | 1 | 0.99866941 | 0.0500 | 0.94866941 | 0.06 |
| 81 | 1 | 0.94646377 | 0.0500 | 0.89646377 | 0.17 |
| 83 | 1 | 0.99981049 | 0.0500 | 0.94981049 | 0.27 |
| 85 | 1 | 0.99996275 | 0.0500 | 0.94996275 | 0.11 |
| 87 | 1 | 0.99978510 | 0.0500 | 0.94978510 | 0.95 |
| 89 | 1 | 0.99992310 | 0.0500 | 0.94992310 | 0.02 |
| 90 | 1 | 0.99125975 | 0.0500 | 0.94125975 | 0.02 |
| **Average** | 0.99999989 | 0.99822090 | – | 0.94822090 | 0.4012 |
| **SD** | 0.00000081 | 0.00765436 | – | 0.00765436 | 0.35812801 |

Table 7.10: Duplication of Complementary Data Fusion Statistics for Threshold Calculation

Figures in column 2 (the Area Under the ROC Curve) of Table 7.10 show that 49 of the 50 training group members achieved an area of 1. This was an extremely positive result, as an area of 1 demonstrates perfect recognition (at the nominated decision threshold). The member with an AUC value less than 1 was member 23 with an area of 0.999994.

Other points of interest in Table 7.10 are:

- The confidence level for all 50 training group members was 0.5, which indicated that no participant required an adjustment to their confidence level because of poor classifier performance.

- Members 36 and 87 had the highest decision threshold value of 0.95. Again, there does not seem to be any evidence of a correspondence between the adjustable determinant value and the decision threshold for any member.

- The decision thresholds for the complementary data fusion phase of the experiment were generally lower than those for both the fingerprint recognition and the keystroke dynamics phases. The average for the decision thresholds for the complementary data fusion phase was 0.4012 with a standard deviation of 0.35812801. These were both lower than those recorded for the fingerprint recognition phase (with an average of 0.4506 and a standard deviation of 0.37543667). The keystroke dynamics phase (with an average of 0.9486 and a standard deviation of 0.03811958) had a much higher average for the decision thresholds than the complementary data fusion phase, but a much lower standard deviation.

The individual training group members' results, demonstrated in Table 7.11, indicate that all training group members achieved a FAR of 0.0. This meant that no training group members had any impostor samples (out of 10,500) accepted as their own. This was an excellent result, and reflective of the AUC figures presented in Table 7.10.

Table 7.11 demonstrated that only one member registered a non-zero FRR; member 23 with a rate of 0.02. This meant that member 23 had 2 of 100 genuine samples incorrectly rejected. Recall that member 23 was the only member to register an AUC value less than 1. As an area of 1 reflects perfect recognition (at the nominated decision threshold), the non-zero FRR for member 23 verifies this less than perfect performance.

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.98 | 0.02 | 2 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| **Average** | 0.9996 | 0.0004 | 0.04 | 0 | 0 |
| **SD** | 0.002828 | 0.002828 | – | 0 | – |

Table 7.11: Duplication of Complementary Data Fusion Results

The last two rows of Table 7.11 provide the average and standard deviation FAR and FRR figures, for all training group members, for the complementary data fusion phase of the experiment. The average FAR was 0.0 with a standard deviation of 0.0, and the average FRR was 0.0004 with a standard deviation of 0.002828. These figures demonstrated that the complementary data fusion phase of the experiment performed exceptionally well.

It meant that on average, there was a 4 in 10,000 chance that any of the 50 training group members would have one of their own genuine samples incorrectly rejected. Also, there was no chance that any training group member would have an impostor sample (out of 10,500) incorrectly accepted as their own.

### Comparison with Reviewed Complementary Data Fusion Research

The average FAR and average FRR—for the 50 training group members—provide figures that permit a comparison with the results of the research efforts reviewed in Chapter 2 section 2.3.2.2, that utilised a complementary data fusion approach. Though the research efforts reviewed conducted experiments that varied in certain aspects of data fusion (for example: the number of modalities, the level of fusion, and the fusion method), it is still feasible to compare their results with those of the current study, as in most cases the same performance variables (that is, FAR and the FRR) were used[18].

Table 7.12 provides a summary of the results achieved by the research efforts that utilised a complementary data fusion approach. The information in Table 7.12 is duplicated from Table 2.2 in Chapter 2 section 2.3.2.2, and is provided for convenient comparison between the results of the reviewed research and the results of the current study[19].

| Reviewed Paper | Fusion Level | FAR | FRR |
|---|---|---|---|
| Jain et al., 1999 | Decision | 0.0001 | 0.14 |
| Chatzis et al., 1999 | Decision | 0.0039 | 0.0 |
| Ross et al., 2001 | Confidence Score | 0.0003 | 0.0178 |
| Wang et al., 2003 | Confidence Score | 0.0 | 0.0 |
| Nandakumar, 2008 | Confidence Score | 0.0001 | 0.009 |
| **Current Study, 2010** | **Feature** | **0.0** | **0.0004** |

Table 7.12: Summary of Reviewed Papers Using The Complementary Data Fusion Paradigm

[18]As noted in Chapter 2 section 2.3.2.2, numerous authors expressed the performance variables as a percentage. That is, the percentage of the FAR and the percentage of the FRR. However in Table 2.2 columns 7 and 8 (Chapter 2 section 2.3.2.2), the figures for the performance variables are denoted as their actual rates (i.e. the percentage rate divided by 100). Therefore, any discussion comparing the performance variables will denote both the actual rate and the percentage rate (in parentheses).

[19]Note that a direct comparison between the experimental results of the publications listed in Table 7.12 (and also between them and the results of the current experiment) would be misleading. All of these results can only be viewed as an informal indicator of algorithmic performance because different data sets were used in the experiments.

Though not stated, the nature of the fusion method, employed by Jain et al., (1999a), indicates that the complementary data fusion approach was adopted[20]. In the current study, two of the three possible fusion paradigms were investigated; the complementary data fusion approach and the cooperative data fusion approach. In this section, only the results of the complementary data fusion approach (achieved in the current study) will be compared with the results achieved by Jain et al., (1999a).

Jain et al., (1999a) combined the data of three modalities at the decision level. For this process, each matching module determined a confidence score; these were then passed to the corresponding decision modules, where individual accept/reject decisions were made. The individual decisions were then passed to the fusion module, where fusion was achieved using the joint class-conditional probability density function. In the current study, feature level fusion was applied. The key difference is that fusion at the feature level combines the richer feature data, whereas at the decision level only accept/reject decisions are fused.

For their experiment, Jain et al., (1999a) recruited 50 participants to compile a template database of 500 fingerprint samples, 450 facial image samples, and 600 speech samples. They then had 25 of the original 50 participants provide 15 more samples each (for each modality) to formulate a test database. Thus the test database comprised 15 genuine samples to test false rejection of each of the 25 test participants, and 360 (24x15) impostor samples to test false acceptance.

In the current experiment, 90 participants were recruited (50 randomly assigned to the training group and 40 to the non-training group) who each provided 140 samples for both modalities (keystroke samples and fingerprint samples). Thus the test database comprised 100 genuine samples to test false rejection of each of the 50 training group members, and 10,500 impostor samples to test false acceptance (for each of the 50 training group members).

Jain et al., (1999a) presented their results using ROC graphs, from which the best results (for the fusion of all three modalities) were derived; a FAR of 0.0001

[20] As all features were utilised in attaining the confidence scores for each matching module, and all local decisions (based on these scores) were then fused, complementary data fusion is implied.

(0.01%) and a FRR of 0.14 (14%). The current experiment achieved a FAR of 0.0 and a FRR of 0.0004 (0.04%) for the complementary data fusion phase. The results of the current study were therefore more accurate than those achieved by Jain et al., (1999a), and were achieved using a larger sample size, with more samples provided by each participant.

Another experiment to perform data fusion at the decision level was conducted by Chatzis et al., (1999). As explained in Chapter 2 section 2.3.2.2, the authors applied fused data to six different decision functions. The fused data was derived from three biometric characteristics to form five modalities. Different combinations of the five modalities were applied to the six decision functions.

As with Jain et al., (1999a), a complementary data fusion approach is assumed (though not specifically stated). In this section, only the results of the complementary data fusion approach (achieved in the current study) will be compared with the results achieved by Chatzis et al., (1999).

For their experiment, Chatzis et al., (1999) used samples from 37 participants from the "M2VTS multimedia face database". The samples consisted of four speech and image sequences (per participant) from video data, which thus provided 144 (4x36) impostor tests for each participant (an overall total of 5,328 impostor tests). However, the small number of samples used meant that there were only 4 genuine available to test false rejection. In the current experiment, 90 participants provided 140 samples for both modalities, which provided 10,500 impostor tests and 100 genuine tests for each training group member. This eventuated in an overall total of 525,000 (10,500 for each of the 50 training group members) impostor tests and 5,000 (100 for each of the 50 training group members) genuine tests.

Results were reported for all five modalities applied to all six decision functions. The best results were achieved when the morphological dynamic link architecture (MDLA), profile shape matching (PSM), and maximum signal processing (MSP) modalities were applied to the fuzzy data k-means (FDKM) decision function. For this combination the experiment achieved a FAR of 0.0039 and a FRR of 0.0.

The results of the current study (with a FAR of 0.0 and a FRR of 0.0004) were more accurate (particular the FAR of 0.0 compared with 0.0039) than those achieved by Chatzis et al., (1999), and were achieved using a larger sample size, with more samples provided by each participant. With only 4 genuine samples to each participant for false rejection, the FRR of 0.0 achieved by Chatzis et al., (1999) may not have been attainable if more genuine samples had been available for testing.

Ross et al., (2001) experimented with confidence score level data fusion. As explained in Chapter 2 section 2.3.2.2, the authors combined three modalities: facial recognition, fingerprint recognition, and hand geometry. Three fusion methods were investigated: the sum rule, decision trees, and linear discriminant analysis. In the current experiment, the feature vectors from two modalities were fused and classification was performed by ANNs.

Again, in the Ross et al., (2001) experiment a complementary data fusion approach is assumed (though not specifically stated). In this section, only the results of the complementary data fusion approach (achieved in the current study) will be compared with the results achieved by Ross et al., (2001).

For their experiment, Ross et al., (2001) recruited 50 participants, who provided 9 samples each for each of the modalities. Therefore, 450 genuine scores (true positives) and 22,050 impostor scores (false positives) could be generated. In the current experiment, there were 100 genuine scores and 10,500 impostor scores generated for each of the 50 training group members (an overall total of 5,000 genuine tests and 525,000 impostor tests).

The best results were achieved by Ross et al., (2001) when the simple Sum Rule was employed to fuse the confidence scores, with a FAR of 0.0003 (0.03%) and a FRR of 0.0178 (1.78%). The results demonstrate similar accuracy to the previous research efforts, with a very good FAR and an acceptable FRR. The results of the current study (with a FAR of 0.0 and a FRR of 0.0004) were more accurate than those achieved by Ross et al., (2001), and were achieved with more samples provided by each participant.

Wang et al., (2003) combined the confidence scores from two modalities: facial recognition and iris recognition. Again a complementary data fusion approach was assumed, as no feature selection was evident. The current study also utilised two modalities, and investigated a complementary data fusion approach.

Five facial image samples and five iris image samples were attained from 90 participants. The current study also collected data from 90 participants, however each participant provided 140 samples for both modalities.

Three methods were investigated to fuse and classify the data from both modalities: the weighted sum rule, Fisher discriminant analysis, and radial based function networks. Classification in the current study was performed by ANNs only (specifically the multi-layer perceptron with back propagation).

Best results were achieved by Wang et al., (2003) using the radial based function network; perfect performance (that is, a FAR and FRR of 0.0) was achieved at various threshold values. The results demonstrated that radial based function networks perform admirably for fusion and classification. The results of the current study (with a FAR of 0.0 and a FRR of 0.0004) were as accurate for the FAR and only marginally less accurate for the FRR.

Note that both experiments achieved their results with the same number of participants, however the current experiment collected more samples from each participant (140 compared to 5). Therefore, the Wang et al., (2003) methodology was tested using only 5 genuine samples and 445 (89x5) impostor samples per participant. As such, some doubt remains as to whether similar results could be attained if a larger number of samples were applied.

Nandakumar (2008) experimented with combining the confidence scores from two modalities. A complementary data fusion approach was assumed, as no feature selection was evident. For the experiment, two multi-modal databases which provided confidence or match scores were utilised. The NIST-BSSR1 multi-modal database (Partition 1) provided two fingerprint and two face scores from 517 participants. Whilst tests were performed on other partitions of that database (and the XM2VTS-Benchmark database), the tests for partition 1 returned the best results.

Gaussian Mixture Models (GMM) were employed to estimate score densities, and confidence score level fusion was achieved using the complete likelihood ratio test (CLRT). For the current experiment, these statistical techniques were not relevant because feature level fusion, rather than confidence score level fusion, was investigated. The feature level fusion approach requires different considerations (for example, data alignment and feature selection).

Best results were achieved using the GMM and the complete likelihood ratio fusion, tested on the NIST-BSSR1 multi-modal database (Partition 1); a FAR of 0.0001 (0.01%) and a FRR of 0.009 (0.9%). The results achieved by Nandakumar (2008) were not as accurate as those achieved in the current study. However, the Nandakumar (2008) results might still be considered well within acceptable requirements, depending on the nature of application (that is, mission critical or otherwise).

#### 7.2.3.2 Summary of Complementary Data Fusion Results

The results of the current study, for the complementary data fusion phase of the experiment, out-performed four of the five experiments discussed in the previous section (with the exception being the Wang et al., (2003) experiment).

As demonstrated in Table 7.12, the first five research efforts conducted data fusion at either the decision level or the confidence score (within the complementary data fusion paradigm). The current study conducted data fusion at the feature level.

As discussed in Chapter 2 section 2.3.2.1, the level at which data fusion is performed comes with specific requirements. Confidence score level fusion requires the combining of scalar values from different modalities. At that level, raw and feature data is not available. Therefore, classification based on confidence scores may not always perform as expected because of the loss of discriminative information.

Feature level fusion requires combining features that are a low level representation of raw data. Fusion at this level encounters problems associated with the unknown compatibility of features from different sources, and the size of combined feature vectors.

As shown in Table 7.12 and the preceding discussion, there seems minimal difference between the accuracy achieved at the two data fusion levels (i.e. decision or confidence score). Aside from the obvious methodology differences (associated with the fusion levels), one of the most glaring differences between the current study and the other reviewed research efforts was the smaller number of samples provided by participants in the reviewed experiments; this would obviously result in a smaller number of genuine and impostor tests. As a consequence, attributing a high level of confidence in the accuracy of the results achieved (in the reviewed experiments) may be questionable.

The next section presents a discussion of results when the cooperative data fusion approach was adopted.

#### 7.2.3.3 Cooperative Data Fusion

In this section, a discussion is provided for the results of the cooperative data fusion phase of the experiment (explained in Chapter 5 section 5.6.3). Recall that the cooperative data fusion paradigm requires determination of the specific features (from each of the data sources) that best represent a region or physical attributes/aspects of an object. Thus this approach necessitates feature selection.

As discussed in Chapter 5 section 5.6.3.1, the feature selection process that was adopted for the experiment resulted in four stages for this phase. These stages related to the use of 40%, 50%, 60%, and 70% of the available metrics respectively. The appropriate proportion of metrics were selected from all available metrics (i.e. 100%), as used in the complementary phase of the experiment.

For most participants, percentages above 70% and below 40% were not practicable for the experiment because (according the proportion levels determined) their were either not enough fingerprint feature metrics available to meet the required number of metrics, or the keystroke dynamic metrics were not represented at all. Percentages rounded to 10% were used for ease of calculation.

The figures in Tables 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, and 7.20 are discussed in detail in the following sections. Then a comparison is provided between the cooperative data fusion results achieved in the current study, and the results achieved by the research efforts discussed in Chapter 2 section 2.3.2.2. Note, only those research efforts that investigated the cooperative data fusion paradigm will be discussed in this section.

### Stage 1 (40%)

The statistics in Table 7.13 provide information regarding the calculation of the decision threshold, for all 50 training group members, for stage 1 of the cooperative data fusion phase of the experiment.

Figures in column 2 (the Area Under the ROC Curve) of Table 7.13 show that 47 of the 50 training group members achieved an area of 1. This was a very positive result. Those members with AUC values less than 1 were 23, 25, and 87, with areas of 0.99998095, 0.99995238, and 0.99995238 respectively.

Other points of interest in Table 7.13 are:

- The confidence level for all 50 training group members was 0.5, which indicated that no participant required an adjustment to their confidence level because of poor classifier performance.

- Participant 46 had the highest decision threshold value of 0.98.

- The decision thresholds for the cooperative data fusion phase of the experiment (at 40%) were generally similar to those for the complementary data fusion phase. The average for the decision thresholds was 0.4234 with a standard deviation of 0.34155473; the average was slightly higher, and the standard deviation slightly lower, than those recorded for the complementary data fusion phase.

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99991754 | 0.0500 | 0.94991754 | 0.78 |
| 2 | 1 | 0.99988399 | 0.0500 | 0.94988399 | 0.94 |
| 3 | 1 | 0.99436450 | 0.0500 | 0.94436450 | 0.04 |
| 5 | 1 | 0.99858813 | 0.0500 | 0.94858813 | 0.08 |
| 7 | 1 | 0.99995643 | 0.0500 | 0.94995643 | 0.27 |
| 9 | 1 | 0.99999629 | 0.0500 | 0.94999629 | 0.20 |
| 12 | 1 | 0.99997473 | 0.0500 | 0.94997473 | 0.18 |
| 14 | 1 | 0.99998118 | 0.0500 | 0.94998118 | 0.94 |
| 16 | 1 | 0.99956927 | 0.0500 | 0.94956927 | 0.03 |
| 18 | 1 | 0.99987230 | 0.0500 | 0.94987230 | 0.93 |
| 20 | 1 | 0.99997892 | 0.0500 | 0.94997892 | 0.43 |
| 21 | 1 | 0.99997063 | 0.0500 | 0.94997063 | 0.65 |
| 23 | 0.99998095 | 0.98141660 | 0.0500 | 0.93141659 | 0.77 |
| 24 | 1 | 0.99906967 | 0.0500 | 0.94906967 | 0.40 |
| 25 | 0.99995238 | 0.99994703 | 0.0500 | 0.94994703 | 0.80 |
| 27 | 1 | 0.99996674 | 0.0500 | 0.94996674 | 0.72 |
| 29 | 1 | 0.99999984 | 0.0500 | 0.94999984 | 0.48 |
| 32 | 1 | 0.99990204 | 0.0500 | 0.94990204 | 0.07 |
| 34 | 1 | 0.99987635 | 0.0500 | 0.94987635 | 0.42 |
| 36 | 1 | 0.99997806 | 0.0500 | 0.94997806 | 0.51 |
| 38 | 1 | 0.99997937 | 0.0500 | 0.94997937 | 0.90 |
| 40 | 1 | 0.99996609 | 0.0500 | 0.94996609 | 0.11 |
| 41 | 1 | 0.99990079 | 0.0500 | 0.94990079 | 0.03 |
| 43 | 1 | 0.99994124 | 0.0500 | 0.94994124 | 0.05 |
| 45 | 1 | 0.99991208 | 0.0500 | 0.94991207 | 0.29 |
| 46 | 1 | 0.99998315 | 0.0500 | 0.94998315 | 0.98 |
| 47 | 1 | 0.99996134 | 0.0500 | 0.94996134 | 0.50 |
| 49 | 1 | 0.99992628 | 0.0500 | 0.94992628 | 0.87 |
| 52 | 1 | 0.99993853 | 0.0500 | 0.94993853 | 0.92 |
| 54 | 1 | 0.99996554 | 0.0500 | 0.94996554 | 0.04 |
| 56 | 1 | 0.99995322 | 0.0500 | 0.94995322 | 0.57 |
| 58 | 1 | 0.99993582 | 0.0500 | 0.94993582 | 0.94 |
| 60 | 1 | 0.99997868 | 0.0500 | 0.94997868 | 0.31 |
| 61 | 1 | 0.99996597 | 0.0500 | 0.94996597 | 0.02 |
| 63 | 1 | 0.99994824 | 0.0500 | 0.94994824 | 0.86 |
| 65 | 1 | 0.99996813 | 0.0500 | 0.94996813 | 0.07 |
| 67 | 1 | 0.99995870 | 0.0500 | 0.94995870 | 0.11 |
| 68 | 1 | 0.99996510 | 0.0500 | 0.94996510 | 0.95 |
| 69 | 1 | 0.99996257 | 0.0500 | 0.94996257 | 0.01 |
| 72 | 1 | 0.99997178 | 0.0500 | 0.94997178 | 0.02 |
| 74 | 1 | 0.99398993 | 0.0500 | 0.94398993 | 0.74 |
| 76 | 1 | 0.99813954 | 0.0500 | 0.94813954 | 0.18 |
| 78 | 1 | 0.99997119 | 0.0500 | 0.94997119 | 0.61 |
| 80 | 1 | 0.97866954 | 0.0500 | 0.92866954 | 0.29 |
| 81 | 1 | 0.95296646 | 0.0500 | 0.90296646 | 0.21 |
| 83 | 1 | 0.99994713 | 0.0500 | 0.94994713 | 0.25 |
| 85 | 1 | 0.99992489 | 0.0500 | 0.94992489 | 0.03 |
| 87 | 0.99995238 | 0.99996281 | 0.0500 | 0.94996281 | 0.49 |
| 89 | 1 | 0.99997073 | 0.0500 | 0.94997073 | 0.04 |
| 90 | 1 | 0.99238965 | 0.0500 | 0.94238965 | 0.14 |
| **Average** | 0.99999867 | 0.99774449 | – | 0.94774449 | 0.4234 |
| **SD** | 0.00000973 | 0.00768056 | – | 0.00768056 | 0.34155473 |

Table 7.13: Duplication of Cooperative Data Fusion (Stage 1 – 40%) Statistics for Threshold Calculation

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.98 | 0.02 | 2 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0.00009524 | 1 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0.00009524 | 1 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| Average | 0.9996 | 0.0004 | 0.04 | 0.00000381 | 0.04 |
| SD | 0.002828 | 0.002828 | – | 0.00001885 | – |

Table 7.14: Duplication of Cooperative Data Fusion (Stage 1 – 40%) Results

The individual training group members' results, demonstrated in Table 7.14, indicate that 48 of the 50 training group members achieved a FAR of 0.0. That is, 48 training group members had no impostor samples (out of 10,500) accepted as their own. This was a very good result, and reflective of the AUC figures presented in Table 7.13.

The two training group members to register a non-zero FAR were 25 and 87; both achieved a FAR of 0.00009524. This meant that members 25 and 87 had 1 impostor sample each (out of 10,500) incorrectly accepted as their own.

Table 7.14 also showed that only member 23 registered a non-zero FRR of 0.02. This meant that member 23 had 2 of 100 genuine samples incorrectly rejected. Note that members 23, 25 and 87 were the only members to register AUC values less than 1. As an area of 1 reflects perfect recognition (at the nominated decision threshold), the non-zero FAR and FRR scores for these members verifies the less than perfect performance.

The last two rows of Table 7.14 provide the average and standard deviation FAR and FRR figures, for all training group members, for the cooperative data fusion phase of the experiment (at 40%). The average FAR was 0.00000381 with a standard deviation of 0.00001885, and the average FRR was 0.0004 with a standard deviation of 0.002828. These figures demonstrated that the cooperative data fusion phase of the experiment (at 40%) performed very well.

It meant that on average, there was a 4 in 10,000 chance that any of the 50 training group members would have one of their own genuine samples incorrectly rejected. Also, there was approximately a 4 in 1,000,000 chance that any of the 50 training group members would have an impostor sample incorrectly accepted as their own.

**Stage 2 (50%)**

The statistics in Table 7.15 provide information regarding the calculation of the decision threshold, for all 50 training group members, for stage 2 of the cooperative data fusion phase of the experiment.

Figures in column 2 (the Area Under the ROC Curve) of Table 7.15 show that 49 of the 50 training group members achieved an area of 1. This was an extremely positive result. The member with an area less than 1 was member 81 with an area of 0.99999714.

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99995281 | 0.0500 | 0.94995281 | 0.82 |
| 2 | 1 | 0.99993966 | 0.0500 | 0.94993966 | 0.88 |
| 3 | 1 | 0.99786012 | 0.0500 | 0.94786012 | 0.70 |
| 5 | 1 | 0.99999326 | 0.0500 | 0.94999326 | 0.67 |
| 7 | 1 | 0.99988564 | 0.0500 | 0.94988564 | 0.15 |
| 9 | 1 | 0.99997344 | 0.0500 | 0.94997344 | 0.09 |
| 12 | 1 | 0.99999695 | 0.0500 | 0.94999695 | 0.28 |
| 14 | 1 | 0.99998740 | 0.0500 | 0.94998740 | 0.96 |
| 16 | 1 | 0.99978117 | 0.0500 | 0.94978117 | 0.11 |
| 18 | 1 | 0.99990018 | 0.0500 | 0.94990018 | 0.95 |
| 20 | 1 | 0.99991915 | 0.0500 | 0.94991915 | 0.95 |
| 21 | 1 | 0.99998011 | 0.0500 | 0.94998011 | 0.24 |
| 23 | 1 | 0.99796202 | 0.0500 | 0.94796202 | 0.93 |
| 24 | 1 | 0.99897221 | 0.0500 | 0.94897221 | 0.02 |
| 25 | 1 | 0.99994968 | 0.0500 | 0.94994968 | 0.98 |
| 27 | 1 | 0.99993967 | 0.0500 | 0.94993967 | 0.68 |
| 29 | 1 | 0.99997148 | 0.0500 | 0.94997148 | 0.03 |
| 32 | 1 | 0.99991157 | 0.0500 | 0.94991157 | 0.06 |
| 34 | 1 | 0.99991700 | 0.0500 | 0.94991700 | 0.64 |
| 36 | 1 | 0.99997909 | 0.0500 | 0.94997909 | 0.47 |
| 38 | 1 | 0.99986791 | 0.0500 | 0.94986791 | 0.87 |
| 40 | 1 | 0.99998716 | 0.0500 | 0.94998716 | 0.08 |
| 41 | 1 | 0.99993588 | 0.0500 | 0.94993588 | 0.02 |
| 43 | 1 | 0.99995880 | 0.0500 | 0.94995880 | 0.06 |
| 45 | 1 | 0.99993733 | 0.0500 | 0.94993733 | 0.19 |
| 46 | 1 | 0.99997161 | 0.0500 | 0.94997161 | 0.93 |
| 47 | 1 | 0.99989028 | 0.0500 | 0.94989028 | 0.12 |
| 49 | 1 | 0.99994911 | 0.0500 | 0.94994911 | 0.88 |
| 52 | 1 | 0.99993010 | 0.0500 | 0.94993010 | 0.93 |
| 54 | 1 | 0.99997683 | 0.0500 | 0.94997683 | 0.04 |
| 56 | 1 | 0.99995208 | 0.0500 | 0.94995208 | 0.79 |
| 58 | 1 | 0.99990854 | 0.0500 | 0.94990854 | 0.85 |
| 60 | 1 | 0.99997807 | 0.0500 | 0.94997807 | 0.15 |
| 61 | 1 | 0.99996351 | 0.0500 | 0.94996351 | 0.03 |
| 63 | 1 | 0.9999384 | 0.0500 | 0.9499384 | 0.66 |
| 65 | 1 | 0.99997876 | 0.0500 | 0.94997876 | 0.14 |
| 67 | 1 | 0.99993852 | 0.0500 | 0.94993852 | 0.09 |
| 68 | 1 | 0.99996217 | 0.0500 | 0.94996216 | 0.95 |
| 69 | 1 | 0.99994770 | 0.0500 | 0.94994770 | 0.03 |
| 72 | 1 | 0.99996497 | 0.0500 | 0.94996497 | 0.02 |
| 74 | 1 | 0.99671857 | 0.0500 | 0.94671857 | 0.82 |
| 76 | 1 | 0.99903137 | 0.0500 | 0.94903136 | 0.26 |
| 78 | 1 | 0.99999157 | 0.0500 | 0.94999157 | 0.20 |
| 80 | 1 | 0.99675004 | 0.0500 | 0.94675004 | 0.05 |
| 81 | 0.99999714 | 0.99820160 | 0.0500 | 0.94820160 | 0.98 |
| 83 | 1 | 0.99995360 | 0.0500 | 0.94995360 | 0.14 |
| 85 | 1 | 0.99996115 | 0.0500 | 0.94996115 | 0.12 |
| 87 | 1 | 0.99998683 | 0.0500 | 0.94998683 | 0.92 |
| 89 | 1 | 0.99996138 | 0.0500 | 0.94996138 | 0.04 |
| 90 | 1 | 0.99462889 | 0.0500 | 0.94462889 | 0.13 |
| Average | 0.99999994 | 0.99955791 | – | 0.94955791 | 0.442 |
| SD | 0.00000040 | 0.00105515 | – | 0.00105515 | 38194988 |

Table 7.15: Duplication of Cooperative Data Fusion (Stage 2 – 50%) Statistics for Threshold Calculation

Other points of interest in Table 7.15 are:

- The confidence level for all 50 training group members was 0.5, which indicated that no participant required an adjustment to their confidence level because of poor classifier performance.

- Participants 25 and 81 had the highest decision threshold value of 0.98.

- The decision thresholds for the cooperative data fusion phase of the experiment (at 50%) were generally similar to those for the complementary data fusion phase. The average for the decision thresholds was 0.442 with a standard deviation of 0.38194988; the average, and the standard deviation, were slightly higher than those recorded for the complementary data fusion phase.

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 1 | 0 | 0 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 0.97 | 0.03 | 3 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| Average | 0.9994 | 0.0006 | 0.06 | 0 | 0 |
| SD | 0.004243 | 0.004243 | – | 0 | – |

Table 7.16: Duplication of Cooperative Data Fusion (Stage 2 – 50%) Results

The individual training group members' results, demonstrated in Table 7.16 indicate that all 50 training group members achieved a FAR of 0.0. This meant that no training group member had any impostor samples (out of 10,500) accepted as their own. Again, this was an excellent result and reflective of the AUC figures presented in Table 7.15.

Table 7.16 also showed that only training group member 81 registered a non-zero FRR of 0.03. This meant that member 81 had 3 out of 100 genuine samples incorrectly rejected. Note that member 81 was the only member to register an AUC value less than 1. As an area of 1 reflects perfect recognition at the nominated decision threshold, the non-zero FRR for member 81 verifies this less than perfect performance.

The last two rows of Table 7.16 provide the average and standard deviation FAR and FRR figures, for all training group members, for the cooperative data fusion phase of the experiment (at 50%). The average FAR was 0.0 with a standard deviation of 0.0, and the average FRR was 0.0006 with a standard deviation of 0.004243. These figures demonstrated that the cooperative data fusion phase of the experiment (at 50%) performed extremely well.

It meant that on average, there was a 6 in 10,000 chance that any of the 50 training group members would have one of their own genuine samples incorrectly rejected. Also, there was no chance that any training group member would have an impostor sample (out of 10,500) incorrectly accepted as their own.

### Stage 3 (60%)

The statistics in Table 7.17 provide information regarding the calculation of the decision threshold, for all 50 training group members, for stage 3 of the cooperative data fusion phase of the experiment.

Figures in column 2 (the Area Under the ROC Curve) of Table 7.17 show that 49 of the 50 training group members achieved an area of 1. This is an extremely positive result. The member with an area less than 1 was member 23 with an AUC of 0.99998571.

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99992793 | 0.0500 | 0.94992793 | 0.85 |
| 2 | 1 | 0.99996556 | 0.0500 | 0.94996556 | 0.89 |
| 3 | 1 | 0.99843240 | 0.0500 | 0.94843240 | 0.38 |
| 5 | 1 | 0.98928223 | 0.0500 | 0.93928223 | 0.03 |
| 7 | 1 | 0.99998868 | 0.0500 | 0.94998868 | 0.16 |
| 9 | 1 | 0.99996960 | 0.0500 | 0.94996960 | 0.10 |
| 12 | 1 | 0.99998107 | 0.0500 | 0.94998107 | 0.18 |
| 14 | 1 | 0.99982194 | 0.0500 | 0.94982194 | 0.69 |
| 16 | 1 | 0.99985554 | 0.0500 | 0.94985554 | 0.58 |
| 18 | 1 | 0.99995425 | 0.0500 | 0.94995425 | 0.93 |
| 20 | 1 | 0.99992648 | 0.0500 | 0.94992648 | 0.94 |
| 21 | 1 | 0.99993923 | 0.0500 | 0.94993923 | 0.20 |
| 23 | 0.99998571 | 0.99222820 | 0.0500 | 0.94222819 | 0.95 |
| 24 | 1 | 0.99828443 | 0.0500 | 0.94828443 | 0.25 |
| 25 | 1 | 0.99990365 | 0.0500 | 0.94990365 | 0.92 |
| 27 | 1 | 0.99993836 | 0.0500 | 0.94993836 | 0.81 |
| 29 | 1 | 0.99996302 | 0.0500 | 0.94996302 | 0.03 |
| 32 | 1 | 0.99990327 | 0.0500 | 0.94990327 | 0.03 |
| 34 | 1 | 0.99985921 | 0.0500 | 0.94985921 | 0.73 |
| 36 | 1 | 0.99995089 | 0.0500 | 0.94995089 | 0.64 |
| 38 | 1 | 0.99764964 | 0.0500 | 0.94764963 | 0.13 |
| 40 | 1 | 0.99994105 | 0.0500 | 0.94994104 | 0.20 |
| 41 | 1 | 0.99992359 | 0.0500 | 0.94992359 | 0.03 |
| 43 | 1 | 0.99995974 | 0.0500 | 0.94995974 | 0.05 |
| 45 | 1 | 0.99993968 | 0.0500 | 0.94993968 | 0.04 |
| 46 | 1 | 0.99997368 | 0.0500 | 0.94997368 | 0.90 |
| 47 | 1 | 0.99993063 | 0.0500 | 0.94993063 | 0.15 |
| 49 | 1 | 0.99995690 | 0.0500 | 0.94995690 | 0.90 |
| 52 | 1 | 0.99995330 | 0.0500 | 0.94995330 | 0.92 |
| 54 | 1 | 0.99995585 | 0.0500 | 0.94995585 | 0.07 |
| 56 | 1 | 0.99995054 | 0.0500 | 0.94995054 | 0.80 |
| 58 | 1 | 0.99994746 | 0.0500 | 0.94994746 | 0.95 |
| 60 | 1 | 0.99998110 | 0.0500 | 0.94998110 | 0.15 |
| 61 | 1 | 0.99994356 | 0.0500 | 0.94994356 | 0.02 |
| 63 | 1 | 0.99995022 | 0.0500 | 0.94995022 | 0.90 |
| 65 | 1 | 0.99995826 | 0.0500 | 0.94995826 | 0.12 |
| 67 | 1 | 0.99997349 | 0.0500 | 0.94997349 | 0.22 |
| 68 | 1 | 0.99994444 | 0.0500 | 0.94994444 | 0.95 |
| 69 | 1 | 0.99998234 | 0.0500 | 0.94998234 | 0.01 |
| 72 | 1 | 0.99996257 | 0.0500 | 0.94996257 | 0.02 |
| 74 | 1 | 0.99440462 | 0.0500 | 0.94440462 | 0.57 |
| 76 | 1 | 0.99914280 | 0.0500 | 0.94914280 | 0.64 |
| 78 | 1 | 0.99997657 | 0.0500 | 0.94997657 | 0.73 |
| 80 | 1 | 0.99899996 | 0.0500 | 0.94899996 | 0.03 |
| 81 | 1 | 0.98798671 | 0.0500 | 0.93798671 | 0.77 |
| 83 | 1 | 0.99985162 | 0.0500 | 0.94985162 | 0.59 |
| 85 | 1 | 0.99998320 | 0.0500 | 0.94998320 | 0.25 |
| 87 | 1 | 0.99995217 | 0.0500 | 0.94995217 | 0.80 |
| 89 | 1 | 0.99995057 | 0.0500 | 0.94995057 | 0.04 |
| 90 | 1 | 0.9947447 | 0.0500 | 0.94474474 | 0.09 |
| Average | 0.99999971 | 0.99897694 | – | 0.94897694 | 0.4466 |
| SD | 0.00000202 | 0.00263041 | – | 0.00263041 | 0.36663172 |

Table 7.17: Duplication of Cooperative Data Fusion (Stage 3 – 60%) Statistics for Threshold Calculation

Other points of interest in Table 7.17 are:

- The confidence level for all 50 training group members was 0.5, which indicated that no participant required an adjustment to their confidence level because of poor classifier performance.

- Participant 23 had the highest decision threshold value of 0.95 (shared with participants 58 and 68).

- The decision thresholds for the cooperative data fusion phase of the experiment (at 60%) were generally similar to those for the complementary data fusion phase. The average for the decision thresholds was 0.4466 with a standard deviation of 0.36663172; the average, and the standard deviation, was slightly higher than those recorded for the complementary data fusion phase.

The individual training group members' results, demonstrated in Table 7.18 indicate that all 50 training group members achieved a FAR of 0.0. This meant that all training group members had no impostor samples (out of 10,500) accepted as their own. Again, this is an excellent result and reflective of the AUC figures presented in Table 7.17.

Table 7.18 also showed that only one training group member registered a non-zero FRR; member 23 with a rate of 0.05. This meant that member 23 had 5 out of 100 genuine samples incorrectly rejected. Note that member 23 was the only member to register an AUC value less than 1. As an area of 1 reflects perfect recognition at the nominated decision threshold, the non-zero FRR for member 23 verifies this less than perfect performance.

The last two rows of Table 7.18 provide the average and standard deviation FAR and FRR figures, for all training group members, for the cooperative data fusion phase of the experiment (at 60%). The average FAR was 0.0 with a standard deviation of 0.0, and the average FRR was 0.001 with a standard deviation of 0.007071. These figures demonstrated that the cooperative data fusion phase of the experiment (at 60%) performed extremely well.

It meant that on average, there was a 1 in 1,000 chance that any of the 50 training group members would have one of their own genuine samples incorrectly rejected. Also, there was no chance that any training group member would have an impostor sample (out of 10,500) incorrectly accepted as their own.

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.95 | 0.05 | 5 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| Average | 0.9990 | 0.0010 | 0.1 | 0 | 0 |
| SD | 0.007071 | 0.007071 | – | 0 | – |

Table 7.18: Duplication of Cooperative Data Fusion (Stage 3 – 60%) Results

### Stage 4 (70%)

The statistics in Table 7.19 provide information regarding the calculation of the decision threshold, for all 50 training group members, for stage 4 of the cooperative data fusion phase of the experiment.

| Participant | Area Under ROC Curve | True Positive Mean | Confidence Level | Adjustable Determinant Value | Decision Threshold |
|---|---|---|---|---|---|
| 1 | 1 | 0.99991163 | 0.0500 | 0.94991163 | 0.67 |
| 2 | 1 | 0.99995292 | 0.0500 | 0.94995292 | 0.93 |
| 3 | 1 | 0.99709036 | 0.0500 | 0.94709036 | 0.41 |
| 5 | 1 | 0.98615842 | 0.0500 | 0.93615841 | 0.10 |
| 7 | 1 | 0.99997614 | 0.0500 | 0.94997614 | 0.44 |
| 9 | 1 | 0.99997112 | 0.0500 | 0.94997112 | 0.25 |
| 12 | 1 | 0.99995742 | 0.0500 | 0.94995742 | 0.24 |
| 14 | 1 | 0.99979208 | 0.0500 | 0.94979208 | 0.51 |
| 16 | 1 | 0.99988488 | 0.0500 | 0.94988488 | 0.35 |
| 18 | 1 | 0.99996285 | 0.0500 | 0.94996285 | 0.94 |
| 20 | 1 | 0.99995091 | 0.0500 | 0.94995091 | 0.95 |
| 21 | 1 | 0.99996662 | 0.0500 | 0.94996662 | 0.47 |
| 23 | 0.99998238 | 0.99344748 | 0.0500 | 0.94344748 | 0.95 |
| 24 | 1 | 0.99873973 | 0.0500 | 0.94873973 | 0.71 |
| 25 | 1 | 0.99987783 | 0.0500 | 0.94987783 | 0.95 |
| 27 | 1 | 0.99989272 | 0.0500 | 0.94989272 | 0.76 |
| 29 | 1 | 0.99997935 | 0.0500 | 0.94997935 | 0.03 |
| 32 | 1 | 0.99995041 | 0.0500 | 0.94995041 | 0.03 |
| 34 | 1 | 0.99988808 | 0.0500 | 0.94988808 | 0.68 |
| 36 | 1 | 0.99998034 | 0.0500 | 0.94998034 | 0.70 |
| 38 | 1 | 0.99767674 | 0.0500 | 0.94767674 | 0.14 |
| 40 | 1 | 0.99997277 | 0.0500 | 0.94997277 | 0.25 |
| 41 | 1 | 0.99994357 | 0.0500 | 0.94994357 | 0.19 |
| 43 | 1 | 0.99992265 | 0.0500 | 0.94992265 | 0.02 |
| 45 | 1 | 0.99992186 | 0.0500 | 0.94992186 | 0.03 |
| 46 | 1 | 0.99994753 | 0.0500 | 0.94994753 | 0.84 |
| 47 | 1 | 0.99987963 | 0.0500 | 0.94987963 | 0.14 |
| 49 | 1 | 0.99989966 | 0.0500 | 0.94989966 | 0.95 |
| 52 | 1 | 0.99990184 | 0.0500 | 0.94990184 | 0.99 |
| 54 | 1 | 0.99998099 | 0.0500 | 0.94998099 | 0.15 |
| 56 | 1 | 0.99989832 | 0.0500 | 0.94989831 | 0.52 |
| 58 | 1 | 0.99987557 | 0.0500 | 0.94987557 | 0.95 |
| 60 | 1 | 0.99997128 | 0.0500 | 0.94997128 | 0.17 |
| 61 | 1 | 0.99981937 | 0.0500 | 0.94981937 | 0.02 |
| 63 | 1 | 0.99997335 | 0.0500 | 0.94997335 | 0.91 |
| 65 | 1 | 0.99994637 | 0.0500 | 0.94994637 | 0.09 |
| 67 | 1 | 0.99993493 | 0.0500 | 0.94993493 | 0.20 |
| 68 | 1 | 0.99996165 | 0.0500 | 0.94996165 | 0.90 |
| 69 | 1 | 0.99992784 | 0.0500 | 0.94992784 | 0.04 |
| 72 | 1 | 0.99993243 | 0.0500 | 0.94993243 | 0.03 |
| 74 | 1 | 0.99493857 | 0.0500 | 0.94493857 | 0.82 |
| 76 | 1 | 0.99817754 | 0.0500 | 0.94817754 | 0.17 |
| 78 | 1 | 0.99995635 | 0.0500 | 0.94995635 | 0.64 |
| 80 | 1 | 0.99315162 | 0.0500 | 0.94315162 | 0.14 |
| 81 | 1 | 0.95857320 | 0.0500 | 0.90857320 | 0.47 |
| 83 | 1 | 0.99985074 | 0.0500 | 0.94985074 | 0.50 |
| 85 | 1 | 0.99995945 | 0.0500 | 0.94995945 | 0.01 |
| 87 | 1 | 0.99992721 | 0.0500 | 0.94992721 | 0.92 |
| 89 | 1 | 0.99996624 | 0.0500 | 0.94996624 | 0.03 |
| 90 | 1 | 0.99305809 | 0.0500 | 0.94305809 | 0.03 |
| Average | 0.99999965 | 0.99816357 | – | 0.94816357 | 0.4466 |
| SD | 0.00000249 | 0.00626269 | – | 0.00626269 | 0.35438714 |

Table 7.19: Duplication of Cooperative Data Fusion (Stage 4 – 70%) Statistics for Threshold Calculation

Figures in column 2 (the Area Under the ROC Curve) of Table 7.19 show that 49 of the 50 training group members achieved an area of 1. This is an extremely positive result. The member with an area less than 1 was member 23 with an AUC of 0.99998238.

Other points of interest in Table 7.17 are:

- The confidence level for all 50 training group members was 0.5, which indicated that no participant required an adjustment to their confidence level because of poor classifier performance.

- Participant 23 had the second highest decision threshold value of 0.95 (shared with participants 20, 25, 49, 58).

- The decision thresholds for the cooperative data fusion phase of the experiment (at 70%) were generally similar to those for the complementary data fusion phase. The average for the decision thresholds was 0.4466 with a standard deviation of 0.35438714; the average was slightly higher, and the standard deviation slightly lower, than those recorded for the complementary data fusion phase.

The individual training group members' results, demonstrated in Table 7.20 indicate that all 50 training group members achieved a FAR of 0.0. This meant that all training group members had no impostor samples (out of 10,500) accepted as their own. Again, this is an excellent result and reflective of the AUC figures presented in Table 7.19.

Table 7.20 also showed that only one training group member registered a non-zero FRR; member 23 with a rate of 0.05. This meant that member 23 had 5 out of 100 samples genuine incorrectly rejected. Note that member 23 was the only member to register an AUC value less than 1. As an area of 1 reflects perfect recognition at the nominated decision threshold, the non-zero FRR for member 23 verifies this less than perfect performance.

The last two rows of Table 7.20 provide the average and standard deviation FAR and FRR figures, for all participants, for the cooperative data fusion phase of the experiment (at 70%). The average FAR was 0.0 with a standard deviation of 0.0, and the average FRR was 0.001 with a standard deviation of 0.007071. These figures demonstrated that the cooperative data fusion phase of the experiment (at 70%) performed extremely well.

| Participant | True Positive Rate | False Rejection Rate | Number of False Rejections | False Acceptance Rate | Number of False Acceptances |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 12 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 |
| 16 | 1 | 0 | 0 | 0 | 0 |
| 18 | 1 | 0 | 0 | 0 | 0 |
| 20 | 1 | 0 | 0 | 0 | 0 |
| 21 | 1 | 0 | 0 | 0 | 0 |
| 23 | 0.95 | 0.05 | 5 | 0 | 0 |
| 24 | 1 | 0 | 0 | 0 | 0 |
| 25 | 1 | 0 | 0 | 0 | 0 |
| 27 | 1 | 0 | 0 | 0 | 0 |
| 29 | 1 | 0 | 0 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 |
| 34 | 1 | 0 | 0 | 0 | 0 |
| 36 | 1 | 0 | 0 | 0 | 0 |
| 38 | 1 | 0 | 0 | 0 | 0 |
| 40 | 1 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 0 | 0 | 0 |
| 43 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 0 | 0 | 0 |
| 46 | 1 | 0 | 0 | 0 | 0 |
| 47 | 1 | 0 | 0 | 0 | 0 |
| 49 | 1 | 0 | 0 | 0 | 0 |
| 52 | 1 | 0 | 0 | 0 | 0 |
| 54 | 1 | 0 | 0 | 0 | 0 |
| 56 | 1 | 0 | 0 | 0 | 0 |
| 58 | 1 | 0 | 0 | 0 | 0 |
| 60 | 1 | 0 | 0 | 0 | 0 |
| 61 | 1 | 0 | 0 | 0 | 0 |
| 63 | 1 | 0 | 0 | 0 | 0 |
| 65 | 1 | 0 | 0 | 0 | 0 |
| 67 | 1 | 0 | 0 | 0 | 0 |
| 68 | 1 | 0 | 0 | 0 | 0 |
| 69 | 1 | 0 | 0 | 0 | 0 |
| 72 | 1 | 0 | 0 | 0 | 0 |
| 74 | 1 | 0 | 0 | 0 | 0 |
| 76 | 1 | 0 | 0 | 0 | 0 |
| 78 | 1 | 0 | 0 | 0 | 0 |
| 80 | 1 | 0 | 0 | 0 | 0 |
| 81 | 1 | 0 | 0 | 0 | 0 |
| 83 | 1 | 0 | 0 | 0 | 0 |
| 85 | 1 | 0 | 0 | 0 | 0 |
| 87 | 1 | 0 | 0 | 0 | 0 |
| 89 | 1 | 0 | 0 | 0 | 0 |
| 90 | 1 | 0 | 0 | 0 | 0 |
| Average | 0.9990 | 0.0010 | 0.1 | 0 | 0 |
| SD | 0.007071 | 0.007071 | – | 0 | – |

Table 7.20: Duplication of Cooperative Data Fusion (Stage 4 – 70%) Results

It meant that on average, there was a 1 in 1,000 chance that any of the 50 training group members would have one of their own genuine samples incorrectly rejected. Also, there was no chance that any training group member would have an impostor sample (out of 10,500) incorrectly accepted as their own.

As demonstrated in Tables 7.14, 7.16, 7.18, and 7.20, the cooperative data fusion results for all stages performed extremely well. The only stage to register a non-zero average FAR was stage 1, where 40% of the available data was fused. For that stage, an average FAR of 0.0004 resulted. In attaining that average FAR, only 2 training group members had 1 impostor sample (out of 10,500) incorrectly accepted as their own; all other members achieved a FAR of 0.0. The other three stages for this phase of the experiment achieved a average FAR of 0.0.

Stage 2 performed the best of the 4 stages, with an average FAR of 0.0 and an average FRR of 0.0006. In attaining that average FRR, only 1 training group member had 3 of 100 genuine samples incorrectly rejected; all other members achieved a FRR of 0.0. Stages 3 and 4 also performed very well, with an average FAR of 0.0 and an average FRR of 0.001. In both stages, only 1 training group member had 5 of 100 genuine samples incorrectly rejected.

**Comparison with Reviewed Cooperative Data Fusion Research**

The average FAR and average FRR—for the 50 training group members—provide figures that permit a comparison with the results of the research efforts reviewed in Chapter 2 section 2.3.2.2, that utilised a cooperative data fusion approach. Though the research efforts reviewed conducted experiments that varied in certain aspects of data fusion (for example: the number of modalities and the fusion method), it is still feasible to compare their results with those of the current study, as in most cases the same performance variables (that is, FAR and the FRR) were used[21].

Table 7.21 provides a summary of the results achieved by research efforts that utilised a cooperative data fusion approach. The information in Table 7.21 is duplicated from Table 2.2 in Chapter 2 section 2.3.2.2, and is provided for convenient comparison between the results of the reviewed research and the results of the current study[22].

[21] As noted in Chapter 2 section 2.3.2.2, numerous authors expressed the performance variables as a percentage. That is, the percentage of the FAR and the percentage of the FRR. However in Table 2.2 columns 7 and 8 (Chapter 2 section 2.3.2.2), the figures for the performance variables are denoted as their actual rates (i.e. the percentage rate divided by 100). Any discussion comparing the performance variables will denote both the actual rate and the percentage rate (in parentheses).

[22] Note that a direct comparison between the experimental results of the publications listed in Table 7.21 (and also between them and the results of the current experiment) would be misleading. All of these results can only be viewed as an informal indicator of algorithmic performance because different data sets were used in the experiments.

| Reviewed Paper | Fusion Level | FAR | FRR |
|---|---|---|---|
| Son and Lee, 2005 | Feature | 0.0 | 0.0 |
| Ross and Govindarajan, 2005 | Feature | 0.0001 | 0.13 |
| Rattani et al., 2007 | Feature | 0.0102 | 0.0195 |
| Yao et al., 2007 | Feature | na | na |
| **Current Study, 2010 Stage 1** | **Feature** | **0.00000381** | **0.0004** |
| **Stage 2** | **Feature** | **0.0** | **0.0006** |
| **Stage 3** | **Feature** | **0.0** | **0.001** |
| **Stage 4** | **Feature** | **0.0** | **0.001** |

Table 7.21: Summary of Reviewed Papers Using The Cooperative Data Fusion Paradigm

Like the current experiment, Son and Lee (2005) used two modalities. For their experiment, the authors used facial and iris recognition fused at the feature level. The Son and Lee (2005) experiment utilised 10 facial image samples each from 140 individuals and 10 iris scan image samples each from 200 individuals (eventuating in 1,000 high quality images and 1,000 low quality images). In the current experiment, 140 typing samples and 140 fingerprint samples from 90 participants were used.

In Chapter 2 section 2.3.2.2 it was explained that the Son and Lee (2005) adopted a statistical approach to reduce the dimensionality of the combined feature vector. The specific technique employed was Direct Linear Discriminant Analysis (DLDA), which utilises Principle Component Analysis (PCA) to help reduce dimensionality. These techniques not only reduced dimensionality, but in the process performed feature selection. The result was a 'Reduced Joint Feature Vector' (RJFV). The authors provided no information in regard to the proportion of features from each of the modalities that constituted the combined feature vector.

The above method of feature selection differed from that performed in the current study. For the current experiment, the weights from trained ANNs (a by-product of the complementary phase of the experiment) were available, and a method that effectively utilised these weights (to select relevant features) was sought and implemented (described in Chapter 5 section 5.6.3.1).

The best results achieved in the Son and Lee (2005) experiment were a FAR of 0.0 and FRR of 0.0. These are outstanding results (with perfect recognition for both performance variables), and occurred when high quality iris scan image features

were combined with the facial image features. The best results in the current study (employing cooperative data fusion) were achieved in Stage 2 with a FAR of 0.0 and a FRR of 0.0006. These results are comparable to the Son and Lee (2005) results.

Like the current study and the Son and Lee (2005) experiment, Ross and Govindarajan (2005) also experimented with two modalities. For their experiment, the authors used facial images and hand geometry fused at the feature level. The Ross and Govindarajan (2005) experiment used 5 facial image samples and 5 hand geometry samples collected from 100 participants. For the current experiment, 140 typing samples and 140 fingerprint samples from 90 participants were used.

Features were extracted (using Principle Component Analysis and Linear Discriminant Analysis) from a facial image to return a vector of 27 elements. Hand geometry features consisted of physical measurements. Normalisation (or data alignment) was conducted so that their respective components could equally contribute to the matching process.

As feature selection was evident (to reduce the dimensionality of combined feature vectors), the paradigm can be designated as a cooperative data fusion approach. The sequential forward floating selection technique was adopted. This method of feature selection differed from the method in the current study, which used the weights from trained ANNs (available from the complementary phase of the experiment) for this purpose.

Also, the fusion method used by Ross and Govindarajan (2005) differed from that used in the current study. Ross and Govindarajan (2005) used various distance measures applied to the available feature data, to calculate confidence or match scores; the match scores were then combined. Final classification was based on the combined match score.

Even though the match scores were determined (at various stages of calculation) according to combined feature vectors, the methodology has a strong resemblance to confidence level fusion. As stated in Chapter 2 section 2.3.2.2, data fusion appears to have been performed as a combination of feature level and confidence score level fusion.

The best results achieved in the Ross and Govindarajan (2005) experiment were a FAR of approximately 0.0001 (0.01%) and a FRR of approximately 0.13 (13%). The FAR result was very good, however the FRR was not as good. The results of the current study were more accurate for both performance variables, and had more samples for performing genuine and impostor tests.

Rattani et al., (2007) also experimented with two modalities: facial recognition and fingerprint recognition. During feature extraction, a descriptor was determined for each feature from both modalities. Thus each feature was represented by spatial coordinates, orientation, and a descriptor.

Fusion was conducted at the feature level, by concatenating feature vectors from the two modalities. This is the simplest method of feature level fusion, but has the side effect of culminating in a large (possibly very large) combined feature vector. Thus three different strategies were used to reduce the dimensionality of the combined feature vectors. These were k-means clustering, neighbourhood elimination, and the authors' strategy '3 points in a specific region'. The aim of all three strategies was to reduce vector size by removing redundant data. As feature selection was used, a cooperative data fusion approach was evident.

The methodology of concatenating feature vectors and then selecting the more meaningful features was similar to the strategy used in the current study. The only real difference in the current study was that the features were selected before concatenation rather than after; this was a very minor difference and would be unlikely to have impacted on any findings (though this is not known for certain).

Matching involved the comparison of a fused query feature set with a fused template feature set (both constructed in the same manner). Two different classification approaches were used: point pattern matching and Delauney triangulation.

For the point pattern matching, a matching probability was calculated as the ratio of the number of correspondences to the total number of feature points in both feature sets.

For the Delauney triangulation approach, the feature vectors (comprising information derived from the delauney triangulation) were compared and a matching

score was determined based on the number of corresponding triangles that both feature sets had in common.

For the experiment, 5 facial images and 5 fingerprint samples were collected, from 50 individuals. Only 1 of the 5 samples per participant—for a facial image/fingerprint pair—was used to compute a reference template. The 4 remaining sample pairs per participant were used for testing purposes. This involved comparing a participants 4 query samples against the database of templates. Therefore, there were 200 (50x4) genuine test scores and 2,450 (50x49) impostor test scores generated. For the current experiment, the 140 typing samples and 140 fingerprint samples (from 90 participants) generated 100 genuine test scores per participant (compared with 200 for all 50 participants). Also, 10,500 impostor test scores were generated per participant (compared with 2,450 for all 50 participants).

The Rattani et al., (2007) experiment achieved best results when the '3 points in a specific region' feature reduction strategy and the Delauney triangulation classifier were used in combination. For this methodology, a FAR of 0.0102 (1.02%) and a FRR of 0.0195 (1.95%) resulted[23]. The results of the current study were more accurate for both performance variables, and had many more samples for performing genuine and impostor tests.

As the experiment conducted by Yao et al., (2007) reported results as a recognition rate of 90.73% (rather than the FAR and FRR), a direct comparison cannot be made between their results and the results of either the current study or the other reviewed research efforts. The methods adopted by Yao et al., (2007) were novel and it would have been advantageous to compare results using the usual performance variables.

#### 7.2.3.4 Summary of Cooperative Data Fusion Results

The results of the current experiment for the four stages of the cooperative data fusion phase (listed below), were nearly as accurate as the Son and Lee (2005) experiment, and more accurate than the Ross and Govindarajan (2005) and Rattani et al., (2007) experiments:

[23]Note that the authors also demonstrated that feature level fusion out-performed confidence score level fusion in all tests.

1. Stage 1 (40% of available data): a FAR of 0.00000381 and a FRR of 0.0004.

2. Stage 2 (50% of available data): a FAR of 0.0 and a FRR of 0.0006.

3. Stage 3 (60% of available data): a FAR of 0.0 and a FRR of 0.001.

4. Stage 4 (70% of available data): a FAR of 0.0 and a FRR of 0.001.

Fusion for stages 2 to 4 (where perfect performance was achieved for the FAR variable), performed better than for stage 1. The FRR rates for these stages would be considered more than acceptable for most applications. For stage 2, only one of the fifty training group members incurred 3 false rejections; all other members achieved perfect recognition for both performance variables. For stages 3 and 4, only one of the fifty training group members incurred 5 false rejections; all other members achieved perfect recognition for both performance variables.

For stage 1, only 40% of the combined feature vector was used, and the results suggest that this percentage of data was insufficient to produce the accuracy achieved in the other three stages. Logically, this make sense as it could be expected that some important features may be lost with only $2/5^{th}$ of data utilised. Even so, the results for stage 1 could still be acceptable in some applications. Only two of the fifty training group members incurred 1 false acceptance, and only one of the fifty training group members incurred 2 false rejections; all other members achieved perfect performance for both variables.

It should be noted that the experiments by Son and Lee (2005) and Ross and Govindarajan (2005) made no mention of the proportion of features (from both sources) that constituted the combined feature vectors.

The current experiment investigated fusion at the four different percentages (40%, 50%, 60%, and 70%) of available data. The proportions that constitute the combined vectors (of keystroke dynamics and fingerprint features) were different for each participant for each stage of this experimental phase. The exact proportions were determined by the feature selection process (refer Chapter 5 section 5.6.3.1).

The cooperative data fusion results achieved in the current study (particularly for stages 2 to 4) demonstrated a very high degree of accuracy, and were better (or at

least comparable to) other research efforts in this field. The cooperative data fusion results (particularly stage 2) were only marginally less accurate than the results for the complementary data fusion phase (which performed best overall, with a FAR of 0.0 and a FRR of 0.0004).

The next section provides a conclusion to this chapter, by summarising the results and some of the main differences between the experiments reviewed.

## 7.3 Conclusion

This chapter has discussed in detail the results obtained for the keystroke dynamics phase of the experiment (section 7.2.1), the fingerprint recognition phase of the experiment (section 7.2.2), and the data fusion phase of the experiment (section 7.2.3. In each section, the experimental results of the current study presented in Chapter 6 section 6.4 were discussed in detail, before comparing these results with those of the reviewed research efforts discussed in Chapter 3 section 3.4.1, Chapter 4 section 4.5, and Chapter 2 section 2.3.2.2 respectively.

Results demonstrated that the data fusion phase of the experiment performed better than the other two phases. Of the data fusion approaches, the complementary paradigm performed best with a FAR of 0.0 and a FRR of 0.0004. The cooperative paradigm also performed very well, with stage 2 (fusion of 50% of available data) performing the best with a FAR of 0.0 and a FRR of 0.0006. Stages 3 and 4 (fusion of 60% and 70% of available data) also performed very well with FARs of 0.0 and FRRs of 0.001. All were excellent results, and performed better than stage 1 (fusion of 40% of available data) with a FAR of 0.004 and a FRR of 0.004.

The fingerprint recognition phase of the experiment, whilst not achieving a FRR as low as the data fusion phase (with the exception of stage 1 of the cooperative data fusion phase), did achieve a FAR of 0.0 and FRR of 0.0022. With a FAR of 0.0, these are still excellent results even though the FRR was marginally higher.

The keystroke dynamics phase of the experiment performed worse than the other two phases, with a FAR of 0.02766095 and a FRR of 0.0862. Given the known variability of keystroke dynamics data, this was to be expected. However, the re-

sults demonstrated that feature selection—to remove noisy/redundant data—does improve accuracy for this biometric characteristic. Also, the data fusion results suggest that even with the variability of keystroke dynamics data, good results are attainable when data from this biometric characteristic are fused with data from another (less variable) source.

As shown in the relative sections of this chapter, the results achieved in this study were generally better than (or at least comparable to) other research efforts in the associated fields. The keystroke dynamics results did not achieve accuracy comparable to the other two phases of the experiment, but did achieve accuracy comparable to other research in the field (excluding those that used ANNs for classification). The fingerprint recognition and data fusion results also achieved accuracy comparable to other research in those fields.

In relation to the fingerprint recognition phase of the experiment, it should remembered that the treatment applied to the data was very different from other research efforts and was specifically designed to facilitate feature level data fusion. The fact that the results achieved for this phase were comparable with other research, suggests that this methodology could be useful even if fingerprint recognition was to be used in a uni-modal application.

The excellent results achieved by the data fusion phase of the experiment demonstrated that combining two or more sources of data does improve accuracy in the authentication procedure. Whilst the results of the complementary paradigm were better than the results for the cooperative paradigm, they were only marginally so.

The premise for the use of more than one biometric characteristic in the authentication procedure, is that it will make the system more accurate and robust. Though accuracy and robustness are closely related, a subtle difference exists. Clearly, accuracy can be measured, and has been established in both the current study and the other research efforts reviewed. Robustness is not as clearly obvious or measurable, and may be thought of as a hidden property; it is more concerned with making impersonation of a legitimate user more difficult for an impostor, rather than measuring the rate at which impostor samples are incorrectly accepted.

No attempt has been made to evaluate the robustness of the biometric authentication process presented in this study, because the author is not aware of any methods to do so. However, considered belief among researchers in this field suggests that combining more than one data source will make a system more robust, because an impostor would need to impersonate multiple biometric characteristics. In comparison to impersonating just one biometric characteristic, this logically becomes a more difficult task.

An advantage of cooperative data fusion—over complementary data fusion—is that the feature selection process involved means that it would be more difficult to forge the biometric signature, than if no selection was conducted. Some may consider this to be "security through obscurity" (where the security level cannot be measured or known). However, the feature selection process does not attempt to obscure anything; it is transparent. The particular features selected are dependent on the nature of the data rather than any obscurity due to the selection process.

A disadvantage of cooperative data fusion—compared to complementary data fusion—is that the feature selection process adds extra complexity (and computational cost) to the task at hand. In the current study for example, the weights from trained ANNs (from the complementary data fusion phase) were required for the feature selection needed for the cooperative data fusion phase. This requirement could render this particular methodology impractical for general use (because of the extra computational cost), but may be acceptable for applications requiring stricter security.

Given the results of the current study (with the complementary data fusion phase achieving marginally better accuracy than the cooperative data fusion phase) it may be questionable whether there is any real advantage to the cooperative paradigm, other than the perceived robustness resulting from the feature selection. Until methods are available for testing the difficulty of impersonating a legitimate user (rather than relying solely on accuracy measures), this question remains a moot point.

The next chapter provides a conclusion to this dissertation.

# Chapter 8

# Conclusion

This chapter concludes the current dissertation by summarising:

- The purpose and objectives of the research (section 8.1);
- The main contributions of the research to the field of biometric authentication (section 8.2);
- The limitations of the research (section 8.3);
- The implications and practical applications of the research (section 8.4); and
- The future research that this dissertation stimulates (section 8.5).

## 8.1 Research Purpose and Objectives

In Chapter 1 section 1.2, the traditional authentication model was noted as being the most utilised method for authentication procedures. This model is typically token based, with a token consisting of either a key, a smart-card, or a password. Password-based authentication systems are responsible for a large proportion of computer network security breaches. Passwords may be compromised by loss, theft, guessing or cracking. On the positive side, passwords are easily revocable. That is, upon discovery of a compromised password, the token can simply be changed.

In recent years, biometric authentication systems have been increasingly investigated as a prospective alternative to password-based authentication systems.

In Chapter 2 section 2.2, biometrics were introduced and considerations in relation to the practical implementation of the various biometric characteristics were discussed.

The concept of utilising more than one biometric characteristic (for authentication purposes) was also discussed in Chapter 2 sections 2.2 and 2.3. Although few authentication systems implement multiple biometrics in real world applications, research has demonstrated that this approach has numerous benefits (refer Chapter 2 section 2.3.2.2). However, there still remain practical concerns for biometric authentication systems (in general), if they are to be implemented securely.

The purpose of this research was to provide a means for improving the initial authentication procedure by evaluating a multi-modal biometric authentication system, where data from two sources were fused at the feature level. A methodology to implement and test such a system was developed (refer Chapter 5). The research was undertaken on the premise that if two sources of data could be successfully fused, then it was highly likely that more than two sources could be successfully fused using a similar methodology.

There were three main objectives for the research, which became the three phases of the experiment:

1. To investigate the effective use of the biometric characteristic keystroke dynamics (refer Chapter 5 section 5.4), and to assess the results in comparison to previous research in this field (refer Chapter 7 section 7.2.1).

2. To investigate the effective use of the biometric characteristic fingerprint recognition, such that later feature level data fusion was facilitated (refer Chapter 5 section 5.5), and to assess the results in comparison to previous research in this field (refer Chapter 7 section 7.2.2).

3. To investigate the feature level fusion of data from the previously stated two sources (refer Chapter 5 section 5.6), and to assess the results in comparison to previous research in this field (refer Chapter 7 section 7.2.3).

It should be noted that the approaches adopted for phases 1 and 2 of these experiments may not be considered the optimum approaches. However, as the key objective was considered to be that related to phase 3, the methodology for phases 1 and 2 were adopted to facilitate phase 3.

In relation to phase 1, keystroke dynamics data (when evaluated) exhibits a high degree of variability, and some might question the choice of this characteristic for the experiment. The choice was made because keystroke dynamics has operational advantages which make it practical and cost effective to implement. Provided steps were taken to reduce the effect of the high degree of variability (as much as realistically possible), it was considered that this biometric characteristic could still be useful in a biometric authentication system (particularly a multi-modal authentication system where data fusion was performed at the feature level).

In relation to phase 2, fingerprint recognition has proven accuracy in forensics and biometric authentication research and applications. However in order to facilitate ANN training and testing, and to meet the third objective of the study (feature level data fusion), it was necessary to develop a new representation method for fingerprint features, which was very different to that commonly used in biometric authentication research (refer Chapter 5 section 5.5 and the review of related research in Chapter 4 section 4.5).

The method involved a pre-processing step to register fingerprint features from participant's fingerprint samples. That is, all samples provided by the same participant were geometrically transformed such that feature correspondences were determinable. Then 8 local features common to all 140 samples (for each participant) were selected; this meant that all samples had a standard length feature vector.

In relation to phase 3, feature level data fusion (of the two previously stated biometric characteristics) was investigated because fusion at this level was expected to provide richer information upon which to base verification (in comparison to fusion performed at the decision or confidence score levels); thus making the process more accurate and reliable.

To achieve feature level data fusion, where the data sets of the two different biometric characteristics exhibited different units of measurement, data alignment of the two data sets was necessary. A simple data alignment method was applied to the transformed fingerprint feature data (in its newly represented format), so that data from this biometric could be meaningfully combined with the keystroke dynamics data. This process resulted in feature vectors of a standard length for each fingerprint, which facilitated classification by Artificial Neural Networks (ANNs) for phases 2 and 3 of the experiment.

As discussed in Chapter 2 section 2.3.1.1, there are three paradigms of data fusion. Two of the three paradigms were investigated in the experiment: the complementary and the cooperative approaches. The complementary approach combines 100% of feature level data from all sources into one feature vector, whereas the cooperative approach employs feature selection techniques and combines the selected features from each individual source into one feature vector.

Classification by an ANN is a very quick process. However, training an ANN may take some time; training time can be reduced by training an individual ANN for each legitimate user (as was done in this study), rather than training one ANN for all legitimate users (as done in most other research studies reviewed). The approach taken in the current study means that the method is scalable; if a new user needs to be added to a system, one ANN can be trained for that user. It need have no impact on the already trained ANNs for all other legitimate users.

The results for phases 1 and 2 of the experiment (the individual biometric characteristics, keystroke dynamics and fingerprint recognition) were presented in Chapter 6 sections 6.4.1 and 6.4.2 respectively, and were compared to results achieved by other researchers in their respective fields in Chapter 7 sections 7.2.1 and 7.2.2 respectively. Also, the results for the feature level data fusion phase of the experiment were presented Chapter 6 section 6.4.3, and were compared to results achieved by other researchers in that field in Chapter 7 section 7.2.3.

The next section discusses the knowledge gained from this research, and the contribution it makes to the field of biometric authentication.

## 8.2 Main Contribution of the Research

This study is one of many into biometrics for authentication. The many biometric characteristics achieve varying degrees of accuracy because of variations in instrument accuracy, raw data format, data pre-processing, software accuracy, and the uniqueness (between different people) of the biometric characteristic under consideration. Though standards for data collection methods and verification processes have been introduced in recent years, these efforts are taking some time to filter into the biometric community (research and commercial).

The results for phase 1 of the experiment (keystroke dynamics), as presented in Chapter 6 section 6.4.1 and discussed in Chapter 7 section 7.2.1, demonstrated that the methodology adopted achieved results comparable to most previous studies involving keystroke dynamics. The experimental methodology involved more participants than most studies reviewed in Chapter 3 section 3.4.1. Also, there were generally many more samples provided by participants in the current study compared to those studies reviewed.

The results for phase 2 of the experiment (fingerprint recognition), as presented in Chapter 6 section 6.4.2 and discussed in Chapter 7 section 7.2.2, demonstrated that the methodology adopted achieved results as good as or better than most previous studies involving fingerprint recognition. The experimental methodology involved more participants than most studies reviewed in Chapter 4 section 4.5. Also, there were generally many more samples provided by participants in the current study compared to those studies reviewed.

The results for phase 3 of the experiment (feature level data fusion), as presented in Chapter 6 section 6.4.3 and discussed in Chapter 7 section 7.2.3, demonstrated that the feature level data fusion achieved results as good as or better than most previous studies involving data fusion. Of particular interest is that both complementary and cooperative paradigms achieved excellent results. The experimental methodology involved more participants than most studies reviewed in Chapter 2 section 2.3.2.2. Also, there were generally many more samples provided by participants in the current study compared to those studies reviewed.

As discussed in Chapter 2, biometric verification systems rarely indicate that two samples, taken from the same person at different times, are a perfect match. Instead the system only indicates the probability that two samples are from the same person; that is, there is seldom absolute certainty.

When making the final verification decision, it is crucial that the biometric verification system attains the most accurate probability score possible. This is particularly true when multiple characteristics are being used. Early studies using multiple characteristics focused on fusing data at the decision or confidence score levels.

More recent studies have attempted data fusion at the feature level. The current study also investigated data fusion at the feature level. The concepts of decision, confidence score and feature level fusion were discussed in Chapter 2 section 2.3.2.1. Fusion at the feature level uses data closest to the raw data, and therefore is richer in feature information. It stands to reason that utilising the richness of these features would be advantageous (taking full advantage of the uniqueness of the biometric characteristics), to attain the best probability score. The current research aimed to demonstrate this, and to develop a methodology which was efficient and scalable.

At the commencement of the current study, relatively little work had been done on feature level data fusion for biometric authentication. Feature level data fusion requires combining data from different biometric characteristics in a way that the individual features of each characteristic are truly represented. From an operational perspective, this poses difficulties in relation to the additional processing required because of differences in data format between the characteristics.

In the current study, calculation of keystroke dynamics metrics of a specified character string, resulted in feature vectors proportional to the string length (thus reducing extra processing for this characteristic). Fingerprint samples required a new representation for feature data, which produced a standard length feature vector that could then be simply fused with the scalar values of keystroke dynamics metrics. Data alignment facilitated the attainment of a standard length feature vector for the combined data. This meant that the fused feature vectors (for each sample) had a common length, making the training and testing of ANNs manageable.

It was demonstrated that feature level data fusion provided accuracy in the verification process that was comparable to, and in many cases better than, previous research. The representation method used for the fingerprint features (for the data alignment and fusion purposes) may be applicable to other biometric characteristics whose extracted features include coordinates in a two dimensional plane (for example, facial features and hand geometry).

Therefore, the current investigation contributes to the field of biometrics generally, and multi-modal biometrics specifically, as follows:

- It confirmed findings in previous research that uni-modal biometric systems provide an accurate alternative to traditional authentication methods. In particular:

    1. The feature selection method, using normality statistics, developed to filter 'noisy' data from the raw data collected for this experiment, demonstrated comparable results with previous research. This reinforced previous research which indicated that filtering 'noisy' data from keystroke dynamics raw data improved verification accuracy (though other filtering methods than that used in this research may improve accuracy further).
    2. The new fingerprint representation method developed to facilitate classification of fingerprint feature data by Artificial Neural Networks (and to facilitate the fusion of this fingerprint feature data with keystroke dynamics data) demonstrated excellent results in comparison to previous research.

- It confirmed that multi-modal systems provide additional accuracy improvements as well as a perceived robustness to the verification process.

- It demonstrated the viability of the data fusion method for combining biometric data at the feature level. The results show an improvement in accuracy compared with confidence score level and decision level data fusion methods. This demonstrates the importance of using rich feature level data.

What makes this research of particular interest is the fusion of data at the feature level. As mentioned in Chapter 2, most research into multi-modal biometrics has concentrated on performing data fusion at the confidence score level, where valuable feature data is lost in the processing. Even though fusion at the confidence score level provides better accuracy and robustness than uni-modal biometric systems, the current study demonstrated that even better accuracy gains can be achieved by fusion of data at the feature level.

The methodology proposed in the current study can be generically applied to any number, and any type, of biometric characteristics provided integrity of data is ensured. As the algorithm is mathematically based, it is reasonable to assume that other algorithms exist that can also be applied to fusion at the feature level.

The next section discusses the limitations of this research.

## 8.3 Limitations of the Research

Methodological issues that may impact on the internal and/or external validity of an experiment were discussed in Chapter 5 section 5.7. In that section, the steps taken to address possible validity concerns during the design and implementation of the current experiment (which could then impact on the merit of the results achieved) were discussed.

Though care was taken with the experimental design and implementation, in order to provide as much confidence as possible in the results achieved, there were inevitably some issues that either could not be resolved within the available time-frame of the experiment or that were unforeseen during the design phase; a common theme with most research experiments.

Some of the identified minor limitations with the current experiment are:

- With the time constraints for data collection, and the particular requirements imposed on participants in collecting that data, recruitment was restricted to only 90 participants. Whilst it would have been preferred to recruit many more (in order to accredit more confidence to the results achieved), 90 was

all that could be recruited within the time-frame allocated for data collection. However, it should be noted that this number of participants is higher than most similar research projects reviewed.

- Only 100 samples per participant were available for testing false rejection (for each of the three phases of the experiment). Whilst this quantity was more than was used in most of the research efforts reviewed (refer Chapter 3 section 3.4.1, Chapter 4 section 4.5, and Chapter 2 section 2.3.2.2), it did impose a relatively course granularity on the FRR performance variable in comparison to the FAR performance variable (where 10,500 samples per participant were available for testing false acceptance). However as demonstrated by the results, this only seems to have had an impact with phase 1 of the experiment (keystroke dynamics).

- The discussion of results presented in Chapter 7 section 7.2.1, demonstrated that the feature selection process used to filter out noisy data from the keystroke dynamics raw data was not as successful as expected. Though there was good reason for using the normality statistics to improve classification accuracy during ANN training (as discussed in Chapter 5 section 5.4.4), there may be other appropriate methods of feature selection that could return improved results.

- The feature extraction process for the fingerprint recognition phase relied on third party software (refer Chapter 5 section 5.5.3). Though it was believed that the third party software would extract features accurately, no empirical proof can be furnished. However, the discussion of results presented in Chapter 7 section 7.2.2 indicated that the extraction process was accurate.

- As discussed in Chapter 5 section 5.3, no demographic data about the participants was collected[1]. However, it was believed that gender was slightly biased toward males, age was slightly biased toward people in their 20's (though there was a wide age range), and ethnicity was equally divided between Australian and international students (predominantly South East Asian). Therefore, it

[1]This was because of the sensitivity of the data and the consequent conditions under which Ethics Committee approval was granted for data collection.

cannot be claimed with certainty that the samples provided by participants (for the experiment) were representative of the general population. However, because of the nature of the data collected, there is no reason to suggest that this impacts significantly on the generalisability of the methodology or results.

- As the experiment was conducted in a predominantly English speaking country (though numerous international students participated), data collection for phase 1 of the experiment (keystroke dynamics) was performed on a keyboard with English characters. Many non-English speaking countries have at their disposal the Unicode character set. It is unknown if the results of the experiment would be impacted by the use of Unicode characters in non-English speaking countries. It is suspected that results would not be affected, but this cannot be asserted with any certainty.

An issue that remains unresolved in the field of biometrics for authentication purposes is that of a compromised registered template (for any biometric characteristic). When computers are networked, it must always be assumed that vulnerabilities do exist and that attackers could gain access to at least some part of the network. This being the case (and assuming an attacker can raise privileges to an appropriate level), authentication tokens are subject to theft or corruption.

For password-based authentication systems, a stolen or corrupted token is easily revocable simply by assigning a new password for the legitimate user in question. This of course will happen only if a breach is discovered, or when theft or corruption is reasonably suspected.

However, a specific biometric token is not revisable at all and would need to be replaced. Once compromised the characteristic associated with that biometric for that particular user can no longer be utilised. That is, the user cannot change the features associated with that particular biometric characteristic.

This means that sufficiently secure transmission and storage protocols need to be developed for data that are to be used (for both registered templates and query samples) in a biometric authentication system. To the authors knowledge, no such protocols have yet been developed.

A possible limitation with any classification problem is the method used to determine the final decision boundary. At some point a decision to verify a sample as correct or incorrect must be made. With methods associated with the traditional authentication model, this final verification decision is usually based on a straight forward comparison of discrete values. For example, when logging on to a computer system, a comparison is made between the stored value of a user's password (i.e. their registered template) and the one they supply when attempting to log on (i.e. a query sample). For authentication to be granted, the two passwords (more precisely the hashes of these passwords) must match exactly (via a character-by-character comparison). If they do match exactly, access is granted; if they do not, access is denied.

When using biometrics, the final verification decision cannot be based on a straight forward comparison of discrete values. Instead, it is typically based on the probability that two samples match. That is, the stored registered template is compared with a query sample and a probability score is produced. As mentioned in Chapter 2, it is extremely rare for two biometric samples—taken from the same person at different times—to match exactly. Definite and recognisable patterns will exist within both samples, but they will rarely be exactly the same.

So the final verification decision must allow for some flexibility, whilst still ensuring correct verification. Typically a statistical approach is employed, where the probability score is applied to an arbitrary cut-off value (i.e. final decision threshold). If the probability score meets the threshold condition (i.e. it is greater than the final decision threshold), access is granted; if it does not (i.e. it falls below the final decision threshold), access is denied.

Even though a multi-modal biometric system can be more accurate and robust than a uni-modal system, a final verification decision must still be made. Therefore, neither uni-modal nor multi-modal biometric systems have a clearly definable decision boundary. Thus the method employed to make this final verification decision (in any automated system) is a subject of concern.

It was an intention at the commencement of the current experiment, to investigate this problem and offer some empirically tested methods for making such a decision. However, time limitations did not allow for this, primarily because it was a more complex problem than first thought. How would an automated system anticipate and cater for all possible circumstances?

For human beings, a recognition decision is a matter of judgment which most of us make every day; computers do not inherently possess such recognition capabilities. If a human being is faced with incomplete data, they can try to determine more facts. They can make intuitive guesses based on prior knowledge or experience. Computers on the other hand do not have the capability to retain knowledge or learn from experience unless they are specifically programmed to do so, and attempting to program a computer to perform such tasks is very complex and difficult.

The next section discusses what might be implied from this research, and the practical considerations necessary if applying the methodology in a biometric authentication system.

## 8.4 Implications and Practical Application of the Research

From the discussion presented in Chapter 7 sections 7.2.1, 7.2.2, and 7.2.3, the results indicated the research had the following implications:

- Feature selection (i.e. pre-processing) of keystroke dynamics data does improve accuracy when compared to data that has not been pre-processed.

- The proposed fingerprint feature representation method developed for this experiment—to facilitate ANN training and testing, and feature level data fusion—provides accurate verification in both a uni-modal and a multi-modal context. It also implies that the representation method could be adapted for use by any biometric characteristic whose extracted features included coordinates in a two dimensional plane.

- By normalisation, fingerprint feature data (or data from any biometric characteristic whose extracted features include coordinates in a two dimensional plane) can be simply aligned to be compatible with a scalar valued system such as keystroke dynamics data. This results in all samples for all participants having the same length feature vectors for each individual biometric characteristic, and means that all samples for all participants have combined feature vectors of the same length, which facilitates feature level data fusion.

- Feature level data fusion indeed provides excellent verification accuracy. Though the results for the complementary approach were best overall, the cooperative approach performed nearly as well (with the exception of Stage 1). This implies that feature level data fusion may not require all data from all sources to be combined in order to attain an acceptable level of verification accuracy.

In terms of the practical implementation of the methods proposed in this dissertation, the following points require consideration:

- Feature selection of keystroke dynamics data was performed as a separate process, using the statistical software package SPSS. This approach was taken because it was deemed necessary to have available all data at all stages of the experiment, in case later analysis was required. It would be unlikely that this approach would be acceptable for an actual authentication system, because such a real-world application would require automated processing. However, many statistical software packages (such as SPSS and R) do provide command line interfacing which would allow the required statistical processing to be incorporated into an automated authentication system.

- For the representation method applied to the fingerprint recognition phase of the experiment, prior registration of the fingerprint features (for all samples) was required. This then allowed for the determination of the 8 features (that demonstrated the most accurate alignment) to be selected across all 140 samples for each participant (refer Chapter 5 section 5.5.5). It also allowed for the inclusion of 6 attributes for each fingerprint feature[2]; this provided more

[2]To the authors knowledge, this is the only study to make use of six local feature attributes.

information upon which to base verification. This seems an effective approach for practical applications.

- Considerations in relation to the feature level data fusion using the complementary paradigm eventuate because of the length of the combined feature vector. The complementary paradigm utilises 100% of the data from all sources, and as a result the combined feature vector could be inordinately large. That is, it may suffer from the curse of dimensionality. As is typical for the curse of dimensionality, the combined vector could likely contain data that is either redundant to the recognition task or has very little relevance to it. Therefore, practical applications using the complementary paradigm could investigate mechanisms to reduce the effects of this problem[3].

- Feature level data fusion using the cooperative paradigm requires the following considerations (refer Chapter 5 section 5.6.3):

  1. What percentage of the available data (from all sources) should be utilised in the selection process? The variability attributed to the biometric characteristics under consideration, and the needs of the system under development, should determine the percentage. It is entirely possible that certain percentages will not be practicable for implementation.

  2. What proportion of the chosen percentage should each of the individual data sources contribute to the newly created fused data set? Again, the answer to this question will be dependent on the variability attributed to the biometric characteristics under consideration, and the needs of the system under development.

     As illustrated in Table 5.12 (refer Chapter 5 section 5.6.3.1), the average proportionate ratio (based on average approximate relative local gains) between the keystroke dynamics and fingerprint feature was 14.1848. This average (for the 50 training group members) suggests an approximate 1:14 ratio. Though these are very rough statistics, they do imply that (on

[3]The most common approach is to perform feature selection, which from a data fusion perspective, refers to the use of the cooperative paradigm.

average) there was a much larger representation of fingerprint features in the combined feature vectors than there was keystroke dynamic metrics (for this experiment). Practical applications of this approach would need to include parameters to manage, or at least report, the ratio of usage from the multiple data sets.

3. What method or criteria should be used to determine the relevance—and subsequent selection—of features? In the current study it was decided to utilise the weights from the ANNs, that had been previously trained during the complementary data fusion phase, to select relevant features from the keystroke dynamics and fingerprint feature data sets. This was done for convenience as the weights were available, and there seemed no good reason not the utilise them for this purpose. According to Dash and Liu (1997), there are other methods available for this purpose (i.e. feature selection), although these were not investigated for the current experiment. A practical application of this approach should include a range of options for criteria selection for the relevance of features.

The next section discusses future research directions that may be stimulated by the current investigation.

## 8.5 Future Research Directions

As was apparent in the review of keystroke dynamics related research (in Chapter 3 section 3.4), there have been numerous investigative efforts to reduce (the effects of) the variability in keystroke dynamics data. The review demonstrated that feature selection does in fact reduce the effects of the data variability, at least to some degree. Also demonstrated was that the use of the keystroke duration and digraph latency metrics worked better than other contrived metrics.

In the current investigation, normality statistics were used to filter out extreme valued metrics at the tails of an assumed normal distribution (refer Chapter 5 section 5.4.4). This resulted in better accuracy compared to some other research efforts,

but not all. It is the belief of the author that research into other feature selection methods could result in further reduction in the variability of keystroke dynamics data, and thereby improve verification accuracy.

The fingerprint feature representation method developed for this experiment, returned excellent results. In the author's opinion, this methodology shows promise for both uni-modal and multi-modal applications. Future research should trial practical applications of this approach to fingerprint authentication systems.

Other biometric characteristics that could possibly benefit from this method of representation are facial recognition, hand geometry, iris and retinal recognition. The only requirement for the use of the methodology is that identified features have their locations recorded as coordinates in a two dimensional plane. Research to apply this methodology to other biometric characteristics may be advantageous.

The real benefit of the representation method became apparent when multiple biometric characteristics were used in a multi-modal authentication system, where feature level data fusion was applied. Because the representation method resulted in feature vectors of a standard length, data fusion at the feature level was easily facilitated. Further research on feature level data fusion for multi-modal authentication systems is recommended.

As mentioned in section 8.1, the traditional (password-based) authentication mechanism has associated shortcomings, but also has one major advantage: a compromised password is easily revoked. Unfortunately, the same cannot be said for a compromised registered biometric template. Once compromised, that particular biometric characteristic can no longer be used securely. That is, a biometric token cannot be replaced as simply as a password can.

A biometric token may be compromised in either of the following ways:

1. The template may be stolen, which may occur in one of two ways:

   (a) Stolen online by means of some vulnerability associated with the system, or the authentication procedure, or a network transaction. Once access is gained, an attacker could attempt to elevate their privileges to perform the unauthorised theft.

(b) Stolen by an internal source (i.e. a mischievous employee). Again once access is gained, the employee could attempt to elevate their privileges to perform the unauthorised theft.

2. A physical spoof of the particular biometric characteristic may be created to match the registered biometric template. For example, a latent fingerprint could be 'lifted' from a drink glass and transferred to a 'dummy' finger (typically composed of silicone).

A reason for the success of the above attacks (refer Chapter 1 section 1.2.1), is the lack of sufficiently secure protocols for network transmission of registered and query biometric templates during the authentication procedure. Of equal concern is the lack of sufficiently secure protocols for storage of registered biometric templates. Research to overcome limitations in these areas is vital if biometric authentication procedures are to be considered a viable alternative to the traditional (password-based) authentication model.

## 8.6 Final Remarks

The first objective of the research (phase 1 involving keystroke dynamics) did demonstrate that feature selection (to filter out the most noisy features) achieved results comparable to many of the other research efforts reviewed. However, the feature selection method chosen did not perform as well as expected. The second and third objectives (phases 2 and 3 involving fingerprint recognition and feature level data fusion respectively) demonstrated excellent performance overall and fared as well as, or better than, most other research efforts reviewed.

As just mentioned, the feature selection method chosen to filter keystroke dynamics data was not as effective as anticipated. The rationale for using the normality statistics for this purpose seemed sound, and did return comparable results the many of the other research efforts reviewed. However, other feature selection methodologies may exist which could return improved accuracy.

The fingerprint feature representation method developed for this experiment demonstrated an innovative and effective technique, whether used in a uni-modal or a multi-modal context. As the new fingerprint representation method resulted in standard length feature vectors, data alignment and subsequent feature level data fusion was efficiently and practicably facilitated.

The main contributions of the research are that uni-modal biometric systems provide an accurate alternative to traditional authentication methods, multi-modal biometric systems provide additional accuracy improvements (as well as a perceived robustness) to the verification process, and feature level data fusion provides improved accuracy compared with confidence score level and decision level data fusion methods (which demonstrates the importance of using feature rich data).

Finally, the research implies that feature selection of keystroke dynamics data does improve accuracy (when compared to data that has not been pre-processed), the proposed fingerprint feature representation method developed for this experiment provides accurate verification in both a uni-modal and a multi-modal context (as well as facilitating feature level data fusion), and feature level data fusion indeed provides excellent verification accuracy.

The results indicate that feature level data fusion may not require all data from all sources to be combined in order to attain an acceptable level of verification accuracy. This implies that a cooperative approach to data fusion offers a realistic alternative to the complementary approach, which simply combines all available data, and may offer reductions in processing time.

# Appendix A

## A.1 Reported Security Breaches And Vulnerabilities

| YEAR | INCIDENTS |
|---|---|
| 1988 | 6 |
| 1989 | 132 |
| 1990 | 252 |
| 1991 | 406 |
| 1992 | 773 |
| 1993 | 1,334 |
| 1994 | 2,340 |
| 1995 | 2,412 |
| 1996 | 2,573 |
| 1997 | 2,134 |
| 1998 | 3,734 |
| 1999 | 9,859 |
| 2000 | 21,756 |
| 2001 | 52,658 |
| 2002 | 82,094 |
| 2003 | 137,529 |

Table A.1: Reported Security Breaches (1988-2003)

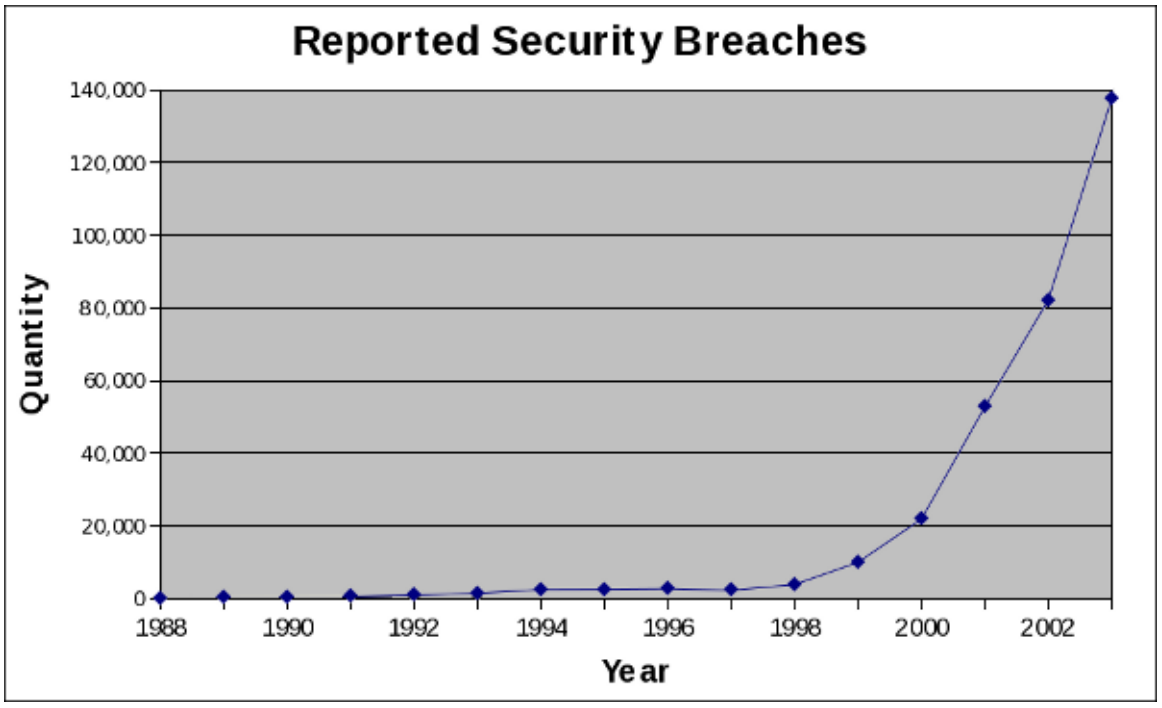


Figure A.1: Reported Security Breaches (1988-2003)

| YEAR | INCIDENTS |
|---|---|
| 1995 | 171 |
| 1996 | 345 |
| 1997 | 311 |
| 1998 | 262 |
| 1999 | 417 |
| 2000 | 1,090 |
| 2001 | 2,437 |
| 2002 | 4,129 |
| 2003 | 3,784 |
| 2004 | 3,780 |
| 2005 | 5,990 |
| 2006 | 8,064 |
| 2007 | 7,236 |
| 2008 | 7,572 |

Table A.2: Reported Vulnerabilities (1995-2008)

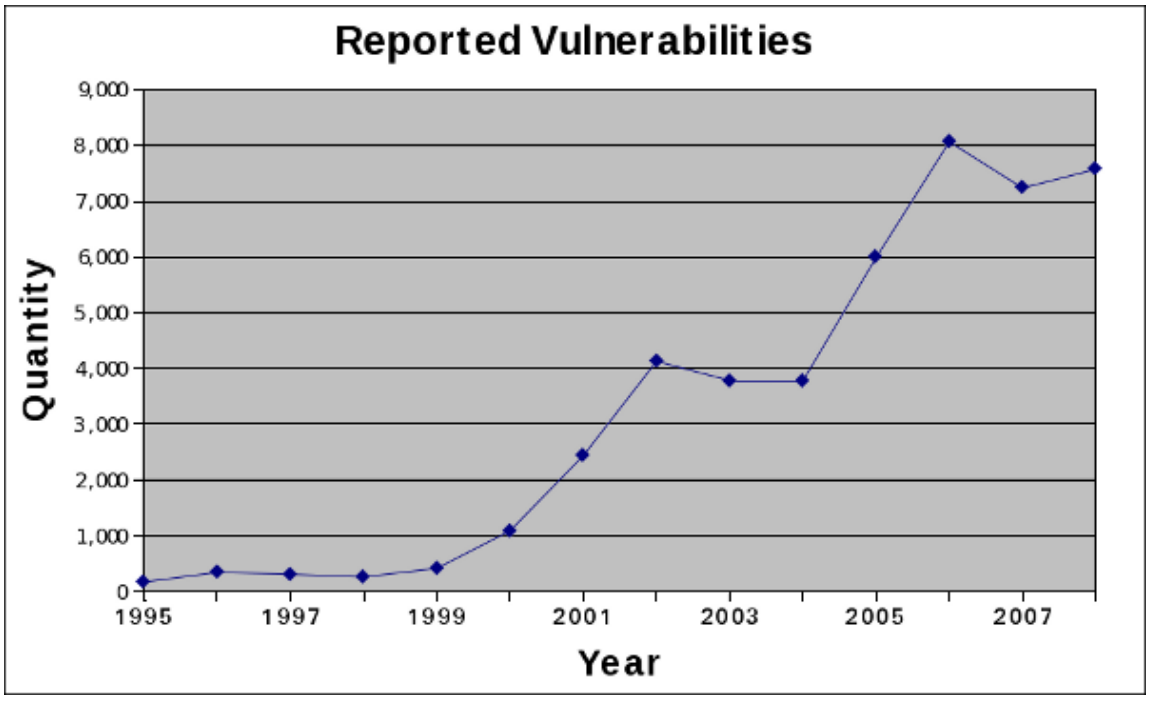


Figure A.2: Reported Vulnerabilities (1995-2008)

| DATE | NUMBER | SOURCE |
|---|---|---|
| June-02 | 580,780,000 | Nua Ltd |
| April-02 | 572,320,000 | Nua Ltd |
| February-02 | 569,140,000 | Nua Ltd |
| January-02 | 562,470,000 | Nua Ltd |
| December-01 | 552,510,000 | Nua Ltd |
| November-01 | 518,550,000 | Nua Ltd |
| October-01 | 518,940,000 | Nua Ltd |
| September-01 | 515,860,000 | Nua Ltd |
| August-01 | 515,580,000 | Nua Ltd |
| July-01 | 510,090,000 | Nua Ltd |
| June-01 | 480,870,000 | Nua Ltd |
| May-01 | 462,620,000 | Nua Ltd |
| April-01 | 460,920,000 | Nua Ltd |
| March-01 | 458,110,000 | Nua Ltd |
| February-01 | 455,550,000 | Nua Ltd |
| January-01 | 455,550,000 | Nua Ltd |
| December-00 | 451,040,000 | Nua Ltd |
| November-00 | 407,100,000 | Nua Ltd |
| October-00 | 381,790,000 | Nua Ltd |
| September-00 | 377,650,000 | Nua Ltd |
| August-00 | 368,540,000 | Nua Ltd |
| July-00 | 359,800,000 | Nua Ltd |
| June-00 | 336,520,000 | Nua Ltd |
| March-00 | 309,700,000 | Nua Ltd |
| February-00 | 280,860,000 | Nua Ltd |
| January-00 | 254,290,000 | Nua Ltd |
| September-99 | 201,050,000 | Nua Ltd |
| August-99 | 195,190,000 | Nua Ltd |
| July-99 | 185,200,000 | Nua Ltd |
| June-99 | 179,000,000 | Nua Ltd |
| May-99 | 171,250,000 | Nua Ltd |
| April-99 | 163,250,000 | Nua Ltd |
| March-99 | 159,000,000 | Nua Ltd |
| February-99 | 153,500,000 | Nua Ltd |
| December-98 | 150,000,000 | Nua Ltd |
| September-98 | 147,000,000 | Nua Ltd |
| July-98 | 129,500,000 | Nua Ltd |
| January-98 | 102,000,000 | MIDS |
| December-97 | 101,000,000 | Nua Ltd |
| November-97 | 76,000,000 | Reuters |
| September-97 | 74,000,000 | Nua Ltd |
| February-97 | 57,000,000 | MIDAS |
| December-96 | 55,000,000 | Nua Ltd |
| January-96 | 30,000,000 | Killen & Associates |
| December-95 | 26,000,000 | Nua Ltd |

Table A.3: Number of Internet Users (December 1995-June 2002)

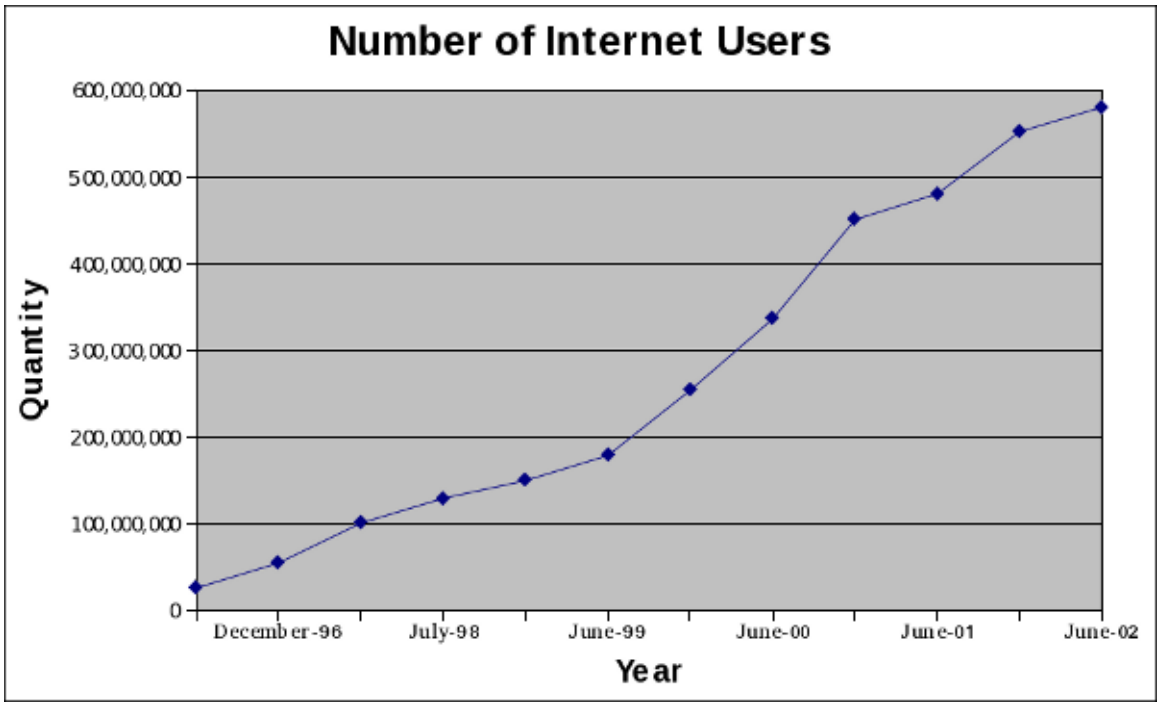


Figure A.3: Number of Internet Users (December 1995-June 2002)

# Appendix B

## B.1 Keystroke Dynamics Metrics Selection Worked Example

This appendix provides a worked example of the keystroke dynamics metrics selection process as described in Chapter 5 section 5.4.4. The example uses data from participant one. The example is presented in four stages, with the relevant data demonstrated in Tables B.1, B.2, B.3, and B.4.

Firstly, Table B.1 shows the normality, kurtosis, skewness, and standard deviation coefficients for each of the 40 metrics/variables, as calculated by the SPSS software. The data is ordered according to the metric/variable number. Note that for ranking and sorting purposes, the normality, kurtosis and skewness coefficients have been multiplied by 1000. Also, because the kurtosis and skewness coefficients may have positive or negative values—and because proximity to 0 is of primary importance for ranking purposes—their absolute values are shown.

Table B.2 illustrates the same data as that presented in Table B.1, but with the individual statistics sorted as described in section 5.4.4. The normality statistics have been sorted in descending order from those closest to 1000 (refer section 5.4.4). The kurtosis, skewness, and standard deviation statistics have been sorted in ascending order from those closest to 0 (refer section 5.4.4). Note that the association between the individual statistics and their metric numbers have been maintained.

| Metric | Normality | Kurtosis | Skewness | Standard Deviation |
|---|---|---|---|---|
| 1 | 847 | 813 | 301 | 19.02006 |
| 2 | 920 | 2453 | 1159 | 35.63588 |
| 3 | 827 | 164 | 442 | 18.11700 |
| 4 | 882 | 239 | 73 | 22.76521 |
| 5 | 871 | 103 | 58 | 20.75653 |
| 6 | 938 | 784 | 42 | 34.63968 |
| 7 | 820 | 538 | 284 | 18.47439 |
| 8 | 896 | 530 | 607 | 34.60668 |
| 9 | 875 | 470 | 268 | 23.73231 |
| 10 | 946 | 972 | 57 | 38.25162 |
| 11 | 828 | 1677 | 97 | 18.76443 |
| 12 | 947 | 3131 | 1373 | 37.03994 |
| 13 | 796 | 1167 | 635 | 16.29006 |
| 14 | 902 | 5438 | 1164 | 26.59559 |
| 15 | 914 | 62 | 607 | 33.31021 |
| 16 | 908 | 926 | 768 | 42.62738 |
| 17 | 950 | 411 | 157 | 34.00310 |
| 18 | 851 | 7981 | 2096 | 30.01648 |
| 19 | 930 | 466 | 293 | 29.06955 |
| 20 | 949 | 37 | 453 | 43.14939 |
| 21 | 856 | 787 | 191 | 20.18614 |
| 22 | 933 | 1329 | 134 | 51.11554 |
| 23 | 893 | 1790 | 1001 | 40.66103 |
| 24 | 949 | 515 | 404 | 48.02088 |
| 25 | 893 | 348 | 191 | 21.13491 |
| 26 | 872 | 479 | 170 | 22.87560 |
| 27 | 905 | 758 | 603 | 27.74115 |
| 28 | 934 | 357 | 146 | 28.95792 |
| 29 | 921 | 96 | 465 | 25.83840 |
| 30 | 976 | 29 | 134 | 58.66757 |
| 31 | 909 | 537 | 320 | 23.15184 |
| 32 | 966 | 929 | 415 | 53.01140 |
| 33 | 941 | 79 | 533 | 40.74174 |
| 34 | 958 | 531 | 63 | 40.18571 |
| 35 | 936 | 236 | 312 | 32.17796 |
| 36 | 870 | 1418 | 253 | 23.90844 |
| 37 | 933 | 182 | 270 | 29.05820 |
| 38 | 905 | 192 | 129 | 27.69836 |
| 39 | 914 | 111 | 239 | 29.21093 |
| 40 | 969 | 251 | 228 | 54.11489 |

Table B.1: Coefficient Values For Each Metric

| Metric | Normality | Metric | Kurtosis | Metric | Skewness | Metric | Standard Deviation |
|---|---|---|---|---|---|---|---|
| 30 | 976 | 20 | 37 | 21 | 3 | 13 | 16.2901 |
| 40 | 969 | 15 | 62 | 30 | 6 | 3 | 18.1170 |
| 32 | 966 | 32 | 64 | 32 | 22 | 7 | 18.4744 |
| 34 | 958 | 33 | 79 | 36 | 41 | 11 | 18.7644 |
| 17 | 950 | 34 | 93 | 6 | 42 | 1 | 19.0201 |
| 20 | 949 | 5 | 103 | 31 | 49 | 21 | 20.1861 |
| 24 | 949 | 39 | 111 | 5 | 58 | 5 | 20.7565 |
| 12 | 947 | 14 | 137 | 4 | 73 | 25 | 21.1349 |
| 10 | 946 | 3 | 164 | 10 | 93 | 4 | 22.7652 |
| 33 | 941 | 37 | 182 | 14 | 95 | 26 | 22.8756 |
| 6 | 938 | 38 | 192 | 11 | 97 | 31 | 23.1518 |
| 35 | 936 | 29 | 202 | 38 | 129 | 9 | 23.7323 |
| 28 | 934 | 35 | 236 | 26 | 133 | 36 | 23.9084 |
| 22 | 933 | 4 | 239 | 28 | 146 | 29 | 25.8384 |
| 37 | 933 | 40 | 251 | 17 | 157 | 14 | 26.5956 |
| 19 | 930 | 31 | 299 | 25 | 191 | 38 | 27.6984 |
| 29 | 921 | 30 | 307 | 40 | 228 | 27 | 27.7412 |
| 2 | 920 | 25 | 348 | 39 | 239 | 28 | 28.9579 |
| 15 | 914 | 28 | 357 | 9 | 268 | 37 | 29.0582 |
| 39 | 914 | 17 | 411 | 37 | 270 | 19 | 29.0695 |
| 31 | 909 | 21 | 426 | 7 | 284 | 39 | 29.2109 |
| 16 | 908 | 26 | 438 | 34 | 290 | 18 | 30.0165 |
| 27 | 905 | 19 | 466 | 19 | 293 | 35 | 32.1780 |
| 38 | 905 | 9 | 470 | 1 | 301 | 15 | 33.3102 |
| 14 | 902 | 12 | 499 | 35 | 312 | 17 | 34.0031 |
| 8 | 896 | 24 | 515 | 29 | 342 | 8 | 34.6067 |
| 23 | 893 | 8 | 530 | 24 | 404 | 6 | 34.6397 |
| 25 | 893 | 7 | 538 | 12 | 412 | 2 | 35.6359 |
| 4 | 882 | 22 | 557 | 3 | 442 | 12 | 37.0399 |
| 9 | 875 | 18 | 616 | 20 | 453 | 10 | 38.2516 |
| 26 | 872 | 27 | 758 | 33 | 533 | 34 | 40.1857 |
| 5 | 871 | 6 | 784 | 27 | 603 | 23 | 40.6610 |
| 36 | 870 | 1 | 813 | 8 | 607 | 33 | 40.7417 |
| 21 | 856 | 10 | 829 | 15 | 607 | 16 | 42.6274 |
| 18 | 851 | 16 | 926 | 13 | 635 | 20 | 43.1494 |
| 1 | 847 | 36 | 1082 | 22 | 661 | 24 | 48.0209 |
| 11 | 828 | 13 | 1167 | 2 | 758 | 22 | 51.1155 |
| 3 | 827 | 2 | 1246 | 16 | 768 | 32 | 53.0114 |
| 7 | 820 | 11 | 1677 | 18 | 770 | 40 | 54.1149 |
| 13 | 796 | 23 | 1790 | 23 | 1001 | 30 | 58.6676 |

Table B.2: Sorted Coefficient Values And Associated Metric Numbers

Table B.3 demonstrates the rankings allocated to the metrics according to the ordered individual statistics that were illustrated in Table B.2. The rankings were assigned as specified in Table 5.1 (refer Chapter 5 section 5.4.4 page 241). Again, the association between the ordered individual statistics and their metric numbers were maintained.

| Metric | Normality | Metric | Kurtosis | Metric | Skewness | Metric | Standard Deviation |
|---|---|---|---|---|---|---|---|
| 30 | 1 | 20 | 0.5 | 21 | 0.2 | 13 | 0.2 |
| 40 | 0.975 | 15 | 0.4875 | 30 | 0.195 | 3 | 0.195 |
| 32 | 0.95 | 32 | 0.475 | 32 | 0.19 | 7 | 0.19 |
| 34 | 0.925 | 33 | 0.4625 | 36 | 0.185 | 11 | 0.185 |
| 17 | 0.9 | 34 | 0.45 | 6 | 0.18 | 1 | 0.18 |
| 20 | 0.875 | 5 | 0.4375 | 31 | 0.175 | 21 | 0.175 |
| 24 | 0.85 | 39 | 0.425 | 5 | 0.17 | 5 | 0.17 |
| 12 | 0.825 | 14 | 0.4125 | 4 | 0.165 | 25 | 0.165 |
| 10 | 0.8 | 3 | 0.4 | 10 | 0.16 | 4 | 0.16 |
| 33 | 0.775 | 37 | 0.3875 | 14 | 0.155 | 26 | 0.155 |
| 6 | 0.75 | 38 | 0.375 | 11 | 0.15 | 31 | 0.15 |
| 35 | 0.725 | 29 | 0.3625 | 38 | 0.145 | 9 | 0.145 |
| 28 | 0.7 | 35 | 0.35 | 26 | 0.14 | 36 | 0.14 |
| 22 | 0.675 | 4 | 0.3375 | 28 | 0.135 | 29 | 0.135 |
| 37 | 0.65 | 40 | 0.325 | 17 | 0.13 | 14 | 0.13 |
| 19 | 0.625 | 31 | 0.3125 | 25 | 0.125 | 38 | 0.125 |
| 29 | 0.6 | 30 | 0.3 | 40 | 0.12 | 27 | 0.12 |
| 2 | 0.575 | 25 | 0.2875 | 39 | 0.115 | 28 | 0.115 |
| 15 | 0.55 | 28 | 0.275 | 9 | 0.11 | 37 | 0.11 |
| 39 | 0.525 | 17 | 0.2625 | 37 | 0.105 | 19 | 0.105 |
| 31 | 0.5 | 21 | 0.25 | 7 | 0.1 | 39 | 0.1 |
| 16 | 0.475 | 26 | 0.2375 | 34 | 0.095 | 18 | 0.095 |
| 27 | 0.45 | 19 | 0.225 | 19 | 0.09 | 35 | 0.09 |
| 38 | 0.425 | 9 | 0.2125 | 1 | 0.085 | 15 | 0.085 |
| 14 | 0.4 | 12 | 0.2 | 35 | 0.08 | 17 | 0.08 |
| 8 | 0.375 | 24 | 0.1875 | 29 | 0.075 | 8 | 0.075 |
| 23 | 0.35 | 8 | 0.175 | 24 | 0.07 | 6 | 0.07 |
| 25 | 0.325 | 7 | 0.1625 | 12 | 0.065 | 2 | 0.065 |
| 4 | 0.3 | 22 | 0.15 | 3 | 0.06 | 12 | 0.06 |
| 9 | 0.275 | 18 | 0.1375 | 20 | 0.055 | 10 | 0.055 |
| 26 | 0.25 | 27 | 0.125 | 33 | 0.05 | 34 | 0.05 |
| 5 | 0.225 | 6 | 0.1125 | 27 | 0.045 | 23 | 0.045 |
| 36 | 0.2 | 1 | 0.1 | 8 | 0.04 | 33 | 0.04 |
| 21 | 0.175 | 10 | 0.0875 | 15 | 0.035 | 16 | 0.035 |
| 18 | 0.15 | 16 | 0.075 | 13 | 0.03 | 20 | 0.03 |
| 1 | 0.125 | 36 | 0.0625 | 22 | 0.025 | 24 | 0.025 |
| 11 | 0.1 | 13 | 0.05 | 2 | 0.02 | 22 | 0.02 |
| 3 | 0.075 | 2 | 0.0375 | 16 | 0.015 | 32 | 0.015 |
| 7 | 0.05 | 11 | 0.025 | 18 | 0.01 | 40 | 0.01 |
| 13 | 0.025 | 23 | 0.0125 | 23 | 0.005 | 30 | 0.005 |

Table B.3: Sorted Metrics With Rank Allocations

Table B.4 illustrates the results of accumulating the allocated rankings for the four statistics (for each metric/variable). The metrics are in ascending order; columns 2 to 5 show the allocated rank for the individual statistics for the corresponding metrics; column 6 provides the sum of the four statistics per metric. The selected metrics for this participant are those with the highest accumulated scores, and are indicated by the '*' character along side the total in column 6.

| Metric | Normality | Kurtosis | Skewness | Standard Deviation | Total |
|---|---|---|---|---|---|
| 1 | 0.125 | 0.1 | 0.085 | 0.18 | 0.49 |
| 2 | 0.575 | 0.0375 | 0.02 | 0.065 | 0.6975 |
| 3 | 0.075 | 0.4 | 0.06 | 0.195 | 0.73 |
| 4 | 0.3 | 0.3375 | 0.165 | 0.16 | 0.9625 |
| 5 | 0.225 | 0.4375 | 0.17 | 0.17 | 1.0025 |
| 6 | 0.75 | 0.1125 | 0.18 | 0.07 | 1.1125* |
| 7 | 0.05 | 0.1625 | 0.1 | 0.19 | 0.5025 |
| 8 | 0.375 | 0.175 | 0.04 | 0.075 | 0.665 |
| 9 | 0.275 | 0.2125 | 0.11 | 0.145 | 0.7425 |
| 10 | 0.8 | 0.0875 | 0.16 | 0.055 | 1.1025* |
| 11 | 0.1 | 0.025 | 0.15 | 0.185 | 0.46 |
| 12 | 0.825 | 0.2 | 0.065 | 0.06 | 1.15* |
| 13 | 0.025 | 0.05 | 0.03 | 0.2 | 0.305 |
| 14 | 0.4 | 0.4125 | 0.155 | 0.13 | 1.0975* |
| 15 | 0.55 | 0.4875 | 0.035 | 0.085 | 1.1575* |
| 16 | 0.475 | 0.075 | 0.015 | 0.035 | 0.6 |
| 17 | 0.9 | 0.2625 | 0.13 | 0.08 | 1.3725* |
| 18 | 0.15 | 0.1375 | 0.01 | 0.095 | 0.3925 |
| 19 | 0.625 | 0.225 | 0.09 | 0.105 | 1.045 |
| 20 | 0.875 | 0.5 | 0.055 | 0.03 | 1.46* |
| 21 | 0.175 | 0.25 | 0.2 | 0.175 | 0.8 |
| 22 | 0.675 | 0.15 | 0.025 | 0.02 | 0.87 |
| 23 | 0.35 | 0.0125 | 0.005 | 0.045 | 0.4125 |
| 24 | 0.85 | 0.1875 | 0.07 | 0.025 | 1.1325* |
| 25 | 0.325 | 0.2875 | 0.125 | 0.165 | 0.9025 |
| 26 | 0.25 | 0.2375 | 0.14 | 0.155 | 0.7825 |
| 27 | 0.45 | 0.125 | 0.045 | 0.12 | 0.74 |
| 28 | 0.7 | 0.275 | 0.135 | 0.115 | 1.225* |
| 29 | 0.6 | 0.3625 | 0.075 | 0.135 | 1.1725* |
| 30 | 1 | 0.3 | 0.195 | 0.005 | 1.5* |
| 31 | 0.5 | 0.3125 | 0.175 | 0.15 | 1.1375* |
| 32 | 0.95 | 0.475 | 0.19 | 0.015 | 1.63* |
| 33 | 0.775 | 0.4625 | 0.05 | 0.04 | 1.3275* |
| 34 | 0.925 | 0.45 | 0.095 | 0.05 | 1.52* |
| 35 | 0.725 | 0.35 | 0.08 | 0.09 | 1.245* |
| 36 | 0.2 | 0.0625 | 0.185 | 0.14 | 0.5875 |
| 37 | 0.65 | 0.3875 | 0.105 | 0.11 | 1.2525* |
| 38 | 0.425 | 0.375 | 0.145 | 0.125 | 1.07* |
| 39 | 0.525 | 0.425 | 0.115 | 0.1 | 1.165* |
| 40 | 0.975 | 0.325 | 0.12 | 0.01 | 1.43* |

Table B.4: Accumulated Rank Score For Metrics

Note that the selected metric correspond to those presented in Table 5.2 Chapter 5 section 5.4.4 page 244. Note also that in Table 5.2 the index numbers are illustrated, whereas Table B.4 presents the metric numbers (the metric numbers are one more than the index numbers).

# Appendix C

This appendix provides information about the programs developed for the three phases of the experiment (as described in Chapter 5 sections 5.4, 5.5, and 5.6). As such, there are three sections to this appendix; one for each phase. Software are listed in the sequence that they were developed. Also, a basic description of the tasks they were designed to perform is provided[1].

## C.1 Keystroke Dynamics Phase Software

As presented in Chapter 5 section 5.4.1, the program used to capture the raw data from keystroke events was developed using Borland C++ Builder. This program presented participants with a Graphical User Interface (GUI) to guide their data entry (see Figure 5.1). It captured and recorded the time values associated with keystroke events at a 1 millisecond resolution, and included mechanisms to filter erroneous participant input.

For the development of the remainder of the software the following directory structure was required. A 'keystrokes' directory contained all software for this phase of the experiment. A parent directory to the keystrokes directory was required for the ANN software. This directory 'mbpann' contained all software related to the Matrix Back Propagation Artificial Neural Network, including the executable programs to train and test ANN's ('mbp' and 'mbpval' respectively).

Table C.1 illustrates the subdirectories (and their contents) that were required under the keystrokes directory. Their requirement will become clear as the software is described.

[1]For a more in depth description, refer to the relevant section in Chapter 5

| Directory | Contents |
|---|---|
| analysis | described in section C.1.1 |
| metrics_data | files containing selected metrics |
| metrics_norm | files containing normalised selected metrics |
| metrics_original | files containing originally extracted metrics |
| non_train | non-training data files (used for testing only) |
| raw_data | files captured during data collection |
| results | results from testing |
| train_data | training data files |
| test_data | testing data files |
| validation_data | validation (during training) data files |

Table C.1: Keystroke Dynamics Directory Structure

### C.1.1 Pre-processing

Under the keystrokes/analysis/ directory, the following pre-processes were implemented (in the order listed), using the specified software:

1. **Replace Outliers:** Prior to SPSS analysis, the script replaceOutliers.pl reads original metrics from keystrokes/metrics_original/m???.txt files, calculates the mean and standard deviation for each metric/variable across all 140 samples, and determines outlier values. Discounting the outlier values, new mean and standard deviation values are calculated. The outlier values in the data files are then replaced with the newly calculated mean. The new data sets (with outliers replaced) are output to spss_data/m???.txt, and used for SPSS analysis.

2. **SPSS Analysis:** Determine normality, kurtosis, and skewness coefficients and standard deviation for the 40 metric/variables (across 140 samples) for each participant. Normality outputs are stored in normality/???norm.txt; kurtosis and skewness outputs are stored in skewkurt/???zsk.txt; standard deviation outputs are stored in stats/???msd.txt.

3. **Extract Normality Coefficients:** After SPSS analysis is completed, the script extract_norm.pl reads normality/???norm.txt files, extracts the normality values, multiplies them by 1000, and outputs to normality/???norm.out files.

4. **Extract Kurtosis and Skewness Coefficients:** After SPSS analysis is completed, the script extract_skewkurt.pl reads skewkurt/???zsk.txt files, extracts the kurtosis and skewness values, multiplies them by 1000, calculates their absolute values, and outputs to skewkurt/???zsk.out files.

5. **Extract Standard Deviation:** After SPSS analysis is completed, the script extract_stats.pl reads stats/???msd.txt files, extracts the mean and standard deviation values, and outputs to stats/???msd.out files.

6. **Rank Statistics:** The script sortMetrics.pl reads all .out files from the normality, skewkurt, and stats directories, sorts each files data into appropriate order (i.e. ranks the normality coefficient in descending order, and the kurtosis and skewness coefficients and standard deviation in ascending order, whilst maintaining the association between the metric/variable numbers and the coefficients), and outputs to spss_output/???.out file.

7. **Select Metrics:** The script selectMetrics.pl reads from spss_data/m???.out and from spss_output/???.out files, and determines the top 20 metrics based on combined ranks from normality, kurtosis, skewness, and standard deviation. Output is written to spss_output/???.met files.

8. **Extract Metrics:** The script extractMetrics.pl reads from spss_output/???.met and from keystrokes/metrics_original/m???.txt files. The best 20 metrics determined by the selectMetrics.pl script are used to select the actual metrics from keystrokes/metrics_original/m???.txt files. The global statistics are also calculated and pre-pended to the chosen metrics. Output is written to keystrokes/metrics_data/m???.txt files.

### C.1.2 Experimental Procedure

Under the keystrokes directory, the following procedures were implemented (in the order listed), using the specified software:

1. **Metrics Calculation:** The script metrics.pl reads from raw_data/???.txt files, calculates the metrics and outputs to metrics_original/m???.txt.

2. **Metrics Selection:** After the calculation of metrics, data analysis was performed on data in metrics_original (refer section C.1.1 above). SPSS software was used to obtain relevant statistics (i.e. normality, kurtosis, and skewness coefficients and standard deviation) used in the keystroke dynamics metric selection process. Selected metrics were output to metrics_data/m???.txt files.

3. **Normalisation:** The script normalise.pl reads from metrics_data/m???.txt files, normalises the chosen metrics (using the min/max normalisation method) and writes output to files in metrics_norm/m???.txt files.

4. **Member Selection:** The script randParts.pl randomly allocates participants to the training and non-training groups; the non-training group members data files are moved from metrics_norm to non_train directory (these data are used for testing purposes only), and the training group members data files are moved from metrics_norm to train_data directory.

5. **Validation Data:** The script valFiles.pl randomly selects 10 samples (from 140 available) from each training group members metrics file (train_data/m???.txt). The chosen samples are written to validation_data/val??? files, and are used for cross validation during training. Note the selected samples are removed from the metrics files, leaving 130 available in each training group members metrics file.

6. **Training Samples:** The script randSamples.pl randomly selects 30 training samples (from the remaining 130) from each training group members metrics file (train_data/m???.txt). The chosen samples are written to train_data/trn??? files, and are used for training purposes only. The remaining 100 samples for each training group member (left after training and validation samples have been removed) are used as testing samples and are written to test_data/tst??? files.

7. **Training Data:** The script trnFiles.pl creates the ANN training files for training group members. It reads from train_data/trn??? files and writes to

train_data/trf???. For each training group member, the script randomly selects (without removing) one training sample from each of the other 49 training group members, and appends these to the 30 training samples selected for that member. This results in 79 samples (30+49) per training file per training group member.

8. **Testing Data** The script tstFiles.pl creates the ANN testing files for training group members. It reads from non_train/tst??? files and test_data/tst??? files, then writes to test_data/tsf???. For each training group member, all 140 samples from each non-training group member ($140 * 40 = 5,600$) and 100 testing samples from each of the other training group members ($100 * 49 = 4,900$), are appended to the 100 testing samples for that member (giving 10,600 samples).

9. **Training The ANN:** The script execTrn.pl executes ANN training using the executable program 'mbp' in the 'mbpann' directory.

10. **Testing The ANN:** The script execTst.pl executes ANN testing using the executable program 'mbpval' in the 'mbpann' directory.

11. **Best Configuration:** The script numMLN.pl compiles information to help select the best ANN configuration. That is, the ANN configuration (more precisely the number of hidden layer neurons) that performed with best accuracy based on the outcome from testing. After running numMLN.pl, the procedure required manually selecting possible candidate configurations.

12. **Calculate Results:** The script runROC.pl calls the executable program 'calcROC'. This program reads from test_data/*.out files, then calculates and plots the ROC points on an ROC graph. The final decision threshold is determined and the FAR and FRR performance variables (at that threshold) calculated. Output is written to results/???.dat.

The following target files were required for training and testing the ANN:

**File kd.trn:** The target file for training the keystroke dynamics data contained 79 lines. The target for positive case training samples was the value 1. Therefore, the first 30 lines were 1's and corresponded to the 30 positive case training samples. The target for the negative case training samples was -1. Therefore, the last 49 lines were -1's and corresponded to the 49 negative case training samples.

**File kd.tst:** The target file for testing the keystroke dynamics data contained 10,600 lines. The target for positive case testing samples was the value 1. Therefore, the first 100 lines were 1's and corresponded to the 100 positive case testing samples. The target for the negative case testing samples was -1. Therefore, the remaining 10,500 lines were -1's and corresponded to the 10,500 negative case testing samples.

**File kd.val:** The target file for cross validation (during training) of the keystroke dynamics data contained 10 lines, with each line providing a correspondence between the target value 1 and a validation sample.

## C.2 Fingerprint Recognition Phase Software

As presented in Chapter 5 section 5.5.1, the program used to capture fingerprint feature data was developed using Visual C++. This program presented participants with a Graphical User Interface (GUI) to facilitate the fingerprint scanning procedure (see Figure 5.3). The program utilised the Verifinger Software Development Kit (SDK) 4.2 Visual C++ library to interface with the fingerprint scanning device and to extract the fingerprint features.

For the development of the remainder of the software the following directory structure was required. A 'fingerprints' directory contained all software for this phase of the experiment. Again a parent directory (mbpann) to the fingerprints directory contained all software related to the ANN utilised.

Table C.2 illustrates the subdirectories (and their contents) that were required under the fingerprints directory. Their requirement will become clear as the software is described.

| Directory | Contents |
|---|---|
| data | output from executing ppm program |
| feature_data | files containing extracted features from captured data |
| metrics_data | files containing selected metrics |
| metrics_norm | files containing normalised selected metrics_mm |
| non_train | non-training data files (used for testing only) |
| preprocess | described in section C.2.1 |
| raw_data | .raw and .msd files captured during data collection |
| results | results from testing |
| train_data | training data files |
| test_data | test data files |
| validation_data | validation files (during training) |

Table C.2: Fingerprint Recognition Directory Structure

### C.2.1 Pre-processing

Under the fingerprint/preprocess directory, the following pre-processes were implemented (in the order listed), using the specified software:

1. **Calculate Means:** The script calcMeans.pl reads from fingerprints/raw_data/*.msd files and outputs to ./means.out. It determines the frequency of the minutiae counts that occur in all samples per participant (importantly, the minutiae count with the highest frequency). This information is required by the next script – runfmi.pl.

2. **Determine Model Image:** The script runfmi.pl passes the following information to the executable program 'findModelImage' (per participant): the total number of samples, the minutiae count that occurs most frequently, and the number of times that minutiae count occurs. The program uses this information in determining the model image for each participant. Input is read from fingerprints/feature_data/*.ftr, and output is written to index.ref.

3. **Align Images:** The script runppm.pl passes the the sample number of a participants model image to the executable program 'ppm'. The program aligns all samples from the same participant with their model image (using the point pattern matching algorithm described in Chapter 5 section 5.5.4). Input is read from fingerprints/feature_data/*.ftr, and written to fingerprints/data/*.err, fingerprints/data/*.ppm, and fingerprints/data/*.tab.

3. **Non-Aligned Images:** The script findError.pl reads from fingerprints/data/*.err files and outputs to fingerprints/data/*.out. Output from this script identifies the number of samples (per participant) that were not correctly aligned. N.B. Each participant required 140 aligned samples.

4. **Extract Metrics:** The script extractMetics.pl reads from fingerprints/data/*.err, fingerprints/data/*.ppm, and fingerprints/data/*.tab files and outputs to ../metrics_data/m???.txt (also redirects stdout to the file a.out on command-line). The information in a.out and fingerprints/data/*.tab files are used to determine whether any participants data did not provide 140 samples for training. Note was it deemed acceptable to allow one feature to be missing in approximately 40 samples. If any participants data exceeded this restriction, their data was seriously considered for replacement.

### C.2.2 Experimental Procedure

Under the fingerprints directory, the following procedures were implemented (in the order listed), using the specified software:

1. **Normalisation:** The script normalise.pl reads files from metrics_data/m???.txt, normalises the chosen metrics (using the min/max normalisation method) and writes output to files in metrics_norm/m???.txt.

2. **Order By Means:** The script selectBestMeans.pl reads from metrics_data/m???.txt files and write output to metrics_data/???.means file. The script orders the samples according to metric proximity to their means. The samples are written in ascending order (per line) for all 140 samples, and are used in the selection of validation and training samples.

3. **Member Selection:** The script randParts.pl uses the same allocation of participants (to the training and non-training groups) as was used in section C.1.2 point 4. The non-training group members data files are moved from metrics_norm to non_train directory (these data are used for testing purposes only), and the training group members data files are moved from metrics_norm to train_data directory.

4. **Validation Data:** The script valFiles.pl randomly selects 10 samples (from 140 available) from each training group members metrics file (i.e. train_data/m???.txt and metrics_data/???.means). The chosen samples are written to validation_data/val??? files, and are used for cross validation during training. Note the selected samples are removed from the metrics files, leaving 130 available in each training group members metrics file.

5. **Training Samples:** The script randSamples.pl randomly selects 30 training samples (from the remaining 130) from each training group members metrics file (i.e. train_data/m???.txt and metrics_data/???.means). The chosen samples are written to train_data/trn??? files, and are used for training purposes only. The remaining 100 samples for each training group member (left after training and validation samples have been removed) are used as testing samples and are written to test_data/tst??? files.

6. **Training Data:** The script trnFiles.pl creates the ANN training files for training group members. It reads from train_data/trn??? files and writes to train_data/trf???. For each training group member, the script randomly selects (without removing) one training sample from each of the other 49 training group members, and appends these to the 30 samples for that member. This results in 79 samples $(30 + 49)$ per training file per training group member.

7. **Testing Data** The script tstFiles.pl creates the ANN testing files for training group members. It reads from non_train/tst??? files and test_data/tst??? files, then writes to test_data/tsf???. For each training group member, all 140 samples from each non-training group member $(140 * 40 = 5,600)$ and 100 testing samples from each of the other training group members $(100 * 49 = 4,900)$, are appended to the 100 testing samples for that member (giving 10,600 samples).

8. **Training The ANN:** The script execTrn.pl executes ANN training using the executable program 'mbp' in the 'mbpann' directory.

**9. Testing The ANN:** The script execTst.pl executes ANN testing using the executable program 'mbpval' in the 'mbpann' directory.

**10. Best Configuration:** The script numMLN.pl compiles information to help select the best ANN configuration. That is, the ANN configuration (more precisely the number of hidden layer neurons) that performed with best accuracy based on the outcome from testing. After running numMLN.pl, the procedure required manually selecting possible candidate configurations.

**12. Calculate Results:** The script runROC.pl calls the executable program 'calcROC'. This program reads from test_data/*.out files, then calculates and plots the ROC points on an ROC graph. The final decision threshold is determined and the FAR and FRR performance variables (at that threshold) calculated. Output is written to results/???.dat.

The following target files were required for training and testing the ANN:

**File fp.trn:** As for the file kd.trn (described in section C.1.2), the target file for training the fingerprint recognition data contained 30 lines of 1's and 49 lines were -1's.

**File fp.tst:** As for the file kd.tst (described in section C.1.2), the target file for testing the fingerprint recognition data contained 100 lines of 1's and 10,500 lines were -1's.

**File fp.val:** As for the file kd.val (described in section C.1.2), the target file for cross validation (during training) of the fingerprint recognition data contained 10 lines of 1's.

## C.3 Data Fusion Phase Software

As the data fusion phase utilised the data sets from the previous two phases of the experiment, no capture program or feature selection or pre-processing of raw data was required. This meant that for this of phase of the experiment, software

to perform data fusion was all that was required. The development of the software required for the data fusion phase, was undertaken for the two data fusion paradigms discussed in Chapter 2 section 2.3.1.1 (complementary and cooperative). The next two section list the sequence in which the software were developed, and the tasks that they were designed to perform.

A 'dataFusion' directory contained all software for this phase of the experiment. Again a parent directory (mbpann) to the dataFusion directory contained all software related to the ANN utilised.

### C.3.1 Complementary Data Fusion Software

Under the dataFusion directory, a 'complementary' directory contained all software for this phase of the experiment. Table C.3 illustrates the subdirectories (and their contents) that were required under the complementary directory. Their requirement will become clear as the software is described.

| Directory | Contents |
|---|---|
| results | results from testing |
| train_data | training data files |
| test_data | test data files |
| validation_data | validation files (during training) |

Table C.3: Complementary Data Fusion Directory Structure

Under the dataFusion/complementary directory, the following procedures were implemented (in the order listed), using the specified software:

1. **Merge Data:** The script mergeData.pl reads the following files from under the keystrokes and fingerprints directories:

   - fingerprints/validation_data/vaf???
   - fingerprints/train_data/trf???
   - fingerprints/test_data/tsf???
   - fingerprints/non_train/tst???
   - keystrokes/validation_data/vaf???

- keystrokes/train_data/trf???
- keystrokes/test_data/tsf???
- keystrokes/non_train/tst???

The script concatenates the fingerprint recognition data to end of the keystroke dynamics data in each of the associated files (as set out above), and writes the merged data to corresponding files in the train_data, test_data, and validation_data directories.

2. **Training The ANN:** The script execTrn.pl executes ANN training using the executable program 'mbp' in the 'mbpann' directory.

3. **Testing The ANN:** The script execTst.pl executes ANN testing using the executable program 'mbpval' in the 'mbpann' directory.

4. **Best Configuration:** The script numMLN.pl compiles information to help select the best ANN configuration. That is, the ANN configuration (more precisely the number of hidden layer neurons) that performed with best accuracy based on the outcome from testing. After running numMLN.pl, the procedure required manually selecting possible candidate configurations.

5. **Calculate Results:** The script runROC.pl calls the executable program 'calcROC'. This program reads from test_data/*.out files, then calculates and plots the ROC points on an ROC graph. The final decision threshold is determined and the FAR and FRR performance variables (at that threshold) calculated. Output is written to results/???.dat.

The following target files were required for training and testing the ANN:

**File df.trn:** As for the file kd.trn (described in section C.1.2), the target file for training the complementary data fusion data contained 30 lines of 1's and 49 lines were -1's.

**File df.tst:** As for the file kd.tst (described in section C.1.2), the target file for testing the complementary data fusion data contained 100 lines of 1's and 10,500 lines were -1's.

**File df.val:** As for the file kd.val (described in section C.1.2), the target file for cross validation (during training) of the complementary data fusion data contained 10 lines of 1's.

### C.3.2 Cooperative Data Fusion Software

Under the dataFusion directory, a 'cooperative' directory contained all software for this phase of the experiment. Table C.4 illustrates the subdirectories (and their contents) that were required under the complementary directory. Their requirement will become clear as the software is described.

| Directory | | Contents |
|---|---|---|
| fp | - | files containing approximate relative local gains of fingerprint metrics. |
| kd | - | files containing approximate relative local gains of keystroke metrics. |
| train_data | - | contains files copied from dataFusion/ complementary/train_data/trf???. |
| test_data | - | contains files copied from dataFusion/ complementary/test_data/tsf???. |
| validation_data | - | contains files copied from dataFusion/ complementary/validation_data/vaf???. |
| merged | merged40<br>merged50<br>merged60<br>merged70 | merged training, testing, and validation sample files (in train_data, test_data, validation_data directories) for each of the 4 stages. |
| results | results40<br>results50<br>results60<br>results70 | results for each of the 4 stages. |
| trained | trained40<br>trained50<br>trained60<br>trained70 | ANN training files for each of the 4 stages. The appropriate cross validation files are also stored in the directories associated with each stage. |
| tested | tested40<br>tested50<br>tested60<br>tested70 | ANN testing files for each of the 4 stages. |

Table C.4: Directory Structure

Note that the following files are copied to train_data/, test_data/, and validation_data/ (respectively):

- dataFusion/complementary/train_data/trf???,
- dataFusion/complementary/ test_data/tsf???, and
- dataFusion/complementary/validation_data/vaf???

These are used to create the training, testing, and validation files for the 4 stages (i.e. 40%, 50%, 60%, and 70%) of the cooperative data fusion phase.

Under the dataFusion/cooperative directory, the following procedures were implemented (in the order listed), using the specified software:

1. **Calculate Local Gains:** The script calcLocalGains.pl reads the weight files (*.w) from the fingerprint and keystroke complementary test_data/ directories. Calculates the approximate relative local gains for each input layer neuron for each participant. Sorts them in descending order but maintains correspondence between gains and neuron numbers. Writes output to fp/fpmetrics.txt and kd/kdmetrics.txt.

2. **Calculate Proportions:** The script calcProportions.pl reads from the files /fp/fpmetrics.txt and ./kd/kdmetrics.txt, and determines the ratio between keystroke and fingerprint local gains. This information is used to determine the appropriate proportions of each characteristic to use in the merging process. Writes output to proportions.txt.

3. **Fuse Data:** The script fuseMetrics.pl reads metrics files (train_data/trf???, test_data/tsf??, and validation_data/vaf??? respectively), local gain files (fp/fpmetrics.txt and kd/kdmetrics.txt), and proportions.txt. Uses this information to select appropriate features and merges them into metrics files. Writes output for different quantities of input layer neurons. The complementary data files have 76 input layer neurons per sample, so the data fusion files are written to contain 40%, 50%, 60%, and 70% of that total per sample respectively. Metrics files are written to corresponding directories under merged directory.

4. **Training The ANN:** The script execTrn.pl executes ANN training using the executable program 'mbp' in the 'mbpann' directory.

5. **Testing The ANN:** The script execTst.pl executes ANN testing using the executable program 'mbpval' in the 'mbpann' directory.

6. **Best Configuration:** The script numMLN.pl compiles information to help select the best ANN configuration. That is, the ANN configuration (more precisely the number of hidden layer neurons) that performed with best accuracy based on the outcome from testing. After running numMLN.pl, the procedure required manually selecting possible candidate configurations.

7. **Calculate Results:** The script runROC.pl calls the executable program 'calcROC'. This program reads from test_data/*.out files, then calculates and plots the ROC points on an ROC graph. The final decision threshold is determined and the FAR and FRR performance (at that threshold) calculated. Output is written to results/???.dat.

8. **Repeat** points 4 -> 7 for next percentage.

The following target files were required for training and testing the ANN:

**File df.trn:** As for the file kd.trn (described in section C.1.2), the target file for training the keystroke dynamics data contained 30 lines of 1's and 49 lines were -1's.

**File df.tst:** As for the file kd.tst (described in section C.1.2), the target file for testing the keystroke dynamics data contained 100 lines of 1's and 10,500 lines were -1's.

**File df.val:** As for the file kd.val (described in section C.1.2), the target file for cross validation (during training) of the keystroke dynamics data contained 10 lines of 1's.

As described above, programs were written for data collection, feature extraction, pre-processing, data fusion, and data analysis. The total number of lines of code developed for the project was approximately 15,000. It should be noted that the Perl scripts—which were used to perform file and data handling, and to execute other programs—were first developed for phase 1 of the experiment (keystroke dynamics). Numerous Perl scripts were re-used for phases 2 and 3 of the experiment (fingerprint recognition and feature level data fusion), though in nearly all cases major modification was required to suit the particular requirements of each phase.

# Appendix D

## D.1 Keystroke Dynamics ROC Examples

This appendix presents (in the order listed) figures illustrating the ROC graphs for participants whose performance demonstrated:

- The best classification performance.

- Good classification performance.

- Average classification performance.

- The worst classification performance.

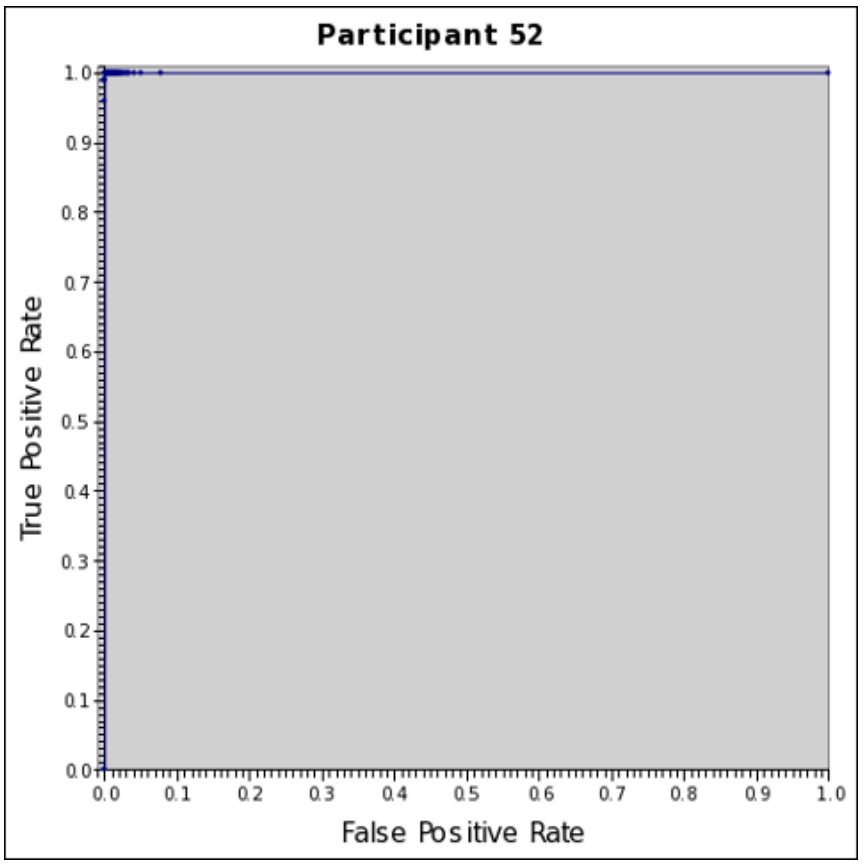


Figure D.1: Best Classification Performance – Participant 52

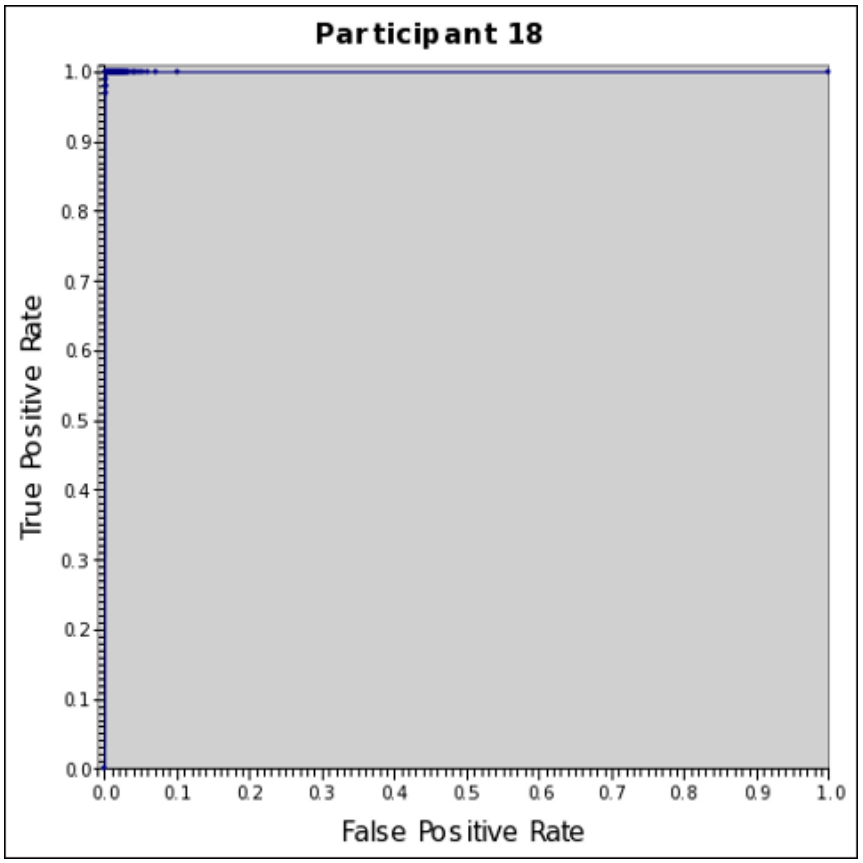


Figure D.2: Good Classification Performance – Participant 18

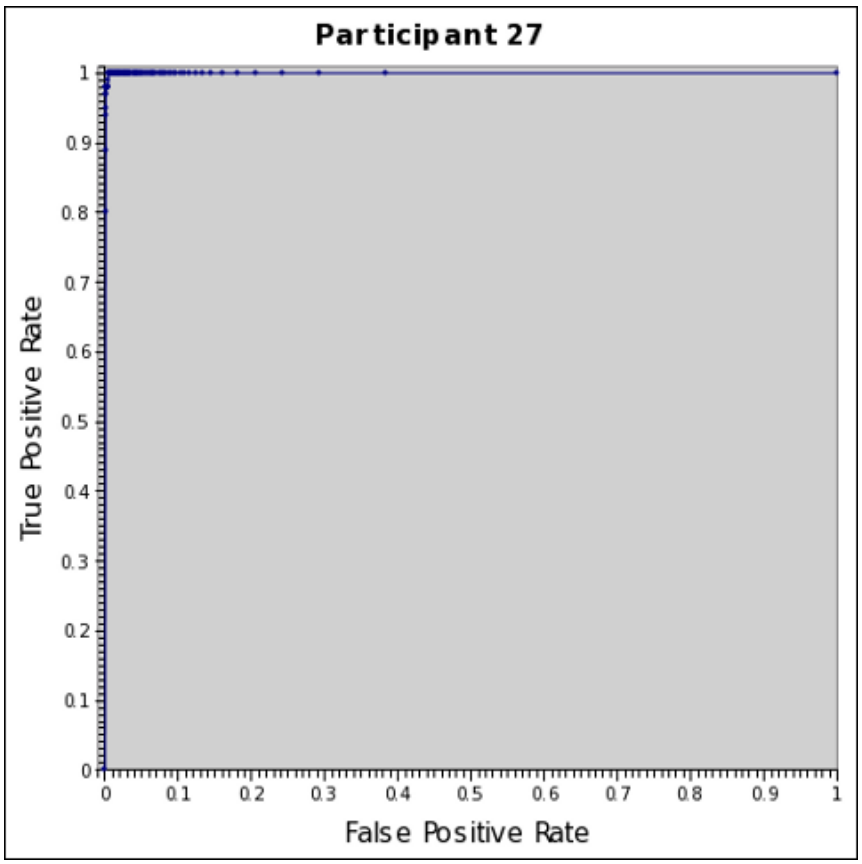


Figure D.3: Good Classification Performance – Participant 27

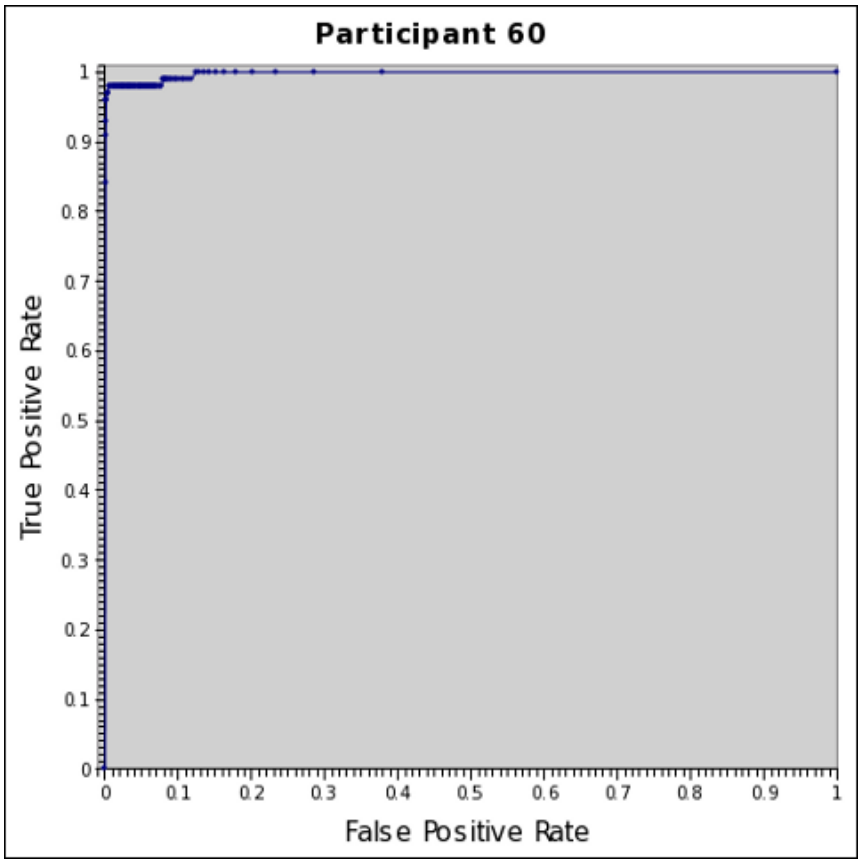


Figure D.4: Good Classification Performance – Participant 60

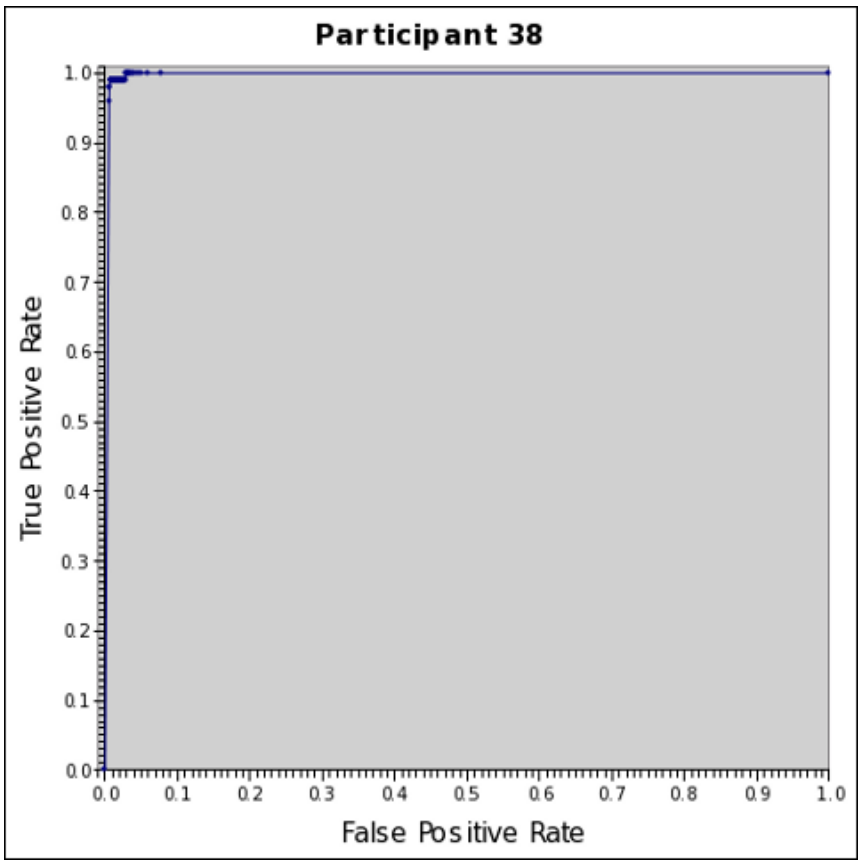


Figure D.5: Average Classification Performance – Participant 38

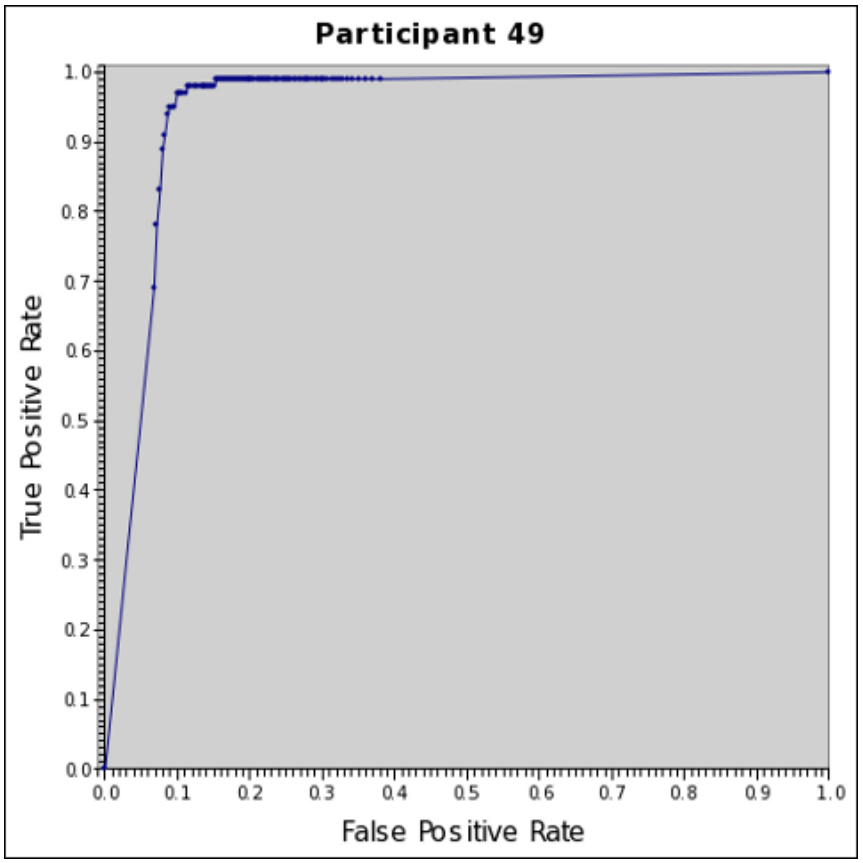


Figure D.6: Average Classification Performance – Participant 49

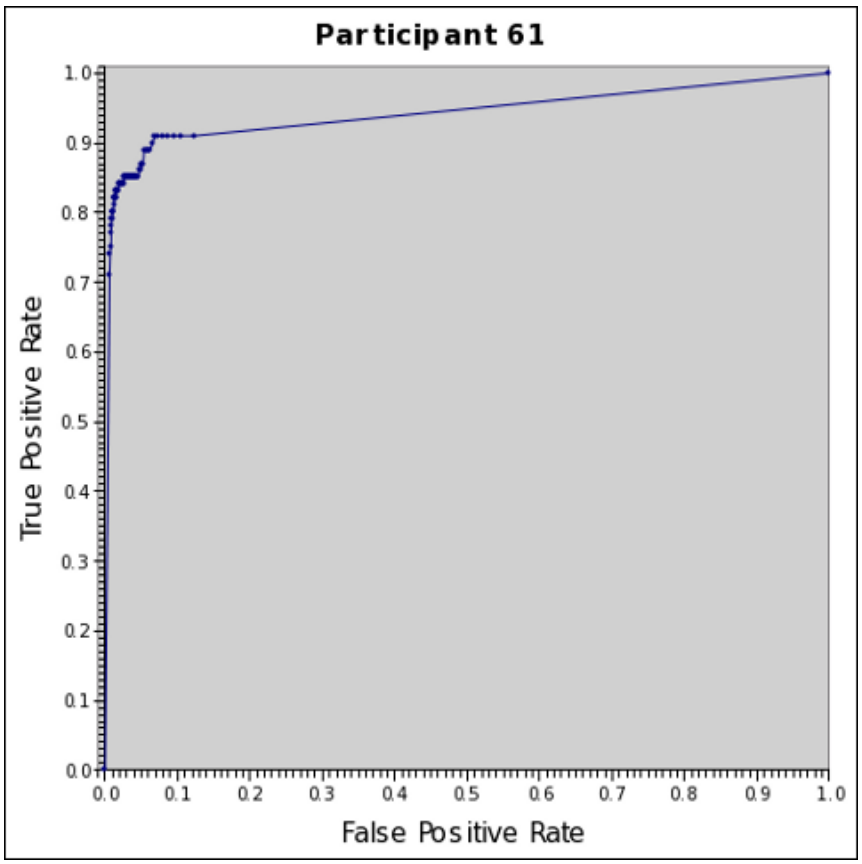


Figure D.7: Average Classification Performance – Participant 61

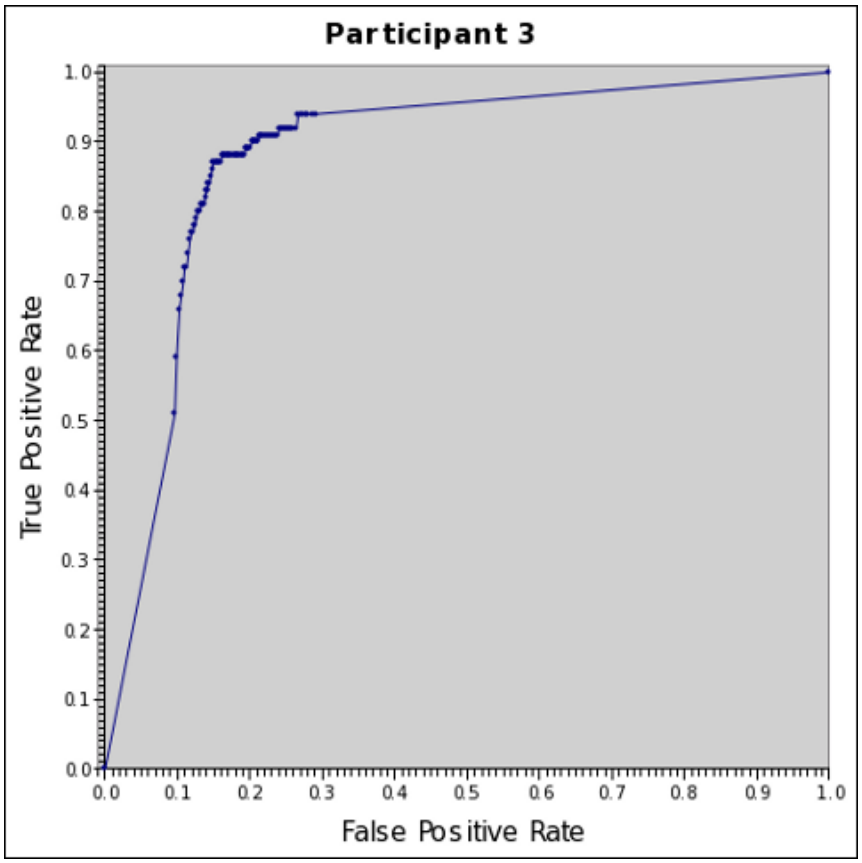


Figure D.8: Worst Classification Performance – Participant 3

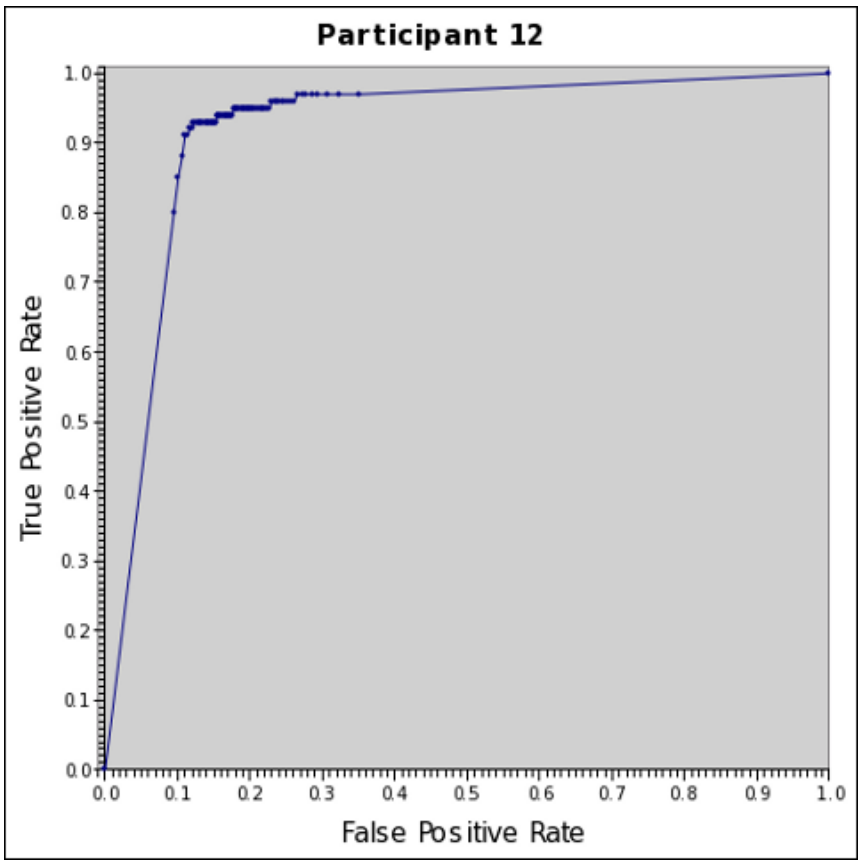


Figure D.9: Worst Classification Performance – Participant 12

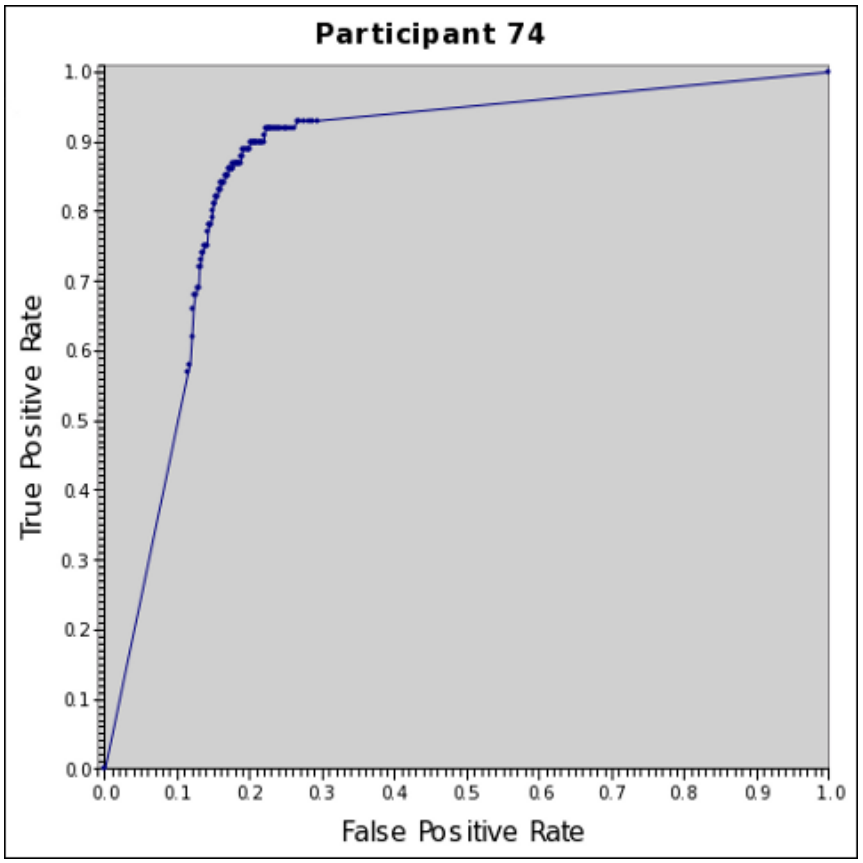


Figure D.10: Worst Classification Performance – Participant 74

# Bibliography


Abernethy, M., Rai, S. M., and Khan, M. S. (2004). User Authentication Using Keystroke Dynamics and Artificial Neural Networks. In *Preceedings of the 5th Australian Information Warfare and Security Conference. Perth Western Australia.*

Abernethy, M. S., Khan, M. S., and Rai, S. M. (2005). Feature Level Fingerprint Representation: A Pilot Study. In *The Sixth Postgraduate Electrical Engineering & Computing Symposium. Perth, Western Australia.*

Albright, L. and Malloy, T. E. (2000). Experimental Validity: Brunswick, Campbell, Cronbach and Enduring Issues. *Review of General Psychology*, 4(4):337–353.

Aleksander, I. and Morton, H. (1990). *Introduction To Neural Computing.* Chapman and Hall, first edition.

Anguita, D. (2007). Matrix Back Propagation. Available: `http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/systems/mbp/`.

Anguita, D., Parodi, G., and Zunino, R. (1993). Speed improvement of the BP on current generation workstations. In *Proceddings of the World Conference on Neural Networks ,WCNN '93, Portland USA.*, pages 165–168.

Beale, R. and Jackson, T. (1990). *Neural Computing: An Introduction.* Adam Hilger, Bristol, England., first edition.

Bergadano, F., Gunetti, D., and Picardi, C. (2002). User Authentication Through Keystroke Dynamics. *ACM Transactions on Information and Systems Security*, 5(4):367–397.

Bicz, W., Banasiak, D., Bruciak, P., Gumienny, S., Gumulinski, Z., Kosz, D., Krysiak, A., Kuszynski, W., Pluta, M., and Rabiej, G. (1999). Fingerprint

Structure Imaging Based On An Ultrasound Camera. *Instrumentation Science And Technology*, 27:295–303.

Bishop, C. M. (1995). *Neural Networks For Pattern Recognition*. Clarendon Press.

Bishop, M. (2003). *Computer Security: Art and Science*. Addison-Wesley.

Bleha, S. and Obaidat, M. S. (1991). Dimensionality Reduction and Feature Extraction in Indentifying Computer Users. *IEEE Transactions on Systems, Man and Cybernetics*, 21(2).

Bowman, C. L. and Steinberg, A. N. (2001). *Handbook Of Multisensor Data Fusion*, chapter 16. Systems Engineering and Implementation, pages 1 – 39. CRC Press.

Boyd, J. and Little, J. (2005). Biometric Gait Recognition. *Advanced Studies in Biometrics*, pages 19–42.

Bradley, A. P. (1997). The Use Of The Area Under The ROC Curve In The Evaluation Of Machine Learning Algorithms. *Pattern Recognition*, 30(7):1145–1159.

Brase, C. H. and Brase, C. P. (2004). *Understanding Basic Statistics*. Houghton Mifflin Company, third edition.

Brooks, R. R. and Iyengar, S. S. (1998). *Multi-Sensor Fusion: Fundamentals and Applications with Software*. Prentice Hall PTR.

Brown, M. and Rogers, S. J. (1993). User Identification via Keystroke Characteristics of Typed Names Using Neural Networks. *International Journal of Man-Machine Studies*, 39(6):999–1014.

Cappelli, R., Lumini, A., Maio, D., and Maltoni, D. (1999). Fingerprint Classification By Directional Image Partitioning. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 21(5):402–421.

Card, S. K., Moran, T. P., and Newell, A. (1980). The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM*, 23(7):396–410.

CERT/CC (2004). Reported Incidents of Security Breaches. Carnegie Mellon University Software Engineering Institute Computer Emergency Response Team/Coordination Center. Available: `http://www.cert.org/stats/cert_stats.html`.

Chang, J.-H. and Fan, K.-C. (2002). A New Model For Fingerprint Classification By Ridge Distribution Sequences. *Pattern Recognition*, 35(6):1209–1223.

Chatzis, V., Bors, A. G., and Pitas, I. (1999). Multimodal Decision Level Fusion For Person Authentication. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 29(6):674–680.

Cho, S., Chigeun, H., Han, D. H., and Kim, H.-I. (2000). Web-Based Keystroke Dynamics Identity Verification Using Neural Networks. *Journal of Organizational Computing and Electronic Commerce*, 10(4):295–307.

Connie, T., Teoh, A., Goh, M., and Ngo, D. (2003). Palmprint Recognition with PCA and ICA. In *Proceedings of the Image and Vision Computing Conference New Zealand 2003. Massie University, Palmerston North, New Zealand.*

Cooper, W. E. (1983). *Cognitive Aspects of Skilled Typewriting*, pages 29–32. Springer-Verlag.

CSI (2009). CSI Computer Crime And Security Survey. Computer Security Institute Online. Available: `http://gocsi.com/survey`.

CSO (2004). E-Crime Watch Survey. CSO Magazine and Online. Available: `http://www.cert.org/archive/pdf/2004eCrimeWatchSummary.pdf`. In cooperation with U.S. Secret Service and CERT/CC.

CSO (2010). E-Crime Watch Survey. CSO Magazine and Online. Available: `http://www.cert.org/archive/pdf/ecrimesummary10.pdf`. In cooperation with U.S. Secret Service, CERT/CC, and Deloitte.

Cummins, H. (1941). Ancient Finger Prints In Clay. *Journal of Criminal Law and Criminology*, 32(4):468–481.

Cummins, H. and Kennedy, R. W. (1940). Purkinje's Observations (1823) On Finger Prints And Other Skin Features. *Journal of Criminal Law and Criminology*, 31(3):343–356.

Dash, M. and Liu, H. (1997). Feature Selection for Classification. *Intelligent Data Analysis*, 1(3):131–156.

Daugman, J. G. (2004). How Iris Recognition Works. *IEEE Transacations on*

*Circuits and Systems for Video Technology*, 14(1):1–10.

De Carlo, L. T. (1997). On The Meaning And Use Of Kurtosis. *Psychological Methods*, 2:292–307.

Delac, K. and Grgic, M. (2004). A SURVEY OF BIOMETRIC RECOGNITION METHODS. In *46th International Synposium Electronics in Marine. ELMAR-2004.*, Zadar, Croatia.

Digital Persona (2004). Guide to Fingerprint Identification (White Paper). http://www.digitalpersona.com/docrequest/reqform?doc=14.

Doric, D., Nikolic-Doric, E., Jevremovic, V., and Malisic, J. (2007). On Measuring Skewness And Kurtosis. *Quality And Quantity*. Available: `http://www.springerlink.com/content/xm57j151p3957317/`.

Emory, C. W. and Cooper, D. R. (1991). *Business Research Methods*, chapter 13. Experimentation, pages 415–445. Homewood, Irwin.

Faulds, H. (1880). On the Skin-furrows of the Hand. *Nature*, (605):1.

Faulds, H. (1905). *Guide To Finger-Print Identification*. Wood, Mitchell & Co. Ltd, Park Street, London.

Faundez-Zanuy, M. (2009). *Multimodal Signals: Cognative And Algorithmic Issues*, volume 5398/2009 of *Lecture Notes In Computer Science*, chapter Data Fusion At Different Levels, pages 94–103. Springer-Verlag Berlin / Heidelberg.

Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recognition Letters*, (27):861–874.

Fisher, R. A. (1930). The Moments of the Distribution for Normal Samples of Measures of Departure from Normality. In *Proceedings of the Royal Society of London*, volume 103 of *A*, pages 16–28.

Flach, P. A. (2004). The Many Faces Of ROC Analysis In Machine Learning. Unpublished. Available: `http://www.cs.bris.ac.uk/~flach/ICML04tutorial/`. Tutorial at International Conference on Machine Learning (ICML-2004).

Fong, L. and Seng, W. (2009). A Comparison Study on Hand Recognition Approaches. In *2009 International Conference of Soft Computing and Pattern*

*Recognition*.

Fosdick, R. B. (1915). The Passing Of The Bertillon System Of Identification. *Journal of the American Institute of Criminal Law and Criminology*, 6(3):363–369.

Friedman, M. and Kandel, A. (1999). *Introduction To Pattern Recognition: Statistical, Structural, Neural And Fuzzy Logic Approaches*, volume 32 of *Machine Learning And Artificial Intelligence*. World Scientific Publishing Company, bunke, h. and wang, p.s.p. edition.

Frischholz, R. W. and Dieckmann, U. (2000). BioID: A Multimodal Biometric Identification System. *IEEE Computer*, 33(2):64–68.

Gaines, R. S., Lisowski, W., Press, S. J., and Shapiro, N. (1980). Authentication by Keystroke Timing: Some Preliminary Results. Technical report, Rand Corporation. Report number: R-2526-NSF.

Galton, F. (1892). *Finger Prints*. Macmillan and Company.

Garfinkel, S., Spafford, G., and Schwartz, A. (2003). *Practical Unix & Internet Security*. O'Reilly & Associates, third edition.

Greiner, M., Pfeiffer, D., and Smith, R. D. (2000). Principles and Practical Application of the Receiver-Operating Characterstic Analysis for Diagnostic Tests. *Preventive Veterinary Medicine*, 45:23–41.

Grew, N. (1684). The Description and Use of the Pores in the Skin of the Hands and Feet. *Philosophical Transactions (1683-1775)*, 14:566–567.

Hall, D. L. and Llinas, J. (1997). An Introduction to Multisensor Data Fusion. In *Proceedings of the IEEE*, volume 85, pages 6–23.

Hall, D. L. and Llinas, J. (2001). *Handbook Of Multisensor Data Fusion*, chapter 1. Introduction to Multisensor Data Fusion, pages 1–10. CRC Press.

Haykin, S. (1999). *Artificial Neural Networks: A Comprehensive Foundation*. Prentice Hall International, second edition.

He, Y., Tian, J., Luo, X., and Zhang, T. (2003). Image Enhancement And Minutiae Matching In Fingerprint Verification. *Pattern Recognition Letters*, 24:1349 – 1360.

Heffner, C. L. (2004). Research Methods. AllPysch Online. Available: `http://allpsych.com/researchmethods/experimentalvalidity.html`. Chapter 7: Variables, Validity, and Reliability. Section 4: Experimental Validity.

Henry, E. R. (1900). *Classification And Uses of Finger Prints*. George Routledge and Sons Limited, London.

Herschel, W. J. (1916). *The Origin Of Finger-Printing*. Oxford University Press, London.

Hong, C. S. (2009). Optimal Threshold From ROC and CAP Curves. *Communications in Statistics-Simulation and Computation*, 38:2060–2072.

Hu, J., Gingrich, D., and Sentosa, A. (2008). A k-Nearest Neighbor Approach for User Authentication Through Biometrc Keystroke Dynamics. In *Proceedings of the IEEE International Conference on Communications*, pages 1556–1560.

Hughes, P. A. and Green, D. P. (1991). The Use Of Neural Networks For Fingerprint Classification. In *Preceedings of the Second International Conference on Artificial Neural Networks*, Bournemouth, UK.

Impedovo, S. and Pirlo, G. (2007). Verification of Handwritten Signatures: an Overview. In *IEEE 14th International Conference on Image Analysis and Processing (ICIAP 2007)*, Bari, Italy.

Inbau, F. E. (1934). Scientific Evidence In Criminal Cases. III. Finger-Prints and Palm-Prints. *Journal of Criminal Law and Criminology*, 25(3):500–516.

Indovina, M., Uludag, U., Snelick, R., Mink, A., and Jain, A. (2003). Multimodal Biometric Authentication Methods: A COTS Approach. In *Proceedings of the Workshop On Multimodal User Authentication (MMUA)*, pages 99–106, Santa Barbara, California.

International Biometric Group (2006). Is DNA a Biometric? Online. Available: `http://www.biometricgroup.com/reports/public/reports/dna.html`.

Jain, A., Hong, L., and Bolle, R. (1997). On-Line Fingerprint Verification. *IEEE Transactions On Pattern Recognition And Machine Learning*, 19(4):302–314.

Jain, A. K., Hong, L., and Kulkarni, Y. (1999a). Multimodal Biometric System Using Fingerprint, Face, And Speech. In *Proceedings of the Second International*

*Conference on Audio- and Video-based Person Authentication (AVBPA)*, pages 182 – 187, Washington D. C., U. S. A.

Jain, A. K., Prabhakar, S., and Hong, L. (1999b). A Multichannel Approach To Fingerprint Classification. *IEEE Transactions On Pattern Analysis And Machine Learning*, 21(4):348 – 359.

Jain, A. K., Ross, A., and Prabhakar, S. (2004). An Introduction to Biometric Recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20.

Jiang, C. H., Shieh, S., and Liu, J. C. (2007). Keystroke Statistical Learning Model for Web Authentication. In *Proceedings of the 2nd ACM Symposium on Information, Computer, and Communications Security*, pages 359–361. ACM.

Jiang, X., Yau, W., and Ser, W. (2001). Detecting The Fingerprint Minutiae By Apadtive Tracing The Gray-Level Ridge. *Pattern Recognition*, 34:999 – 1013.

Jiang, X. and Yau, W.-Y. (2000). Fingerprint Minutiae Matching Based on the Local And Global Structures. In *Proceedings of the 15th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 1038 – 1041.

Jie, Y., Yi fang, Y., Renjie, Z., and Qifa, S. (2006). Fingerprint Minutiae Matching Algorithm For Real Time System. *Pattern Recognition*, 39(1):143 – 146.

John, G. H., Kohavi, R., and Pfleger, K. (1994). Irrelevant Features And The Subset Selection Problem. In Cohen, W. and Hirsh, H., editors, *Machine Learning: Preceedings of the Eleventh International Conference*, pages 121–129. Morgan Kaufmann Publishers, San Francisco, California, U.S.A.

Joyce, R. and Gupta, G. (1990). Identity Authentication Based on Keystroke Lantencies. *Communications of the ACM*, 33(2):168–176.

Karu, K. and Jain, A. K. (1996). Fingerprint Classification. *Pattern Recognition*, 29(3):389–404.

Kasabov, N. K. (1996). *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*, chapter 5. Neural Networks For Knowledge Engineering And Problem Solving, pages 331–419. MIT Press, Cambridge, MA, USA.

Korotkaya, Z. (2003). Biometric Person Authentication: Odor. In *Presented at Ad-*

*vanced Topics In Information Processing. Lappeenranta University of Technology, Finland.*

Kristensen, T., Borthen, J., and Fyllingsnes, K. (2007). Comparison Of Neural Networks Based Fingerprint Classification Techniques. In *Proceddings of International Joint Conference On Neural Networks*, Orlando Florida, USA. IEEE.

Kumar, R. and Deva Vikram, B. R. (2010). Fingerprint Matching Using Multi-Dimensional ANN. *Engineering Applications of Artificial Intelligence*, 23:222 – 228.

Lee, D., Choi, K., and Kim, J. (2002). A Robust Fingerprint Matching Algorithm Using Local Alignment. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR)*, volume 3, pages 803 – 806, Quebec, Canada.

Leggett, J. and Williams, G. (1988). Verifying Identity via Keystroke Characteristics. *International Journal of Man-Machine Studies.*, 28(1):67–76.

Leggett, J., Williams, G., Usnick, M., and Longnecker, M. (1991). Dynamic Identity Verification via Keystroke Characteristics. *International Journal of Man-Machine Studies.*, 35(6):859–870.

Leung, W. F., Leung, S. H., Lau, W. H., and Luk, A. (1991). Fingerprint Recognition Using Neural Networks. In *Proceedings of the Workshop on Neural Network for Signal Processing*, pages 226 – 235.

Li, W., Zhang, L., Zhang, D., Lu, G., and Yan, J. (2010). Efficient Joint 2D and 3D Palmprint Matching with Alignment Refinement. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR10)*, pages 795–801, San Francisco, California U.S.A.

Lippmann, R. (1987). An Introduction To Computing With Neural Networks. *IEEE ASSP Magazine*, pages 4–22.

Lippmann, R. P. (1989). Pattern Classification Using Neural Networks. *IEEE Communications Magazine*, 27(11):47–50, 59–64.

Liu, S. and Silverman, M. (2001). A Practical Guide to Biometric Security Technology. *IT Pro (IEEE)*, 1(1):27–32.

Luo, X., Tian, J., and Wu, Y. (2000). A Minutiae Matching Algorithm in Fingerprint

Verification. In *Proceedings of the 15th International Conference on Pattern Recognition (ICPR)*, volume 4, pages 833 – 836.

Maher, D., Napier, R., Wagner, M., Laverty, W., Henderson, R. D., and Hiron, M. (1995). Optimal Digraph-Latency Based Biometric Typist Verification Systems; Inter And Intra Typist Differences In Digraph Latency Combinations. *International Journal of Human Computer Studies.*, 43(4):579–592.

Maio, D. and Maltoni, D. (1997). Direct Gray-Scale Minutiae Detection in Fingerprints. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 19(1):27 – 40.

Maltoni, D., Maio, D., Jain, A. K., and Prabhakar, S. (2003). *Handbook of Fingerprint Recognition.* Springer.

Markowitz, J. A. (2000). Voice Biometics. *Communications of the ACM*, 43(9):66–73.

Marques de Sa, J. P. (2001). *Pattern Recognition: Concepts, Methods and Applications.* Springer-Verlag, Germany., first edition.

Masters, T. (1993). *Practical Neural Network Recipes in C++.* Academic Press Inc., San Diego, California., first edition.

Matyas Jr, V. and Riha, Z. (2000). Biometric Authentication Systems. Technical report, Ecom-Monitor. Available: `http://www.ecom-monitor.com/papers/biometricsTR2000.pdf`.

Mitnick, K. D. and Simon, W. L. (2002). *The Art of Deception: Controlling the Human Element of Security.* Robert Ipsen.

Monrose, F., Reiter, K. M., and Wetzel, S. (2002). Password hardening based on keystroke dynamics. *International Journal of Information Security*, 1(2):69–83.

Monrose, F. and Rubin, A. D. (1997). Authentication via Keystroke Dynamics. In *Proceedings of the 4th ACM Conference on Computer and Communication Security.*

Monrose, F. and Rubin, A. D. (2000). Keystroke Dynamics as a Biometric for Authentication. *Future Generation Computer Systems.*, 16(4):351–359.

Nandakumar, K. (2008). *Multibiometric Systems: Fusion Strategies And Template*

*Security*. Doctor of philosophy, Michigan State Univeristy, Department of Computer Science and Engineering.

National Centre for State Courts (2006). Biometric Characteristics. Online. Available:
`http://ctl.ncsc.dni.us/biomet\%20web/BMIndex.html`.

National Institute of Standards and Technology (2010a). DNA Biometrics. Online. Available:
`http://www.nist.gov/mml/biochemical/genetics/dna_biometrics.cfm`.

National Institute of Standards and Technology (2010b). NIST Biometric Scores Set. Online. Available:
`http:///www.itl.nist.gov/iad/894.03/biometricscores`.

Nua (2002). Number of Internet Users. Computerscope Ltd. Available:
`http://www.nua.ie/surveys/how_many_online/world.html`.

Obaidat, M. S. and Sadoun, B. (1997). Verification of Computer Users Using Keystroke Dynamics. *IEEE Transactions On Systems, Man, And Cybernetics-Part B: Cybernetics*, 27(2):261–269.

O'Gorman, L. (1998). An Overview Of Fingerprint Verification Technologies. *Information Security Technical Report*, 3(1):21 – 32. Published: Elsevier Science Ltd.

Osadciw, L., Varshney, P., and Veeramachaneni, K. (2003). *Multisensor Surveillance Systems: The Fusion Perspective*, chapter 15. Optimum Fusion Rules For Multimodal Biometric Systems. Kluwer Academic Publishers.

Ozkaya, N., Sagiroglu, S., and Wani, A. (2006). An Intelligent Automatic Fingerprint Recognition System Design. In *Proceedings of the 5th International Conference on Machine Learning and Applications (ICMLA)*, pages 231 – 238.

Peacock, A. (2000). Learning User Keystroke Latency Patterns. Available:
`http://pel.cs.byu.edu/~alen/personal/CourseWork/cs572/KeystrokePaper`.

Pearson, K. (1905). Das Fehlergesetz und seine Verallgemeinerungen durch Fechner und Pearson. A Rejoinder. *Biometrika*, 4:169–212.

Poh, N. and Kittler, J. (2008). Multimodal Information Fusion.

Polson, C. J. (1951). Finger Prints And Finger Printing: A Historical Study. *Journal of Criminal Law and Criminology*, 41(5):690–704.

Provost, F., Fawcett, T., and Kohavi, R. (1998). The Case Against Accuracy Estimation for Comparing Induction Algorithms. In Shavlik, J., editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 445–453. Morgan Kaufmann.

Qi, J. and Wang, Y. (2005). A Robust Fingerprint Matching Method. *Pattern Recognition*, 38:1665 – 1671.

Ratha, N. K., Karu, K., Chen, S., and Jain, A. K. (1996). A Real-Time Matching System For Large Fingerprint Databases. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 18(8):799–813.

Rattani, A., Kisku, D. R., Bicego, M., and Tistarelli, M. (2007). Feature Level Fusion Of Face And Fingerprint Biometrics. In *Proceeding of First IEEE International Conference on Biometrics: Theory, Applications, and Systems (BTAS)*, pages 1–6.

Reeves, C. (2003). *Handbook Of Metaheuristics*, chapter Chapter 3: Genetic algorithms, pages 55–82. Springer.

Revett, K., Gorunescu, F., Gorunescu, M., Ene, M., Magahaes, S., and Santos, H. (2007). A Machine Learning Approach to Keystroke Dynamics Based User Authentication. *International Journal of Electronic Security and Digital Forensics*, 1(1):55–70.

Ross, A. and Govindarajan, R. (2005). Feature Level Fusion Using Hand And Face Biometrics. In *Proceedings of the SPIE Conference On Biometric Technology For Human Identification II*, pages 196–204, Orlando, Florida. USA.

Ross, A. and Jain, A. K. (2004). Mulitmodal Biometrics: An Overview. In *Proceedings of the 12th European Signal Processing Conference (EUSIPCO)*, pages 1221 – 1224, Vienna, Austria.

Ross, A., Jain, A. K., and Qian, J.-Z. (2001). Information Fusion In Biometrics. In *Proceedings of 3rd International Conference on Audio- and Video-Based Person*

*Authenticiation (AVBPA)*, pages 354–539, Sweden.

Rychetsky, M. (2001). *Algorithms and Architectures for Machine Learning based on Regularized Neural Networks and Support Vector Approaches*. Shaker Verlag.

Schneider, J. K. and Wobschall, D. C. (1991). Live Scan Fingerprint Imagery Using High Resolution C-SCAN Ultrasonography. In *Proceedings of the 25th International Carnahan Conference On Security Technology*, pages 88–95.

Schneier, B. (2000). *Secrets & Lies: Digital Security in a Networked World*. John Wiley and Sons, Inc.

Schuschel, D. and Hsu, C.-N. (1998). A Weight Analysis-Based Wrapper Approach to Neural Nets Feature Subset Selection. In *Proceedings of the 10th IEEE Conference on Tools With Artificial Intelligence*, pages 89–96, Taipei, Taiwan.

Son, B. and Lee, Y. (2005). Biometric Authentication System Using Reduced Joint Feature Vector of Iris And Face. In Kanade, T., Jain, A., and Ratha, N. K., editors, *Audio-and Video-Based Biometric Person Authentication*, LNCS 3546, pages 513–522. Springer-Verlag.

Specter, M. (2002). Do Fingerprints Lie? The New Yorker.

Swets, J. A. (1988). Measuring the Accuracy of Diagnostic Systems. *Science*, 240(4857):1285–1293.

Tanenbaum, A. S. (1996). *Computer Networks*. Prentice-Hall Inc., third edition.

The Northwestern University Medical School (2007a). Descriptive Statistics. PROPHET: StatGuide. Available: `http://www.basic.northwestern.edu/statguidefiles/desc.html`.

The Northwestern University Medical School (2007b). Examining Normality Test Results. PROPHET: StatGuide. Available: `http://www.basic.northwestern.edu/statguidefiles/n-dist_exam_res.html`.

The XM2VTS Database (2010). The Extended Multi Modal Verification for Teleservices and Security Database (XM2VTSDB). Available: `http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/`.

Theodoridis, S. and Koutroumbas, K. (2006). *Pattern Recognition*. Elsevier, third

edition.

Tong, X., Huang, J., Tang, X., and Shi, D. (2005). Fingerprint Minutiae Matching Using The Adjacent Feature Vector. *Pattern Recognition Letters*, 26:1337 – 1345.

Trauring, M. (1963). Automatic Comparison of Finger-Ridge Patterns. *Nature*, 197:938–940.

Trozzi, T. A., Schwartz, R. L., and Hollars, M. L. (2000). Processing Guide For Developing Latent Prints. Technical report, Federal Bureau of Investigation.

Tsekouras, G. E. (2005). A Fuzzy Vector Quantization Approach to Image Compression. *Applied Mathematics and Computation*, 167:539–560.

Umphress, D. and Williams, G. (1985). Identity Verification Through Keyboard Characteristics. *International Journal of Man-Machine Studies.*, 23(3):263–273.

Van Wamelen, P. B., Li, Z., and Iyengar, S. S. (2004). A Fast Expected Time Algorithm For The 2-D Point Pattern Matching Problem. *Pattern Recognition*, 37:1699–1711.

Varshney, P. K. (1997). Multisensor Data Fusion. *Electronics and Communication Engineering Journal*, 9:245–253.

Wang, X. (2002). *Feature Extraction and Dimensionality Reduction in Pattern Recognition and their Application in Speech Recognition.* Doctor of philosophy, School of Microelectronical Engineering Faculty of Engineering and Information Technology, Griffith University.

Wang, Y., Tan, T., and Jain, A. K. (2003). Combining Face And Iris Biometrics For Identity Verification. In *Proceedings of the Fourth International Conference on Audio- and Video-based Person Authentication (AVBPA)*, pages 805 – 813.

Wayman, J., Jain, A., Maltoni, D., and Maio, D. (2005). *Biometric Systems: Technology, Design and Performance Evaluation*, chapter 1: An Introduction to Biometric Authentications Systems, pages 1–20. Springer-Verlag, first edition.

Websters (2010a). Websters Online Dictionary: Rosetta Edition. Online. Available: `http://www.websters-online-dictionary.org/definitions/biometrics`.

Websters (2010b). Websters Online Dictionary: Rosetta Edition. Online. Available: `http://www.websters-online-dictionary.org/definitions/dna`.

Websters (2010c). Websters Online Dictionary: Rosetta Edition. Online. Available: http://www.websters-online-dictionary.org/definitions/retina.

Wilson, C., Candela, G., and Watson, C. (1993). Neural Network Fingerprint Classification. *Journal of Artificial Neural Networks*, 1(2):203–228.

Wong, K. W., Fung, C. C., Gedeon, T. D., and Ong, Y. S. (2005). *Neural Network Applications In Information Technology And Web Publishing*, chapter 22. Generalization Of Neural Networks For Intelligent Data Analysis, pages 304–317. Borneo Publishing, Sarawak, Malaysia.

Wuenschk, K. L. (2007). Descriptive Statistics. Online. Available: http://core.ecu.edu/psyc/wuenschk/docs30/descript.doc.

Xia, X. and O'Gorman, L. (2003). Innovations In Fingerprint Capture Devices. *Pattern Recognition*, 36:361–369.

Yager, N. and Amin, A. (2004a). Fingerprint Classification: A Review. *Pattern Analysis Applications*, 7(1):77–93.

Yager, N. and Amin, A. (2004b). Fingerprint Verification Based On Minutiae Features: A Review. *Pattern Analysis Applications*, 7(1):94–113.

Yao, Y. F., Jing, X, Y., and Wong, H. S. (2007). Face And Palmprint Feature Level Fusion For Single Sample Biometrics Recognition. *Neurocomputing*, 70(7-9):1852–1586.

Yu, E. and Cho, S. (2004). Keystroke Dynamics Identity Verification: Its Problems and Practical Solutions. *Computers & Security*, 23(5):428–440.

Zhang, J., Luo, X., Akkaldevi, S., and Ziegelmeyer, J. (2009). Improving Multiple-Password Recall: An Empirical Study. *European Journal Of Information Systems*, pages 1–12.

Zhang, Q., Huang, K., and Yan, H. (2001). Fingerprint Classification Based On The Detection And Analysis Of Singularities And Pseudoridges. In *Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing, Sydney, Australia.* Available: http://citeseer.ist.psu.edu/535060.html.

Zikmund, W. G. (1997). *Business Research Methods.* Forth Worth: The Dryden

Press., fifth edition.