FINGERPRINTS: ORIENTATION FREE MINUTIAE EXTRACTION AND USING DISTANCES BETWEEN MINUTIAE FOR IDENTIFICATION AND VERIFICATION

David Adam Braude

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 2010

Declaration

I declare that this dissertation is my own, unaided work, other than where specifically acknowledged. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Signed this _____ day of _____ 2010

_____

David Adam Braude

Abstract

Fingerprint recognition has become a standard in both access control and forensics. This is because fingerprints are unique to an individual. While there are many ways in which a fingerprint can be recognised one of the most common is to look at the endings and splitting of the ridges. These are called minutiae. This research undertakes to improve upon existing methods used in all parts of a minutiae based fingerprint verification system. This study aims to find a new way to extract these minutiae. It also seeks to use them in a novel way to identify an individual and verify that two fingerprints come from the same person. This was done in an effort to improve speed in fingerprint recognition systems by reducing the processing overhead.

There is one key difference between the new extraction algorithm and standard methods. In the new method for extraction the orientation of the ridges is not found. This was done to speed the process of extraction. To verify that two fingerprints are the same the distances between minutiae was considered to be binary attributes of a graph. This turned the verification into a graph-matching problem. The distances between the minutiae were split into a histogram and the values in the bins were the inputs to a Multilayer Perceptron (MLP). This MLP was used to group fingerprints into classes to speed the identification process. The MLP was trained using Particle Swarm Optimisation.

The new extraction algorithm finds minutiae very quickly. However, it finds many false minutiae. The graph-matching approach is unable to distinguish between a match and a non-match and is very slow to run. This is also true for the case when the unary attributes are included. These attributes are the type of minutiae and angle of the ridge at the minutiae point. The classifier runs quickly, but places all the fingerprints in the same class. Thus it will not improve identification time.

It is possible that a filtering system could be developed to combat the amount of false minutiae. This would make the new algorithm viable. Care must be taken to

avoid increasing the runtime to beyond industry standard. The amount of spurious minutiae could be affecting the performance of the graph matching and classification. Alternatively it could be due to different minutiae being extracted between scans. This is due to different parts of the finger are observed with each scan. The cause will need to be investigated.

While positive results were not obtained, this research forms the basis of future investigation. Two questions will now need to be answered. The first is can a filter be developed to remove spurious minutiae from the extraction process? The second, are the spurious minutiae the cause of the problem or will only using the distances be sufficient? If the latter is the case, then finding the angle of the ridge is no longer necessary and the minutiae extraction process can be speeded up by using the new algorithm.

To my mother, her memory is still strong

Acknowledgements

I would like to recognise and thank my supervisor, Prof. Anton van Wyk who gave me guidance and advise from initial conception to final presentation. I would also like to extend thanks to Avri Spilka for her efforts in proof reading the dissertation. To Natalie Myburgh for editing the final draft. And to Fulufhelo Nelwamondo and the Information Security research group at the CSIR for additional guidance and financial support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The use of biometrics dates back to ancient human civilisation. Face, voice, gait and body size are some examples of how our ancestors recognised familiar people. However, the earliest recorded examples of biometrics as a form of identification are from Babylon, where merchants would stamp their fingerprints into their clay tablets for business transactions [1]. Thumbprints have also been found on ancient Chinese seals. In the 1300's Joâo de Barros discovered that Chinese merchants took palm and footprints to distinguish between children [1, 2].

Other early examples of biometrics stem from the Bible, for example in Genesis Jacob fooled his father, Isaac, into thinking he was Esau. He did this by putting on a goat's skin so he would feel as hairy as Esau. Isaac first thought it was Jacob based on his voice but was deceived by touch. This is an example of biometric spoofing. Also, in The Book of Judges the men of Gilead found Ephraimites based on accent. Using only this evidence they killed 42 000 people [3].

In 1893 Alphonse Bertillon, a Parisian police desk clerk, developed a system of biometric identification for criminals. This was based on measurements of the body, such as the length of fingers. The system was known as Bertillonage and saw widespread use until it was found that many people shared the same measurements [3].

Sir Edward Henry, a General Inspector of the Bengal police in 1896, began the practice of using fingerprints for criminal identification. He created a method that allowed for quick searching of records by sorting. He used classes that were first developed by Sir Francis Galton in 1892. This was first used in London but rapidly spread throughout the world [4].

The first conviction based on a fingerprint was made in June, 1892. Francisca Rojas

was found guilty of murdering her two sons based on a bloody fingerprint taken at the crime scene. This was pioneered by Juan Vucetich with the help of Eduardo Alvarez. Vucetich was also responsible for the creation of the first fingerprint bureau in March 1892 [5]. Henry's bureau was founded in 1901 [4].

With the advent of modern computing power and developments in the sensors, fingerprints now have another major application. The use of fingerprints in access control is advantageous over other methods such as PIN numbers as a fingerprint cannot be stolen. Some scanners can even ensure that the fingerprint is from a living, uncovered finger. Though other biometrics such as face recognition and iris scans have started gaining popularity, fingerprints remain the most pervasive form. Access control includes the replacement of passwords on personal computers and physical access control.

There are, however, two major controversies over the use of biometrics in general, and fingerprints in particular. The first is that the collection of biometrics infringes on the privacy of the individuals concerned. Despite this, many work places will use fingerprint scanners as a time keeping system. Other uses have included schools where fingerprints are used instead of library cards. This caused an outcry because fingerprints were taken without the parents' knowledge or consent [6].

The other major criticism concerns the reliability of the systems used. There have been cases of false positive identification in criminal proceedings such as Brandon Mayfield. He was falsely identified by the FBI as an accomplice to the Madrid Bombing and was later exonerated by the Spanish police [7]. In access control there is always a False Acceptance Ratio (FAR), even if it is very low. However, the lower the FAR, the higher the False Rejection Ratio (FRR). Despite these problems in South African precedent, it takes a mere six minutiae to make a conviction. Minutiae are explained below.

Every fingerprint is thought to be unique. This differentiation is a vital characteristic for forensic science and access control. There is, however, no intuitive way to sort fingerprint images. As a result it is very difficult to identify a person in a database based on fingerprint alone. Further, to match and verify two fingerprints are from the same finger presents another challenge. Fingerprint recognition is entirely reliant on the methods for solving the problems of identification and verification.

Formally, the difference between the problems is as follows: Identification is the problem of determining if a fingerprint is in a database of known fingerprints. If

they are the problem becomes uniquely determining who the fingerprint is from. Verification is the process by which two fingerprints are compared, to determine if they are from the same finger. Obviously a good verification system is needed for identification. In some cases an identification system is not needed, for example when you enter a user ID before scanning, however, verification is always required [8].

For both the problems of fingerprint sorting and fingerprint matching many approaches have been taken. Classification helps to narrow a database search by reducing the amount of fingerprints that will need to be compared. A proposed tool to help with this is looking at the general shape that the ridges of the fingerprint form, such as left and right loops [9]. Another approach to creating groups uses features of the ridge structure of the fingerprint called the cores and deltas. Cores (or nuclei) and deltas are collectively known as singular points [8, 10]. These are illustrated in Figure 1.1. The definitions according to the ISO standard are:

> DEFINITION 1: A **core** is the topmost point on the innermost recurving ridge line of a fingerprint. Generally, the core is placed upon or within the innermost recurve of a loop. [11]

> DEFINITION 2: A **delta** is that point on a ridge at or nearest to the point of divergence of two type lines, and located at or directly in front of the point of divergence. [11]

> DEFINITION 3: A **type line** is the two innermost ridges that start parallel, diverge and surround or tend to surround the pattern area. [11]

The problem with using shapes or singular points is that the features extracted are only used for identification. Verification is done separately and makes use of different features, increasing the processing time. This is because shapes and singular point positions are not unique enough for positive matching.

In order to match two fingerprints Gabor filters can be used to find edges at different angles [12]. Alternatively minutiae in the ridges are unique and thus can also be used to verify if two fingerprints are the same [13]. They are where a ridge ends (truncation) or splits (bifurcation). There are other types as well which are complex compositions of the two [8, 10, 11]. Example minutiae are shown in Figure 1.2.

For this research a novel process for locating minutiae was considered. Software was provided by the Council for Scientific and Industrial Research, Modeling and Digital

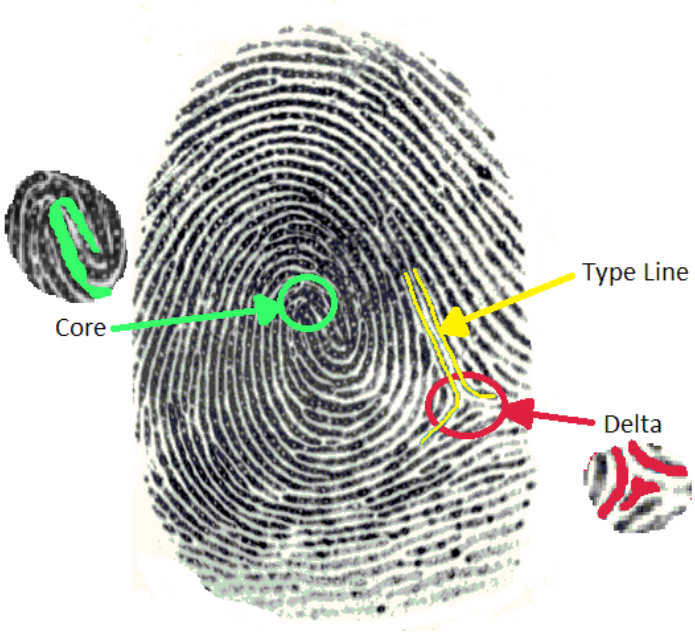


Figure 1.1: Core, Delta and Type Lines

Sciences, Information Security unit [14] for minutiae extraction, and this was used for the majority of the research. Testing the suitability of this software is covered in the Methods and Results chapters.

The question that this research seeks to answer is: Can the distances between minutiae be useful on their own? Since they could be used for identification or verification, the ability to assist each is examined individually. Verification is tackled as a graph-matching problem. Identification is handled by a neural network, primarily by looking for clusters – not necessarily the same as defined by [9] or any other existing classes.

The aim of this research is to improve on the existing techniques for fingerprint identification and verification. To do this new tools are brought in for feature extraction, classification, and fingerprint matching. The scope of this project is limited to minutiae based approaches. It will not look at other techniques such as Gabor filters. Only one new technique will be explored for each of the stages, i.e. minutiae extraction, verification, and identification.

Using only the distances between minutiae would save time for both problems. For verification the angles that the ridges the minutiae lie on and the types of minutiae would not need to be extracted. With identification no additional features, such as the ridge shape or the singular points, would need to be found.

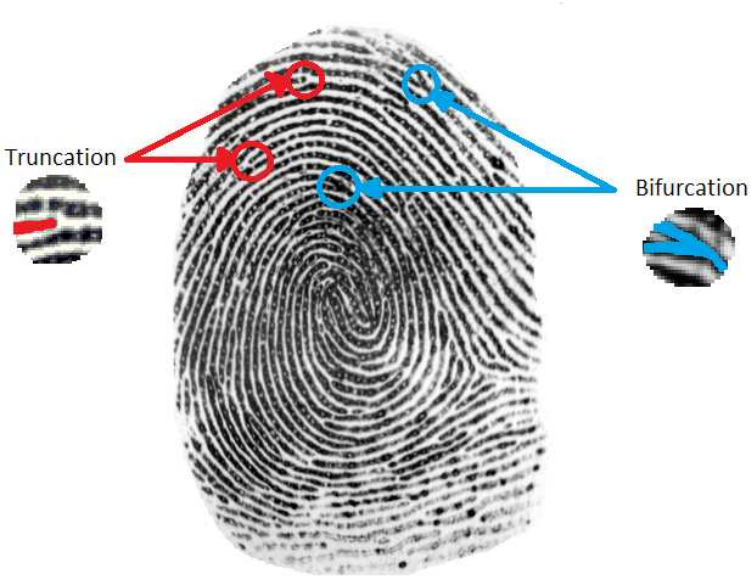


Figure 1.2: Minutiae

The second chapter of this dissertation covers the theory behind the key concepts of the study. Firstly the process by which minutiae are extracted is covered as this provides insight into the problem. For fingerprint matching it is important to understand the basics of graphs as they are understood by computer science and mathematics, which is not a simple diagram. The focus of the graph theory that will be explained is graph matching. Simply put, this is the problem of determining if two graphs are similar. This will be used to determine if two fingerprints are from the same person. Specifically, the "graduated assignment for graph matching" algorithm will be explained and its choice justified. What will also be covered in this section is the method by which a fingerprint is turned from minutiae into a graph.

The two artificial intelligence techniques that are used are Artificial Neural Networks (ANN) and Particle Swarm Optimisation (PSO). Within the realm of ANNs are Multilayer Perceptrons (MLP) which can be used as a classifier. Chapter two will explain how an MLP is built and how it can be trained with PSO.

Chapter three of this dissertation is a description of implementation and testing. First, a novel method of minutiae extraction is shown, this is then followed by a discussion of the method by which it was tested. The graph-matching algorithm is tested independently of the application. This chapter will show how this was done in addition to how it was tested for the application. Finally, the chapter covers how the classifier was built and tested, first apart from the application, then with it. The

point of the application independent tests is to prove that the implementation of the algorithms was correctly done.

Results and analysis are presented together for each of the three experiments in chapter four: Minutiae extraction, fingerprint matching and classification.

Finally, in the last chapter, conclusions are drawn and recommendations for future research are made. The recommendations are based on gaps discovered during the course of the research presented in this dissertation. The conclusions are based on the analysis of the results. A key conclusion to be drawn is whether any, or all, of the methods proposed by this study will have practical application.

Chapter 2

Theory of Key Concepts and Related Research

It is crucially important to explore new ways for biometric identification in the hopes of speeding up processes. The results of such improvements have far-reaching consequences for faster and better access control, as well as an enhanced utility of fingerprints in forensics. The foremost concern in this research is speeding up the processes involved in fingerprint recognition, a concern highlighted in the current literature. Important key concepts are necessary to use and expand on in this venture, particularly the concepts of fingerprint identification and verification. In the praxis of fingerprint biometrics, fingerprints are computer images. As such, the tools used in fingerprint identification and verification are contextually based in the field of computer vision. One of the major challenges posed is how to intuitively store and represent fingerprints as data. Graphs provide a highly useful solution to this problem, for both the human reader and computational processors. Taking particular features of fingerprint images and then visually representing them in graph form is a method which enables identification and verification of different fingerprint images. However, the computational processing speed which is required to do such fingerprint matching, and character identification and verification, poses yet another challenge. The goal of this study is to speed up the process by reducing the amount of information extracted from the fingerprint images. This is a unique approach which opens up new avenues of research.

Both the verification and identification of fingerprints are crucial steps in the biometrics associated with fingerprints. While these methods do not preclude the use of similar methods, if their effectiveness is established, they are sufficient to prove the concept of only using the distances between minutiae for identification

and verification.

There are several basic ideas on which identification is based. They are correlation-based, minutiae-based, textural-based and geometric-based [8]. A brief description of these follows. Minutiae matching is described in more detail, because a modification to this is how verifying that two fingerprints are the same is done in this research.

Correlation-based techniques seek to find similarities between the images as a whole. Normally this is done by maximising an approximation of the cross correlation function of the two images. If the cross correlation is high enough then it is a match [15]. The approximation is used as finding it exactly is computationally expensive [15]. These techniques are the fastest, but they are not very reliable as they are heavily affected by changes in how the fingerprint was scanned [8].

Minutiae-based techniques seek to find the similarity score. It is explained in [8] that if the score is high, it is a match. A general view of minutiae matching is done by considering a minutiae point to be a triplet $m = \{x, y, \theta\}$ where x and y are the coordinates and θ is the angle. If there are two fingerprints, T and I, which are vectors of minutiae m and m', then matching them becomes finding the number of minutiae points that correspond between I and T as a ratio of the average number of minutiae in both, i.e.

$$\textbf{Score} = \frac{k}{(n+m)/2}, \tag{2.1}$$

where k is the amount of matching minutiae, and n and m are the amount of minutiae in I and T. To find k, one much choose a spatial tolerance r_0 and an angular tolerance θ_0. If the spatial distance (sd) between two minutiae is lower than r_0 and the direction difference (dd) is lower than θ_0, then it is a match. In other words if

$$sd(m_i, m'_j) = \sqrt{(x_i - x'_j)^2 + (y_i - y'_j)^2} \leq r_0, \tag{2.2}$$

and

$$dd(m_i, m'_j) = \min(|\theta_i - \theta'_j|, 360^\circ - |\theta_i - \theta'_j|) \leq \theta_0, \tag{2.3}$$

then m_i and m'_j are a match. It is important to rotate and displace one image so that it is in the same orientation and position as the other. This is to maximise the number of matching minutiae. This research aims to make a matching algorithm that is free of the reliance on this transformation.

Textual fingerprint matching is similar to minutiae-based matching techniques. However, instead of finding minutiae, it finds irregularities in the ridge pattern [16]. Geometric-based matching is based on finding sampled points along the ridge lines

to get a sense of the pattern of the ridges. This is then rotated and reorientated to maximise the similarity between the two [17]. Neither of these methods have gained widespread popularity as they are computationally expensive [8].

Fingerprints are identified by verifying that the scanned fingerprint matches one in a database. As comparison is computationally expensive, it is important to limit the size of the search. In aid of this, fingerprints are grouped into classes. There are several techniques for this.

A rule-based approach uses the number of singularities and their position to determine which class a fingerprint is in. This is based on the methods that are done manually. The number and position of the singularities give the shape. For example, if there are two loops and two deltas, then it is a whorl [18].

A structural approach uses low-level features to produce higher level structures. An example of this is from [19] where the orientation of the pixels (see § 2.1) is divided into segments. The centre of these segments then forms a reference against which fingerprints can be compared. This is much faster than directly comparing them.

While there are other techniques, these are the most common. Examples of these other techniques are syntatic, statistical and Computational Intelligence (CI)-based approaches [8]. The most common form of a CI based approach is using a neural network, though other techniques like Hidden Markov Models are used. A description of some techniques used in fingerprint identification and verification is given in § 2.3.1. While a neural network is used for classification in this research, it is not in the same way as is normal. The standard approach being to use features of the orientations [8]. In this research the classification is meant to be free of finding the orientation of the pixels in the image.

The remainder of this chapter covers the tools which can be used for verification and identification. For verification, graph isomorphism is established using the "graduated assignment algorithm for graph matching" [20]. The tool used in identification is a Multilayer Perceptron (MLP) Artificial Neural Network (ANN), tuned with Particle Swarm Optimisation (PSO). However, all the techniques presented here are based on minutiae. Thus an explanation of the process through which they are extracted is given. This will also be important for the novel minutiae extraction process that is developed.

Figure 2.1: Ridge Orientation

2.1 Minutiae Extraction

There are many ways to extract minutiae. Presented here is the method most commonly used in the industry. First the image is normalised, then the orientation of the ridges and their frequency are estimated in local areas [21]. The image is then enhanced to remove small gaps, imperfections and noise. Finally, the image is thinned and the minutiae are located based on the amount of neighbouring pixels [8]. This process is discussed in futher detail below.

Local ridge orientation estimation follows the normalisation process. The direction of the ridges is estimated for each pixel or small windows, illustrated in Figure 2.1[1]. The two most common bases for local ridge orientation estimation tools are gradients, and slits and projections. A gradient-based approach finds the direction of the ridge based on the derivative of the change in shades in the x and y directions over a window around the pixel [22]. These windows are normally 9x9 or 17x17 pixels [23]. The angle of the ridge is orthogonal to the angle at which the derivatives are minimised [8]. Slit and projection-based approaches, on the other hand, work on fixed angles. First the standard deviation of the shades of grey for each slit is found. The direction of the slit with the greatest difference between itself and its orthogonal counterpart is the direction of the ridge [8].

The next part of the process is to calculate the ridge frequency. The methods for

[1]Reproduced with permission from [22]

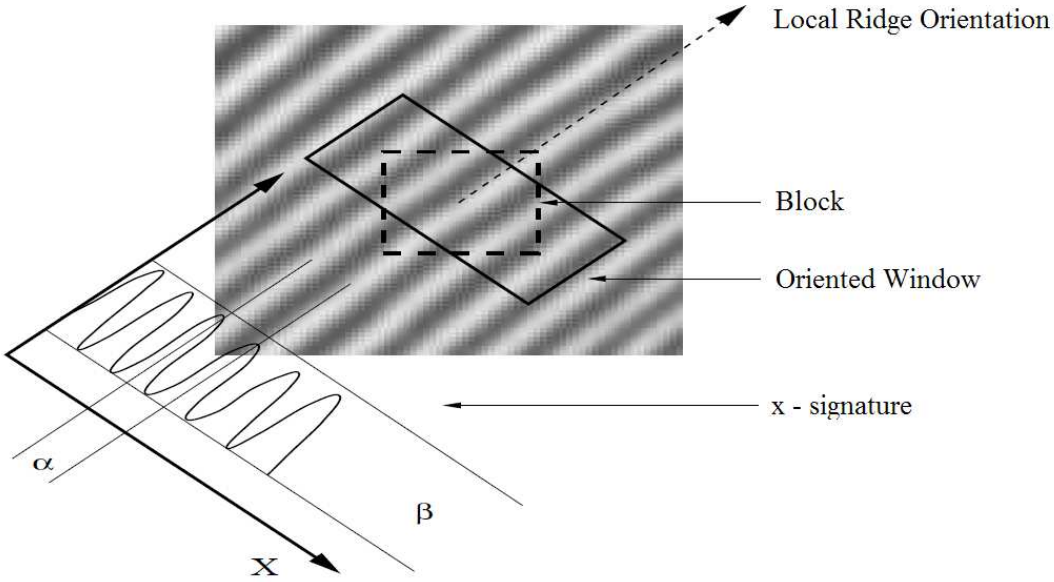


Figure 2.2: Ridge Frequency

finding the ridge frequency are based on the following from [8]. Figure 2.2 represents a small part of a fingerprint image[2]. The x-signature is a measure of how dark the image is on average across the y-axis of the oriented window. This average is termed h. The x-signature is normally modelled as a sinusoidal wave with an amplitude of β and average amplitude of β_m. If the variation V of h in the interval $[x_1,x_2]$ is defined by

$$V(h) \;=\; \int_{x_1}^{x_2} \left| \frac{dh(x)}{dx} \right| \cdot dx. \tag{2.4}$$

Then if the h is periodic or the change over the interval is small then (2.4) becomes

$$V(h) \;=\; (x_2 - x_1) \cdot 2\beta_m \cdot f \tag{2.5}$$

where f is the local frequency and this can then be rearranged as

$$f \;=\; \frac{V(h)}{2 \cdot (x_2 - x_1) \cdot \beta_m}. \tag{2.6}$$

The first part of the image enhancement is called segmentation. This is the process by which the background and unrecoverable parts of the image are discarded. This reduces the amount of processing required in the rest of the enhancement procedure. The difference between a recoverable region and an unrecoverable one is shown in

[2]Reproduced with permission from [21]

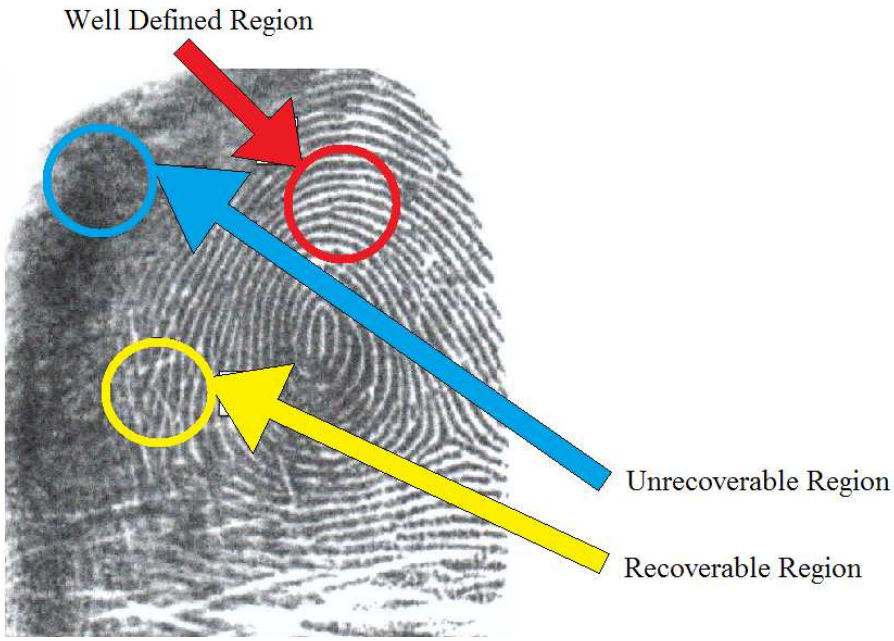


Figure 2.3: Recoverable vs. Unrecoverable Regions

Figure 2.3. Segmentation first divides the image into regions, then it determines the quality of that region. This is based, not on the intensity of the amount of grey, but on the existence of a striped and oriented pattern [8]. This has to be robust to the noise generated, for example, by dust or grease. Two examples of robust techniques that have been developed are those that have been proposed by Mehtre and Chatterjee in 1989 [24], and Ratha, Chen and Jain in 1995 [25].

The Mehtre and Chatter method uses 16x16 blocks. A histogram of the orientation of each pixel in the block is computed. If there is a peak, this represents an oriented area as opposed to flat or near flat regions which are background. This method also finds the grey-scale variance of each block to prevent problems with perfectly uniform ones [24].

Ratha, Chen and Jain's method also uses 16x16 pixel blocks. However, their method finds the variance of grey in the direction orthogonal to the ridge orientation. The variance of this value over the block is used to determine the areas of interest. It does this because regions of interest have high variance in the orthogonal direction [25].

The next step in the enhancement stage of minutiae extraction is generally one which only relies on an individual pixel and a global parameter. Example techniques are normalisation and histogram equalisation [8]. Histogram equalisation will be used in the novel approach described in § 3.1. Briefly, it is a process by which a lookup

table is created in such a way that, by replacing the value of the original shade of grey with the one found in the lookup table, the histogram of all the shades of grey will be flattened. The effect is to increase the contrast of the image.

In order to follow the increase in contrast, some form of contextual filtering is used. It is contextual because the filter will use change based on local information of the surrounding area. The effect of the filter aims to eliminate cuts and other imperfections in the fingerprint. No perfect filter has been discovered but there have been several attempts. These generally rely on the fact that the sinusoidal pattern of ridges and valleys only varies slowly. As an example, the method proposed by [21] works as follows. A Gabor filter, which is even and symmetric, defined by

$$g(x, y : \theta, f) = exp\left\{-\frac{1}{2}\left[\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right]\right\} \cdot cos(2\pi f \cdot x_\theta), \tag{2.7}$$

σ_x and σ_y represent the standard deviation of the Gaussian Envelopes, which define the Gabor filter along the x and y planes. Also, f is the frequency of the sinusoidal plane wave that adds to the description of the filter. Finally, x_θ and y_θ are the coordinates of x and y after they have been rotated by $(\pi/2 - \theta)$,

$$\begin{bmatrix} x_\theta \\ y_\theta \end{bmatrix} = \begin{bmatrix} sin\theta & cos\theta \\ -cos\theta & sin\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2.8}$$

The Gabor filter has four parameters which are chosen as follows:

- f which is chosen to be the same as the local ridge frequency.
- σ_x and σ_y which are found empirically to be $\sigma_x = \sigma_y = 4$.
- θ which is the ridge orientation.

However, to reduce the computational intensity of calculating each pixel's filter, a bank of precomputed filters are generated. These will have a range of θ and f values. The one chosen for each pixel is the one that most closely matches the respective values of that pixel [21]. Figure 2.4[3] shows the effect of using this filtering technique.

The next step is to binarise the image. This means turning the image from grey-scale to black and white only. The simplest approach is to turn everything darker than a threshold to black, and everything lighter to white. A slightly more advanced method will be to base this threshold on the actual average intensity of the image. Other

[3]Reproduced with permission from [8]

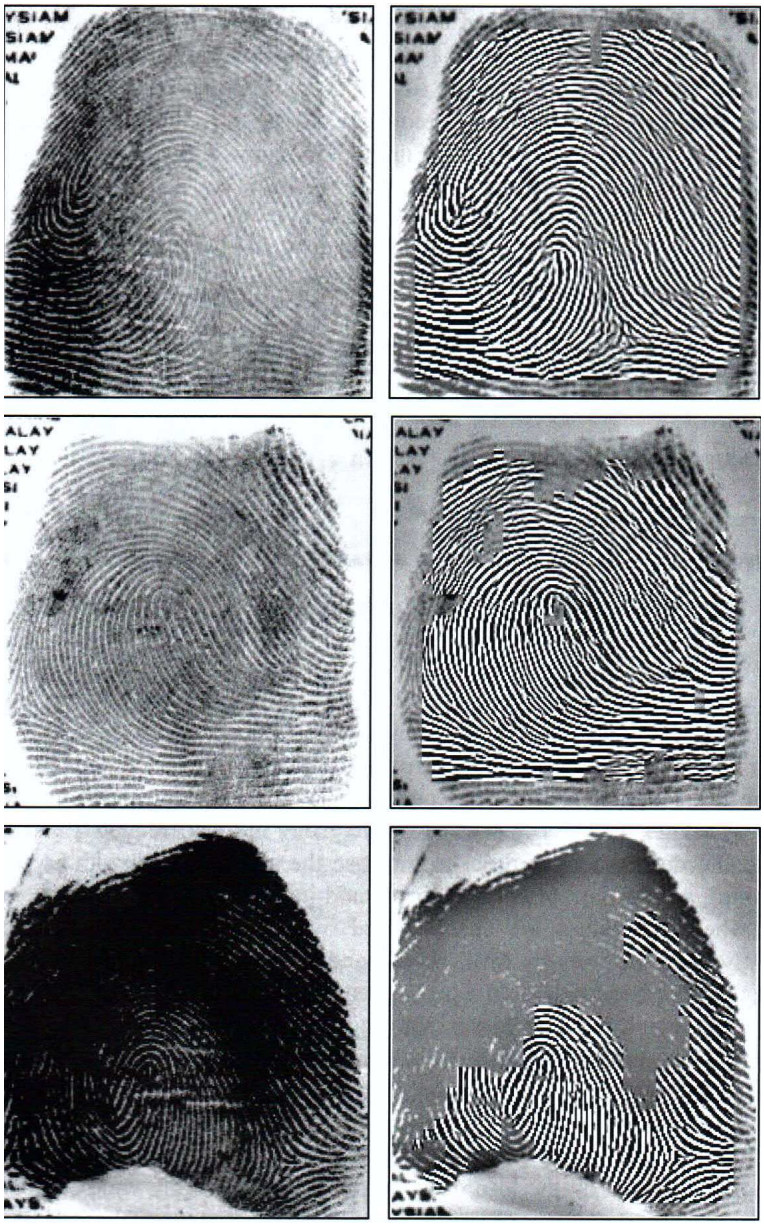


Figure 2.4: Gabor Filter Enhancement

methods specific to fingerprints have been developed. For example Ratha, Chen and Jain in [25] find a 16x16 window around each pixel. This window is orthogonal to the direction of the ridge as was estimated before. A grey-level profile is compiled from this as shown by Figure 2.5[4], the peaks and surrounding two neighbours are moved to the foreground [8]. The effect of this is shown in Figure 2.6.

The last stage before the minutiae are located is called thinning. In this step the fingerprint is reduced to a one pixel wide skeleton [8]. There are many thinning

[4]Reproduced with permission from [8]

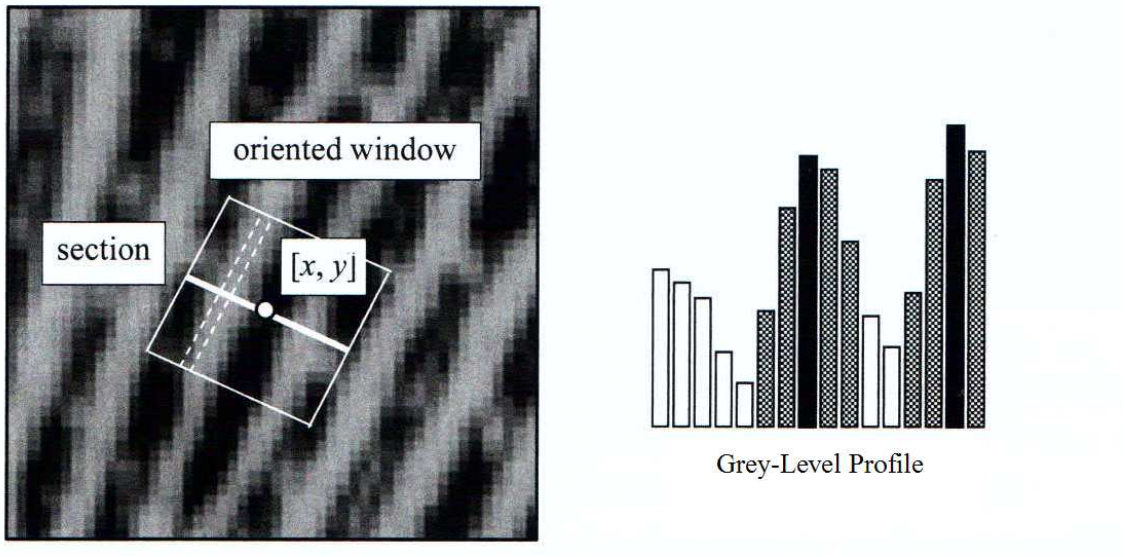


Figure 2.5: Construction of Grey-Level Profile

algorithms; some have been designed specifically for fingerprints, for example [26]. The details of one such algorithm are presented in § 3.1. The effects of this stage is shown is Figure 2.7.

Finally, the minutiae are located on the thinned image by counting the amount of neighbours each pixel has. If it has one, it is a truncation, and if it has three, it is a bifurcation. The angle of the minutiae is the angle of the ridge. Thus all the attributes of the minutiae are known at this point [8].

2.1.1 Current Work

Recent research into minutiae extraction mostly seeks to improve one or more stages of this process. This section gives some examples of this recent work.

In [27] a new approach is taken for extracting the fingerprints. Rather than treating the image as a sequence of pixels they attempt to view the image as a series of curves. The type of curve is called a principle curve. These are curves which satisfy the constraints:

- It does not intersect with itself.
- It has a finite length

Figure 2.6: Binarisation of a Fingerprint

Figure 2.7: Effect of Thinning

- They are self consistent (there are no discontinuities)

They find a curves on the skeletonised image and using logical filtering extract minutiae at the ends of the principle curves. Simply if the point exists in only one curve then it is a ending, otherwise if it exists in multiple curves then it is a bifurcation. Their heuristics for filtering are things like checking that two close together ending points could be connected by a curve that is consistent with the other two curves. Their results are that they find on average 2.3 minutiae that are

false, 2.7 are minutiae that are missing, and 42,7 minutiae are found in each image.

In [28] the authors seek to include new methods of pre and post processing. A subtle difference in their method is that they use fingerprint valleys instead of ridges. The reason for this is that the valleys start out thinner than the ridges making them easier to thin. Their contribution to the preprocessing stage is a new method to remove isolated pixels. And their post processing is a filter for removing spurious minutiae. Their results show that the preprocessing will give a 3.4% increase in accuracy, and the post processing only 15.3% of the minutiae will be false.

In [29] the use of the symmetry of the fingerprint ridge is exploited. By using a Gaussian filter the authors demonstrate an improved ability to remove features that would give spurious minutiae. The advantage is that skeleton is more likely to correctly reflect the fingerprint structure. This resulted in a worst observed error rate of 9.8%.

2.2 Graduated Assignment for Graph Matching

Automated fingerprint recognition naturally falls into the greater problem of computer vision. Computer vision is the science of recognising objects, features and patterns in images. Tools used in fingerprint recognition are typically from this field. Gabor filters, for example, have seen widespread use in computer vision in general [30–32], but are also a method used in fingerprint verification in particular. Gabor filters are a method for detecting edges of objects in an image. When applied to fingerprints, the objects in question are the ridges. Since fingerprint scans are a type of image, this science is highly applicable and useful when verifying and identifying fingerprints.

Computer vision relies heavily on the way features are extracted or recognised, and the way those features are presented. A highly useful visualisation tool is a graph which can be used to represent features [33]. Any particular feature on an image is easily identified by a computer. Features, for example, can be the variations in colour, texture and/or depth within an image. To be useful for any application, these features must be stored in some way that is intuitive to a computer. One way in which features can be stored and then represented is in graph form. There is considerable theory relating to the utility of graphs in computer science. One of the arguments put forward by this body of work is that an advantage of storing features

in graph form is that this avenue is highly intuitive, not only to computers, but also to humans.

A graph is a collection of vertices (otherwise known as nodes) and edges. A vertex is a point on the graph which has attributes. An attribute, in this case, is a clear, recognisable characteristic of the vertex, making each vertex entirely unique. On images in general a vertex can be considered as a 'point feature'. Point features are singular points on an image which would have annotations such as position, type, colour, direction, size and/or shape. [34]. Minutiae are a good example of point features. Minutiae are composed of position, type and angle of the ridge (the raised part of the fingerprint) [8]. Point features are the easiest to extract as they can be a single pixel, as opposed to a whole region of the image which would make it more complex to process.

The normal way to represent each vertex is as a set. In general this would be $v_i = a_{i0}, a_{i1}, ...a_{in}$ where i is the number of vertices and n is the number of attributes. It should be noted that the attributes are not necessarily numerical, but it is standard practice to enumerate as many non-numerical attributes as possible. As an example in the case of minutiae v_i, would be an individual minutia, while the attributes would be labelled for minutia i as follows:

- a_{i0}: The x coordinate
- a_{i1}: The y coordinate
- a_{i2}: The angle
- a_{i3}: The type of minutia, enumerated as 1 for a truncation and 3 for a bifurcation. These values are the number of neighbouring pixels of a minutiae point.

As it only applies to one point in the graph, this is known as a unary attribute. When discussing all the vertices in a graph it is common to use a matrix. In this case each row is a different vertex, so row i would be v_i. The columns are the different attributes. That is column j would be attribute a_j. This is shown graphically in Figure 2.8.

Connecting two vertices, such as two minutiae, are edges [35]. Edges are links which, like vertices, can have attributes such as cost and direction. Direction can mean that the two vertices are only linked in such a way that the flow is unidirectional,

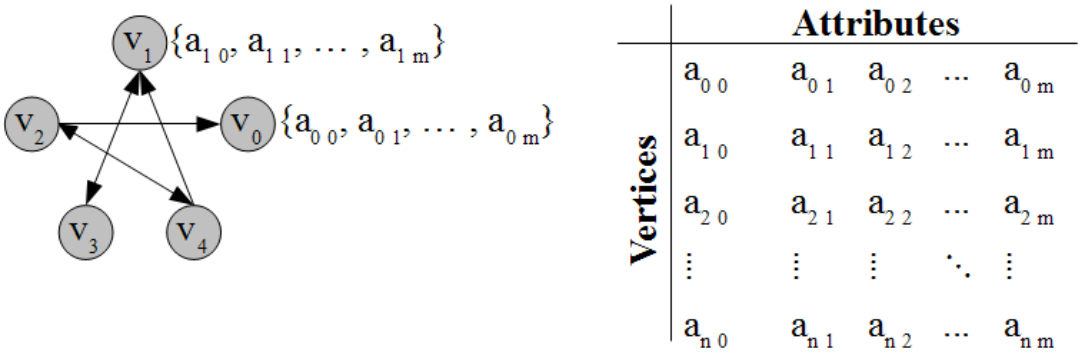


Figure 2.8: Illustration of How Vertices are Represented as a Set or Matrix

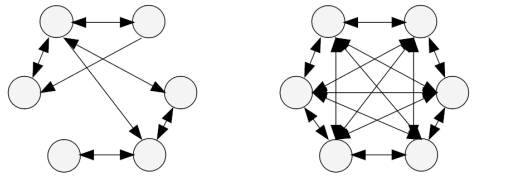


Figure 2.9: Graphs

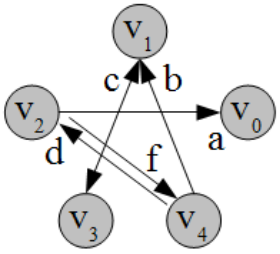
as opposed to exhibiting a bidirectional connection. As these attributes link two points on the graph, they are called binary attributes. One way of creating edges would be to calculate the distance between the two point features of an image. In this case, the edge is bidirectional as the distance is true of both ways. If all the distances are computed, then all the vertices are connected and the graph is complete [35]. A complete graph is a graph in which all vertices are connected to all other vertices. Figure 2.9 shows the difference between a partially connected graph with unidirectional links and a complete graph pictorially. This method of representing a fingerprint as a graph is illustrated in Figure 2.10. It opens up the application of graph theory to fingerprint recognition.

A way of representing a graph's binary attributes is in a matrix. If there is a link between the vertex indexed by i to the vertex indexed by j, then that value is enumerated, if appropriate, and stored on row i column j. This allows for unidirectional links because, element ij is not necessarily the same as ji. If there is no link then a 0 or NULL can be stored. This is shown in Figure 2.11. In the case of the fingerprint graphs, the matrix is symmetrical about the diagonal and the

Figure 2.10: Fingerprint Graph

diagonal elements are 0. This is because the distance from minutia i to minutia j is the same as the distance from minutia j to minutia i. Hence it is also bidirectional.

If a fingerprint is represented as a graph, then verifying whether two fingerprints are the same entails confirming that the two graphs are identical. In graph theory this is called graph matching [36] or isomorphism [35]. In this case the number of nodes of the graph dictates the size of the problem. This definition of the size of the problem is common in graph isomorphism detection algorithms [37, 38]. Graphically speaking, this means finding that in Figure 2.12, A_1 and A_2 are sufficiently similar to be called identical.



		Destination			
Source	0	0	0	0	0
	0	0	0	c	0
	a	0	0	0	f
	0	c	0	0	0
	0	b	d	0	0

Figure 2.11: Illustration of How Edges are Represented as a Matrix

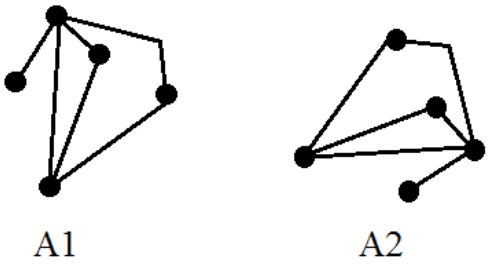


Figure 2.12: Two Isomorphic Graphs

To determine if two graphs are isomorphic a method of comparison needs to be found. One such comparison would be to determine if the graphs are permutations of one another. This would mean

$$A_2 = MA_1M^T \tag{2.9}$$

is true for two graphs defined by A_1 and A_2 for some permutation matrix M. If this can be found exactly, then the graphs are identical. This is for both types of attributes, assuming there is a function C that compares how similar two graphs are. If M is found such that A_2 is sufficiently similar to MA_1M^T, then it can still be said they are isomorphic. Sufficient similarity would be a threshold on the results of C. The problem of finding M, given A_1 and A_2, is the graph-matching problem. This problem is extended to include finding whether M exists at all. It might be the case that

$$A_2 = MA_1M^T + \epsilon, \tag{2.10}$$

where ϵ is a noise matrix. Because of this, it is necessary for a graph-matching

algorithm to be tolerant to the noise, even if the noise causes nodes or edges to be lost [20].

This discussion does not include constraints applied by the vertex attributes. Details of this type of graph matching can be found in [39]. As this research does not focus on matching these attributes it will not be explained here.

Determining if two graphs are the same using graph matching is not a new concept. But research in using graph matching for graph isomorphism detection continues. Normally this is done by computing a distance metric. This is a measure of how similar the two graphs are. In effect it measures the quality of the match defined by M. Examples of this technique can be found in [40, 41]. This is but one technique among many but it is the approach that was used in this research. It is appropriate because it has relatively low complexity (explained below) and works for problems where the common sub graph may not be large. In the case of fingerprints, it is often that only part of the fingerprint will be observed and it may be a different part than the original reference. In this case it is important that the fingerprint matching algorithm be able to deal with different minutiae. It has already been shown that graph matching can be used with fingerprints in [42].

Polynomial complexity means that the amount of steps needed to solve a problem is dependent on the size of the problem with some polynomial function [43]. It is unknown whether the graph-matching problem is NP-Complete or NP-Hard. This means that a solution can be verified in polynomial time. Equation 2.10 shows this to be the case, but it is difficult to generate the optimal solution. If it is NP-Complete, it would be possible to solve it on a computer with an infinite amount of parallel processors, purely by making decisions. If it is NP-Hard then it means that it is not possible to use only simple decisions to generate every possible answer, which would make it even more difficult than an NP-Complete problem. Regardless of which type it is, there is no known polynomial time method for generating the optimal solution. However, in specific cases it can be approximated in polynomial time, such as in the graph-matching problem [35].

Using graph isomorphism to match two fingerprints is the method by which verification is handled in this study. One aim of this research is to determine whether two fingerprints can be matched using only the distances between the minutiae on the two fingerprint images. This means matching fingerprints using only the binary attributes, rather than a combination of both types of attributes. However, a comparative test was done between these two methods to determine the value of

this narrowed focus, using only the binary attributes. This comparison indicated the effect of including the unary attributes.

It is important to find an efficient method for determining whether two graphs are isomorphic, as this is how it can be determined if two fingerprints are the same. The estimated running time will give an idea as to which application it can be used for. If it is, on the one hand, highly accurate but slow, then it is suited for forensics. On the other hand, if it is fast and less accurate, then it could be used for access control. If it is both fast and accurate then it could be used for either. If it is neither fast nor accurate, then it is unsuitable for fingerprint verification. In this section first the algorithm that was used in the research is presented. Then an estimate is made for the runtime of the algorithm. Based on these findings a prediction is given as to which applications it will be suitable for.

The graduated assignment algorithm finds a match matrix [20]. [20] explains that the match matrix represents how one graph is permuted to find the other. The match matrix is called M. The graphs are G with A nodes and g with I nodes. G_{ab} is a matrix wherein each element is the value of the binary attribute between nodes a and b. g_{ij} is similarly defined. The aim of the algorithm is to find an M that minimises (2.11).

$$E_{wg} = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}C_{aibj}. \tag{2.11}$$

Subject to all the rows and columns of M totalling to one and $\forall ai\ M_{ai} \in \{0,1\}$. Where C_{aibj} is defined as

$$C_{aibj} = \begin{cases} 0 & \text{if } G_{ab} \text{ or } g_{ij} \text{ is NULL} \\ c(G_{ab}, g_{ij}) & \text{otherwise} \end{cases}. \tag{2.12}$$

Where $c(G_{ab}, g_{ij})$ is some function which represents how well G_{ab} and g_{ij} match. When the algorithm has finished running

$$M_{ai} = \begin{cases} 1 & \text{if } a \text{ in } G_{ab} \text{ corresponds to } i \text{ in } g_{ij} \\ 0 & \text{otherwise} \end{cases}. \tag{2.13}$$

In the original paper, the authors assume there will be a uniform distribution to the size of the binary attributes in the graphs [20]. This is not the case for fingerprint distance graphs. The probability distribution of the distances in the FVC2000.4 database[5] is shown in Figure 2.13 as an example. This can best be

[5]The fingerprint verification competition provides databases of fingerprints every two years. There are four sets of data in each database. Each one is captured in a different way. FVC2000.4 refers to the fourth set from the 2000 competition.

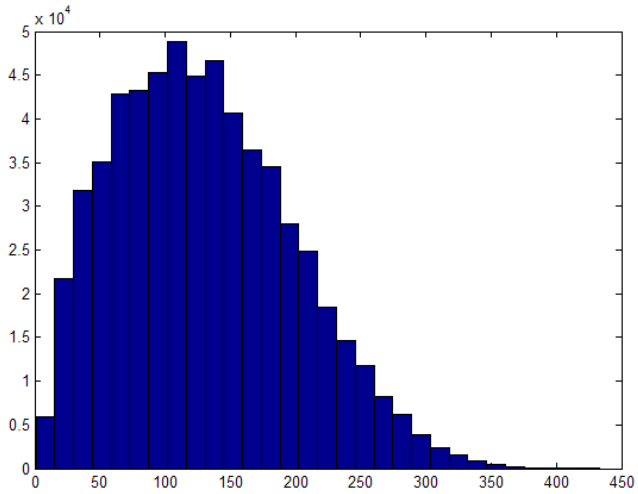


Figure 2.13: Probability Distribution of Minutiae Distances

approximated by a beta probability distribution function [44]. This is because the distribution is bounded by the size of the greatest diagonal on the image. Using this distribution (2.14) is the comparability function,

$$c(G_{ab}, g_{ij}) \;=\; 1 - \frac{(\alpha+\beta)^2(\alpha+\beta+1)}{2\alpha\beta}(G_{ab} - g_{ij})^2, \tag{2.14}$$

which is derived as follows.

In the original paper the comparability function has a maximum value of one and an expected value of zero. This is achieved with the function being defined as

$$c(G_{ai}, g_{ij}) \;=\; 1 - 3|G_{ai} - g_{ij}|. \tag{2.15}$$

This is because the expected absolute distance between two points on a uniform distribution is a third.

For this application the values are beta distributed. To match the requirements of the original comparability function, the perfect concurrence must be one and the expected value must be zero. This can be achieved with a function of the form given in (2.16). Where A is the inverse of the expected value of $(G_{ai} - g_{ij})^2$.

$$c(G_{ai}, g_{ij}) = 1 - A(G_{ai} - g_{ij})^2 \tag{2.16}$$

Note that if $G_{ai} = g_{ij}$ then $c(G_{ai}, g_{ij})$ is one. The expected value of $c(G_{ai}, g_{ij})$ is zero.

The derivation of A is as follows:

$$A = \mathbf{E}[(G_{ai} - g_{ij})^2]^{-1}, \tag{2.17}$$

$$\mathbf{E}[(G_{ai} - g_{ij})^2] = \mathbf{E}[G_{ai}^2 - 2G_{ai}g_{ij} + g_{ij}^2], \tag{2.18}$$

where $\mathbf{E}$ is the expected value. Because G_{ai} and g_{ij} are Independently and Identically Distributed (IID), this becomes

$$\mathbf{E}[(G_{ai} - g_{ij})^2] = \mathbf{E}[G_{ai}^2] - 2\mathbf{E}[G_{ai}]\mathbf{E}[g_{ij}] + \mathbf{E}[g_{ij}^2]. \tag{2.19}$$

Further, if G_{ai} and g_{ij} are replaced by X in the expectations, then this simplifies to (2.20). This is valid because they are IID.

$$\mathbf{E}[(G_{ai} - g_{ij})^2] = \mathbf{E}[X^2] - 2\mathbf{E}[X]\mathbf{E}[X] + \mathbf{E}[X^2] \tag{2.20}$$

Which simplifies to

$$\mathbf{E}[(G_{ai} - g_{ij})^2] = 2\mathbf{E}[X^2] - 2\mathbf{E}[X]^2. \tag{2.21}$$

But the variance of X,

$$\mathbf{Var}(X) = \mathbf{E}[X^2] - \mathbf{E}[X]^2, \tag{2.22}$$

hence

$$\mathbf{E}[X^2] = \mathbf{Var}(X) + \mathbf{E}[X]^2. \tag{2.23}$$

Using this (2.21) becomes

$$\mathbf{E}[(G_{ai} - g_{ij})^2] = 2(\mathbf{Var}(X) + \mathbf{E}[X]^2) - 2\mathbf{E}[X]^2 \tag{2.24}$$

$$\mathbf{E}[(G_{ai} - g_{ij})^2] = 2\mathbf{Var}(X). \tag{2.25}$$

A beta distributes random variable with parameters α and β has a variance of

$$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

hence

$$\mathbf{E}[(G_{ai} - g_{ij})^2] = 2\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}, \tag{2.26}$$

therefore

$$A = \frac{(\alpha+\beta)^2(\alpha+\beta+1)}{2\alpha\beta}. \tag{2.27}$$

So

$$c(G_{ai}, g_{ij}) = 1 - \frac{(\alpha+\beta)^2(\alpha+\beta+1)}{2\alpha\beta}(G_{ai} - g_{ij})^2. \tag{2.28}$$

The identities and results employed are from [45] and [46].

While M has A rows and I columns, the algorithm uses an extra row and column for the optimisation. The extra cells allow for slack in the optimisation process which changes the problem from a binary integer program to a quasi-linear program. This [20] terms an augmented matrix and is denoted $\hat{M}$ [20]. They also use a softmax assignment which has a control parameter β [20]. Using this the matching algorithm is:

1 From β_k = β_0 to β_f where $\beta_{k+1} = \beta_k\beta_r$
2 Do until M converges or reaches an iteration limit
3 $Q_{ai} = -\frac{\partial E_{wg}}{\partial M_{ai}}$
4 $M^0_{ai} = \exp(\beta Q_{ai})$
5 Do until $\hat{M}$ converges or reaches an iteration limit
6 Normalise all rows:
7 $\hat{M}^1_{ai} = \frac{\hat{M}^0_{ai}}{\sum_{i=1}^{I+1}\hat{M}^0_{ai}}$
8 Normalise all columns:
9 $\hat{M}^0_{ai} = \frac{\hat{M}^1_{ai}}{\sum_{i=1}^{I+1}\hat{M}^1_{ai}}$
10 Perform Clean-up heuristic

where Q_{ai} is found using (2.29) [20]

$$Q_{ai} = -\frac{\partial E_{wg}}{\partial M_{ai}} = \sum_{b=1}^{A}\sum_{j=1}^{I} M_{bj}C_{aibj}. \tag{2.29}$$

The clean-up heuristic turns M into a matrix which only contains ones or zeros. Although a more complicated procedure could be used, [20] claims that simply finding the largest value in each column, and setting it to one, and all other values in that column to zero, is sufficient. This is the approach that is used in this research.

Once M has been generated using (2.11), the fitness E_{wg} can be calculated. This gives an indication as to the confidence of the match. If it is low then the graphs are isomorphic. If it is high then the graphs do not match.

Gold and Rangarajan [20] modified their algorithm to account for unary attributes in a graph. This is done by changing the objective function to

$$E_{awg} = -\frac{1}{2}\sum_{a=1}^{A}\sum_{i=1}^{I}\sum_{b=1}^{A}\sum_{j=1}^{I} M_{ai}M_{bj}C_{aibj} + \alpha\sum_{a=1}^{A}\sum_{i=1}^{I} M_{ai}\sum_{s=1}^{S} D_{ai}^{s} \tag{2.30}$$

where S is the amount of unary attributes. In this equation α is the weighting given to the importance of the unary attributes. This is problem dependent. The D term acts like the C term in that it finds the compatibility of the unary attribute in G relative to the unary attribute in g. Because of this, D_{ai} is defined by

$$D_{ai}^{s} = d(G_a^s, g_i^s). \tag{2.31}$$

In this equation G_a^s and g_i^s are vectors that contain the unary attributes of the nodes of the graphs, where a and i are the nodes and s is the node index. Because the nature of unary attributes are problem-dependent, so too is the exact definition of d in (2.31). In § 3.2 the definition for the fingerprint problem is discussed.

The change that is given by (2.30) necessitates a change in the method by which Q_{ai} is found. To accommodate the D term Q_{ai} is evaluated with

$$Q_{ai} = \sum_{b=1}^{A}\sum_{j=1}^{I} M_{bj}C_{aibj} - \alpha\sum_{s=1}^{S} D_{ai}. \tag{2.32}$$

It should be noted in [20] that when discussing the modification to the algorithm to include the unary attributes, they used $C_{aibj}^{(2)}$ for C_{aibj} and $C(1,s)_{ai}$ for $D^{(s)}$. Also, in the original paper, $d(G_a^s, g_i^s)$ was called $c(G_a^s, g_i^s)$. The notation was changed to allow for greater clarity.

In Figure 2.14[6] the effects of randomly generated noise on the ability of the algorithm to make a match is shown with a graph without unary attributes. To generate this graph 100 node graphs were randomly generated and the trials were run 600 times per line. Each line represents a different amount of connectivity and how many nodes were deleted. This means that for one trial, a 100 node graph was generated with a certain percentage of the nodes connected. A certain amount of nodes were then removed and the new graph was compared with the old graph. As the way in which the nodes were deleted is known, the M matrix is also known. In this diagram:

- (a) 15% connectivity and 40% deletion

[6] Repoduced with permission from [20].

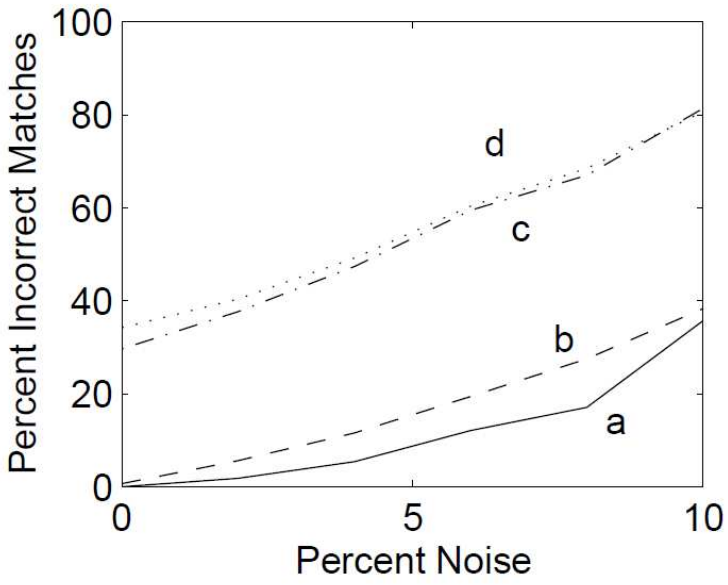


Figure 2.14: Graduated Assignment Resistance to Noise

- (b) 10% connectivity and 40% deletion
- (c) 15% connectivity and 60% deletion
- (d) 10% connectivity and 60% deletion

Figure 2.15[7] is similar to Figure 2.14, but shows the effects of including unary attributes to the graphs. In this case the graphs had three or five binary valued attributes. Meaning they can only take on values of zero or one. None of the unary attributes were mislabeled. In all the experiments there is a 10% connectivity and only one type of link. This denotes that all the links have the same value. The graphs have 100 vertices and each line is the average of 1100 experiments [20]. The lines represent:

- (a) 60% deletion and 5 attributes
- (b) 60% deletion and 3 attributes
- (c) 80% deletion and 5 attributes
- (d) 80% deletion and 3 attributes

[7]Repoduced with permission from [20].

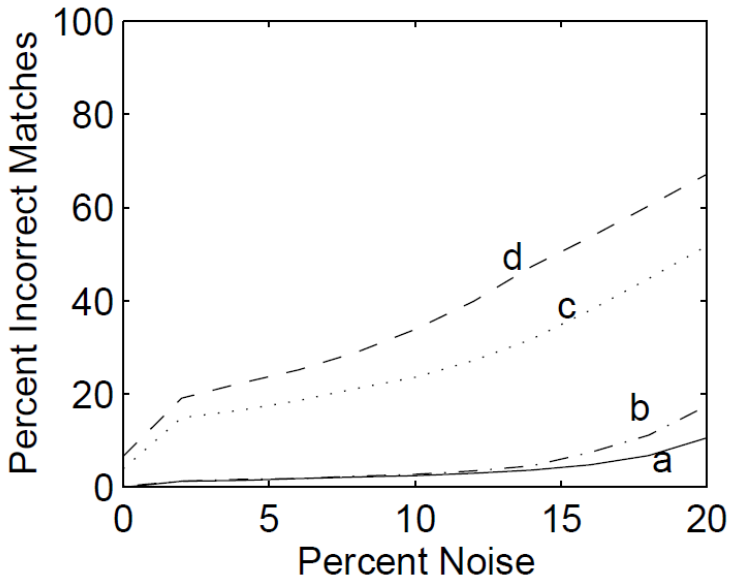


Figure 2.15: Graduated Assignment with Attributes Resistance to Noise

To show the ability of the graduated assignment for graph matching, [20] compared it to Probabilistic Relaxation (PR), another common technique for graph matching. They chose it for three reasons:

> First, they are both non-linear methods, in contrast to combinatorial approaches to graph matching. Second, PR appears to be most successful standard method available for graph matching, at least, among non-linear methods. Third, because it is a widely known method, it can serve as a useful benchmark for our new approach. (Even if relative success can be disputed, being the most widely known non-linear method would make it a suitable control).

Figure 2.16 shows the results of this comparison. For their experiment 90 nodes of varying connectivity runs against 100 node graphs, graduated assignment is tested 700 times and PR is tested 70 times. It is noted in [20] that PR runs five to 15 times slower. This experiment is a test of the subgraph isomorphism (matching) abilities of the two algorithms.

There have been many attempts at finding efficient methods by which graph isomorphism can be established. Some examples are the popular Ullmann [37] and the Nauty [38] algorithms. However, they often have high processing time [47]. The 'graduated assignment for graph matching' algorithm runs in $O(lm)$ time, where

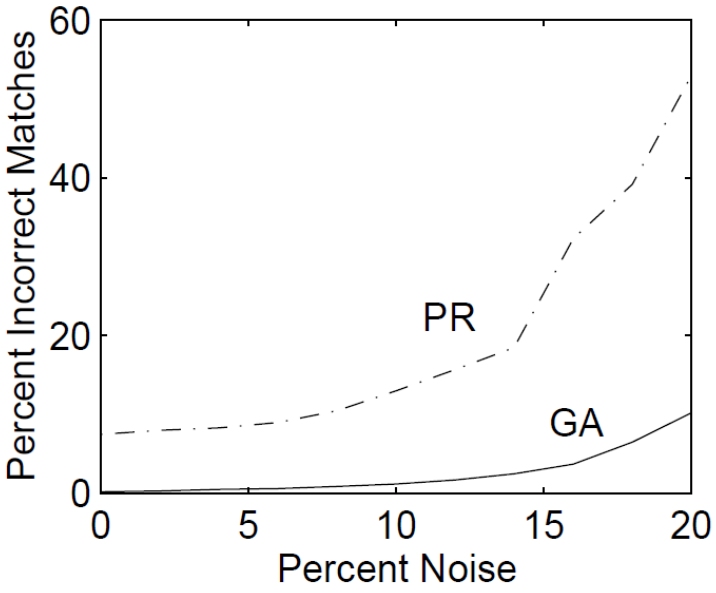


Figure 2.16: Comparison of Probabilistic Relaxation and Graduated Assignment

l and m are the numbers of nodes in each graph [20]. This is considered to be a low runtime relative to the size and complexity of such large graph-matching problems [47].

The software provided by the CSIR [14] will on average extract 34.567 minutiae points, though it can extract as many as 127. The lowest number extracted was 13. These were found using all four FVC2000 databases. This would mean that the graph matching will require between 169 and 16129 steps with an average of 1194.9 instructions. Figure 2.17 shows the distribution of the amount of minutiae extracted. Clearly it is more common for higher numbers to be extracted. This implies it would take longer and, hence, would only be suitable for forensic applications.

The graduated assignment for graph matching is a simple and powerful tool. It is capable of finding the permutation matrix that will match two graphs. It is simple to evaluate how good this match is, and hence determine if they are isomorphic. The algorithm is highly resistant to noise and can determine sub-graph isomorphism; this is important as it is possible that minutiae will be lost or different minutiae extracted on each scan. As this is based on the number of minutiae and the number can be quite high, it is not suitable for access control.

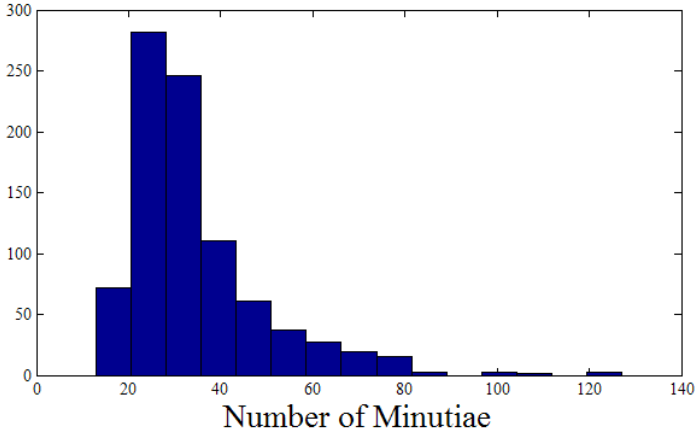


Figure 2.17: Distribution of the Amount of Minutiae Extracted

2.3 Artificial Neural Networks

It is common to look at features of an image to find patterns. These patterns can be used to create groups or clusters of similar images [48–51]. A fingerprint is a pattern of minutiae, just as numbers, letters or symbols are patterns. Identification of these characters can fall within two camps: On-line and off-line character recognition. On-line character recognition uses the direction of pen strokes as they occur to assist in identification. Off-line character recognition only requires the final, static image to determine which character has been written [52]. A fingerprint image is a final, static image. For this reason, off-line character recognition tools were used in this study.

Hewavitharana [53] proposes a method of character identification based on histograms. Histograms are a graphic way of showing and reading the distribution of data. This is done by dividing the data into ranges such as the range between zero and one, or between one and two. These ranges, or 'bins,' always represent valid values that the data can take. What is then shown on the y-axis is the amount of data that falls within any given bin. The x-axis shows the ranges of the bins. In terms of character identification, the Hewavitharana method divides the character image into horizontal and vertical segments. It then counts the number of dark pixels in each segment and uses these values as the inputs to a Hidden Markov Model. The link between this and the neural network approach to fingerprint identification is given below. The exact method for generation is given below and is illustrated by Figure 2.18

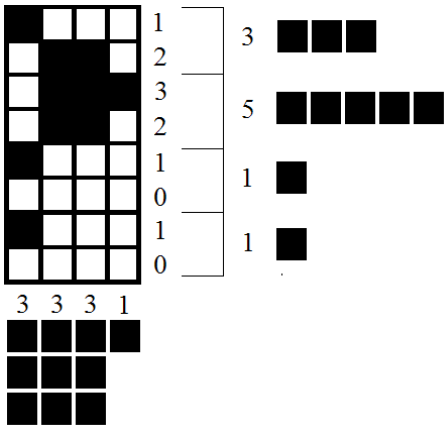


Figure 2.18: Hewavitharana Bin Construction

```
1     ybin is array of size nr bins initialised to 0 and stores the bins
2     For y = 0 to nr of bins
3         For i = 0 to (height in pixels / nr of bins)
4             For x = 0 to width in pixels
5                 If pixel(y * (height in pixels / nr of bins) + i, x)
                    is black
6                     Add 1 to ybin(y)

7     xbin is array of size nr bins initialised to 0 and stores the bins
8     For x = 0 to nr of bins
9         For j = 0 to (width in pixels / nr of bins)
10             For y = 0 to height in pixels
11                 If pixel(y, x * (width in pixels / nr of bins) + j)
                    is black
12                      Add 1 to xbin(x)
```

Märgner [54] uses a similar method to Hewavitharana. Märgner bins take not only the number of dark pixels within the segment under question, but also the number of dark pixels in the segments to either side of the original segment. These windows of segments overlap. This method has a smoothing effect on the data and allows one to add the number of dark pixels in the various segments together. This prevents jump discontinuities. Below is given the method for a window size of two and it is illustrated in Figure 2.19. As is clear, it is more complicated than the generation of Hewavitharana bins. The difference between Hewavitharana and Märgner bins is

shown in Figure 2.20. For fingerprint recognition the distances between minutiae are divided into a single histogram, in other words Hewavitharana bins, creating input for a neural network.

```
1     ybin is array of size nr bins initialised to 0 and stores the bins
2     For y = 0 to nr of bins
3         If first bin
4             For i = 0 to height in pixels / (nr of bins - 1)
5                 For x = 0 to width in pixels
6                 If pixel(i, x) is black
7                     Add 1 to ybin(0)

9         Else if last bin
10             For i = 0 to height in pixels / (nr of bins - 1)
11                 For x = 0 to width in pixels
12                 If pixel(height in pixels - i, x) is black
13                     Add 1 to ybin(nr of bins -1)

14         Else
15             For y = 1 to (nr of bins - 1)
16                 For i = 0 to 2*(height in pixels / (nr of bins - 1))
17                     For x = 0 to width in pixels
18                         If pixel (y*(height in pixels /
                                (nr of bins - 1) + i, x)
                                is black
19                                     Add 1 to ybin(y)

20     xbin is array of size nr bins initialised to 0 and stores the bins
21     For x = 0 to nr of bins
22         If first bin
23             For j = 0 to width in pixels / (nr of bins - 1)
24                 For y = 0 to height in pixels
25                 If pixel(y, j) is black
26                     Add 1 to xbin(0)
```

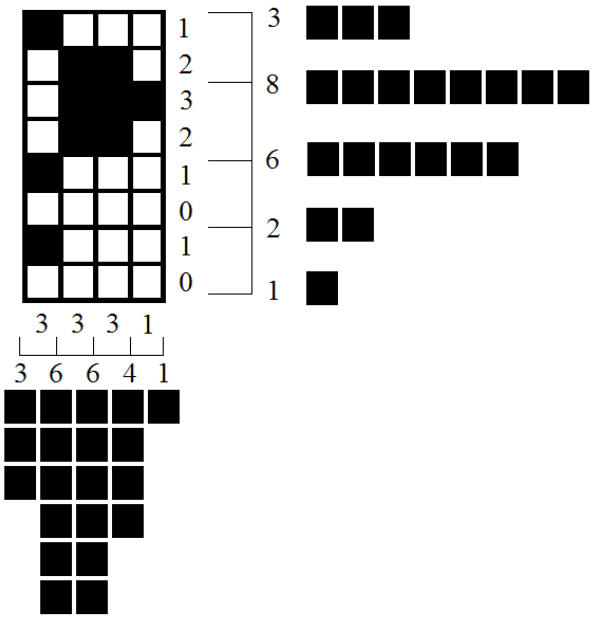


Figure 2.19: Märgner Bin Construction

```
27          Else if last bin
28              For j = 0 to width in pixels / (nr of bins - 1)
29                  For y = 0 to height in pixels
30                  If pixel(y, width in pixels - j) is black
31                      Add 1 to xbin(nr of bins -1)

32          Else
33              For x = 1 to (nr of bins - 1)
34                  For j = 0 to 2*(width in pixels / (nr of bins - 1))
35                      For y = 0 to height in pixels
36                          If pixel(y, x*(width in pixels /
                                (nr of bins - 1) + j)
                                is black
37                                  Add 1 to xbin(x)
```

Characters are identified in images by grouping them into clusters [52]. A character cluster is a set of known examples of that character such as 'A' or '3'. Fingerprints can be identified by placing them into clusters of known examples using the same method of character identification. This is because, just like specific characters, fingerprints are unique. This means that a character or fingerprint is recognised

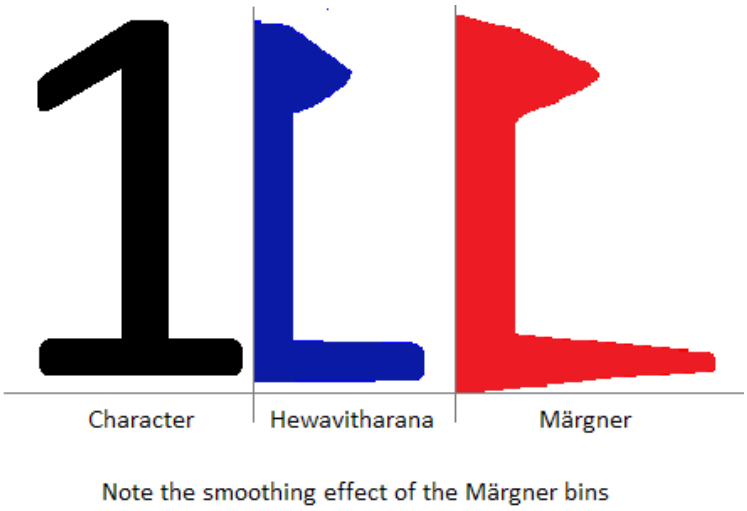


Figure 2.20: Comparison of Hewavitharana and Märgner Bins

based on how similar it is to known examples in the same cluster. However, unlike characters, it is unlikely that fingerprints can be uniquely identified by this method alone. This is based on the fact that fingerprints are very similar to one another as the small number of existing classes proves.

Instead of individual identification, the clustering of fingerprints creates a subset or cluster of individuals that have similar fingerprints [8]. In a search, only one cluster would need to be examined with a more intensive matching algorithm. Examining fingerprints within one class is likely to lead to recognition errors, as the scanned and stored images may be placed into different clusters. It is crucial, then, to establish just how far clustering can be used. This will inform how small a cluster can be made, or the requirements for fuzzy membership ability. Fuzzy membership means that an individual can belong to more than one class with different levels of membership [55].

Hidden Markov Models (HMM) have seen widespread use in a range of applications [56–60]. Included among these is character recognition [53, 54, 61, 62]. However, in this case of the HMM, a Multilayer Perceptron (MLP) can model it, based on the ideas proposed by [63]. MLPs are a type of Artificial Neural Network (ANN) [64]. ANNs have been extensively used in classification problems [65, 66]. In this study an MLP is used to perform classification to aid in identification. An MLP was chosen because it is simple to optimise both the parameters and structure simultaneously. This is described in § 2.4. In this section an introduction as to what an HMM is is given first. Therefore they can be modelled with an MLP. The section ends with

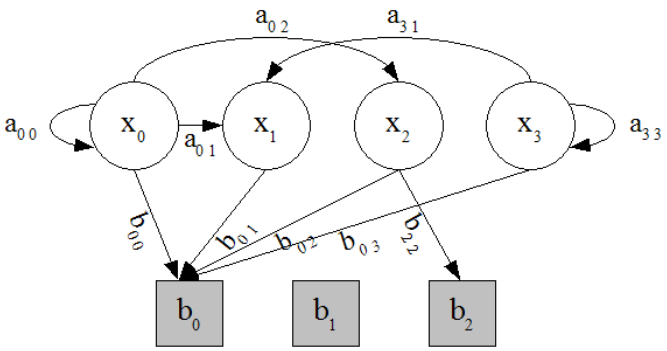


Figure 2.21: Hidden Markov Model

a description of an MLP and the details for how the MLP will be implemented for fingerprints.

An HMM is similar to a state-space model. However, the states are unknown. Only the probabilities of how the states change and the probabilities of an output, given a state, is known. The objective of the model is to predict a sequence of outputs given an initial set of states. Here we will be working with a discrete HMM.

Mathematically speaking, an HMM is described by a probability matrix Λ where a_{ij} is the probability that, given the system is in state x_i, it will move to state x_j. Λ is row stochastic, meaning that the rows all add up to one. It should be noted that i can be a vector, in which case the HMM has N states. The other part of the model is B, which is a matrix where element b_{yx} is the probability that output y will be observed given that the system is in state x. This is illustrated in Figure 2.21. It should be noted that in this diagram not all connections are shown, as this would reduce clarity. The model as a whole is generally termed λ. The probability of the initial states are often termed π. Thus, an HMM is defined as $\lambda = (\Lambda, B, \pi)$ [67].

When using an HMM a set of observations is generated. These are presented as a vector O, whose elements are labelled $o_1, o_2, \ldots, o_k$ where k is the number of observations. This is shown in Figure 2.22. The power of the HMM is threefold. First, it can generate a set of observations based on the most likely change in states. Second, it can determine the probability that a set of observations could be observed. Third, it can decode a set of observations into the most likely states that the system was in during the generation process. All three uses of an HMM require a known initial set of states.

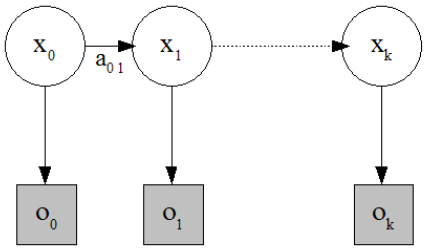


Figure 2.22: Hidden Markov Model Observations

The probability of a set of observations, $p\{O|\lambda\}$, can be found in the following manner from [67]. First, the forward variable needs to be defined. The forward variable is the probability that a partial observation, $o_1, o_2, \ldots, o_t$ will be generated. It terminates with the system in state i:

$$\alpha_t(i) = p\{o_1, o_2, \ldots, o_t, x = i|\lambda\}. \tag{2.33}$$

It is clear that using this variable the following is found

$$\alpha_{t+1}(j) = b_{\alpha_{t+1},j} \sum_{i=1}^{N} \alpha_t(i) a_{ij} \tag{2.34}$$

In other words, the possibility that the next observation will be seen with the system in state j, is the possibility that the system will move to state j from its current state, multiplied by the possibility it will give that output. This is multiplied by the chance of being in the previous state with the previous output. The probability of it being in the first state is

$$\alpha_1(\pi) = \pi b_{o_1,\pi}. \tag{2.35}$$

Thus the probability of O is found recursively by evaluating α until k is reached. This must then be summed by all the possible ways it could get to that observation so,

$$p\{O|\lambda\} = \sum_{i=1}^{N} \alpha_k(i). \tag{2.36}$$

A way to find the most likely set of states in the system is the Viterbi algorithm given in [67]. First we define $\delta_t(i)$ as the highest probability the state and observation sequence can have terminating in state i after t steps. This is written as

$$\delta_t(i) = \max_{x_1, x_2, \ldots, x_t} p\{x_1, x_2, \ldots, x_{t-1}, x_t = i, o_1, o_2, \ldots, o_{t-1}|\lambda\}. \tag{2.37}$$

Thus a recursive method for evaluating $\delta_{t+1}(j)$ would be

$$\delta_{t+1}(j) = b_{o_{t+1},j}[\max_i \delta_t(i) a_{ij}]. \tag{2.38}$$

Which means that the highest probability the state and observation sequence can have is the highest possibility of the state and observation sequence at the previous point, multiplied by the chance of emitting the correct observation at the next time step and multiplied by the possibility of transiting to the next correct state. The first δ is given by

$$\delta_1(j) = \pi b_{o_1,\pi}. \tag{2.39}$$

By keeping track of the state that was selected by the δ function at each point all the way back to x_1, the best path is determined.

Rather than using the state transitions as is, [63] propose that their modification to an HMM could be replaced with an MLP. This is not used directly in this study, but it did inspire the use of an MLP instead of an HMM, whilst using the same sort of inputs. Instead of input states for an HMM, the MLP will use Hewavitharana bins as its inputs. The reason for this change is that, in this case, an MLP runs faster than an HMM. Also MLPs are easier to implement than HMMs [63].

A perceptron is a simple example of a feed-forward ANN. The most basic form of perceptron has two layers of neurons or nodes. These are the input and the output layers. The input layer takes the inputs to the ANN and passes them to every node in the output layer. The output layer takes all the inputs and combines them with some activation function that may include a bias [66, 68]. It is the weights and biases of these activation functions that are tuned to maximise the performance of the perceptron. The MLP adds an extra layer of neurons between the input and output layers. The hidden layer also has activation functions. This allows the network to handle more complicated classification problems [66, 69]. A pictorial view of an MLP is given in Figure 2.23. For clarity, the rest of this text will use neuron for ANNs, and vertex for graphs.

The reason an MLP is called a feed-forward network is that the output relies on no prior states of the network, only the current inputs [69]. Normally an MLP output has a linear activation function. However, other functions are possible. This research uses a sigmoid activation function, given in (2.40), where x_n is the value of input n. There are N inputs and b is the bias term [70]. This makes it fall into the class of MLP known as a Radial Basis Function (RBF) neural network [64, 66].

$$y = \frac{1}{1 + \exp(b + \sum_{n=1}^{N} -x_n)}. \tag{2.40}$$

Finding the weights that will give the most accurate result is an optimisation problem. One of the most common techniques is the expectation maximisation

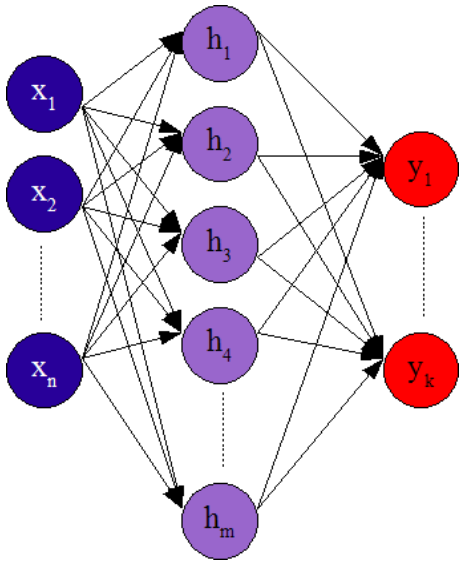


Figure 2.23: Multilayer Perceptron

algorithm. However, there are many others [71]. The reason for the use of the sigmoid activation function is that the weights and the structure can be optimised simultaneously. See § 2.4 for details of this [72, 73].

The complexity of an MLP is $O(abc)$ where a is the amount of input neurons, b is the amount of hidden neurons, and c is the amount of output neurons. After the training procedure, the number of hidden nodes is fixed. When setting up the MLP the number of input and output neurons are chosen. Thus the runtime of the MLP will be constant. The number of input and output neurons must be chosen so that the MLP does indeed reduce the time taken for identification. The number of output neurons being equal to the amount of classes. A large advantage of an MLP is that each layer can have all its neurons processed in parallel. This will greatly improve processing time.

In summary, while there are a lot of tools designed around an HMM, they can be modified to work with an MLP. Hewavitharana bins are used as the inputs for an HMM. After training, it must be established that the optimal amount of bins and classes will reduce the time taken for identification to its lowest possible value.

2.3.1 Existing Use of CI with Fingerprints

So far this section has described general techniques for the use of ANNs in classification. This subsection will focus on how they have been used previously and will also contain a brief description of some other CI based techniques used in fingerprint recognition section.

Wilson, Candela and Watson [74] pioneered the use of ANNs in identification. In their method the ANN was used for classification to reduce the search space, much as this research aims to do. In their method, direction features forms the basis for classification. Using this they obtained an accuracy of 90.2%. They achieved a processing time of 2.65s per fingerprint. Only 1ms of this was classification, the rest was feature extraction, hence this research aims to greatly reduce that figure by not including direction.

Research into the use of ANNs for use fingerprint fingerprint recognition systems continues to this day. For example ANNs can also be used directly for minutiae extraction such as in [75]. There they have trained a neural network to find minutiae in a 300x300 thinned image. The network is trained on 3x3 pixel templates that correspond to the minutiae, but once a minutiae is detected it is re-examined using a 5x5 grid to reduce the amount of spurious minutiae. The second filtering used was rejecting all minutiae detected if they are too close together. Using this they obtained an average of a 46.33% false rejection rate on poor quality images.

A recent development in classification of fingerprints is the use of Support Vector Machines (SVM). An SVM is a binary classifier, that is it places things into one of two classes. However, they have been modified in various ways to divide into multiple classes. One such way is that each class is compared to all the other classes and then the best fit is chosen to be the data's class. This was the approach used by [76]. Like [74], they used human understandable classes. By extracting a type of feature called a FingerCode [77] in addition to singularities, they achieved a classification accuracy of 90.8% when deviding into five classes and 94.9% for four classes.

HMMs have also been used extensively in fingerprint recognition systems. For instance [78] describes an HMM based classifier. It creates a set of lines across the image and where ridges meet these lines it calculates the distance to the last intersection, the angle of intersection, the change in the angle since the last intersection and the curvature of the ridge at the intersection. Dividing the data into four classes they achieved a 10.0% error rate. HMMs can also be used for matching such as in [79].

In this paper they used features based on the fingerprint texture structure, focusing on the orientation was used for matching. It broke the image up into windows in which the orientations were the observation vectors. Given the state sequence of the known fingerprint it calculates the probability of the observation vector and based on that rejects or accepts the fingerprint. Using this approach they obtained an equal error rate of 7.1%. The main advantage of this algorithm is that it does not require thinning.

This subsection is only a brief overview of some common CI techniques used in fingerprint recognition systems. Here we highlighted the basic principle of operation for some algorithms and gave their relevant results. This will be used as the benchmark for the results found in this dissertation.

2.4 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a powerful optimisation tool that was developed in the early 1990's by Eberhart and Kennedy. This section explores how it works and gives an example of its use, which was developed by the author for another application.

PSO is a population based, stochastic, evolutionary search algorithm [72, 73]. It is based on models of how birds flock. The basic form of it is known to be caught in local minima, but there are many modifications that avoid this problem; [80–82] are examples. It has been reported that PSO is faster than expectation maximisation, and has the advantage of being able to optimise the network structure of a sigmoid activated RBF [72, 73].

In [73] it is explained that PSO optimises a fitness function. Normally the fitness function is minimised and can take any mathematical form. The basic form of PSO works by having a swarm of particles. Each particle has a current position, a record of the best fitness it has achieved, and the amount of change it experienced at the last iteration which is called its velocity. The swarm keeps track of the global best fitness. At initialisation, the positions and velocities are randomised. For every particle, at each iteration, the following is done in the given order:

1. Update velocity
2. Update position

3. Find fitness
4. Update particle best
5. Update global best

At each iteration, denoted t, the velocity is updated with

$$v(t) = r_1 c_1(\rho_g - x(t-1)) + r_2 c_2(\rho_l - x(t-1)) + wv(t-1). \tag{2.41}$$

In this equation, c_1, c_2, and w are constants that are tuned. In the case of this study the guidelines in [83] were used. These are simply that c_1, c_2, and w add up to a value dependent on the application. The values r_1 and r_2 are randomly generated on the unit interval. Originally they were uniformly distributed but [84] suggests that it should be exponentially distributed. The interpretation of the equation is that the particle will move towards the global and local bests, and will continue to move in the same direction as before, which is called the particle's inertia [72, 73]. The previous bests are denoted ρ_g for the global, and ρ_l for the local. There are modifications where there is a neighbourhood, but these were not used in this study [72, 73]. It is important to note that x and v can be vectors if multiple variables are being optimised simultaneously. The position is updated with

$$x(t) = v(t) + x(t-1). \tag{2.42}$$

The global and particle (or local) bests are found by comparing the fitness of the current position of the particle with the stored value. This gives the particles some basic memory of the best positions that have been found.

In this study the Gaussian Jumping variation is used to avoid local minima. In this variation, when a particle has a low velocity for several iterations, its position "jumps". This means it is forced to a new value. This is a random number generated on a Gaussian distribution with a centre on the old value and where the deviation is a tunable parameter [80].

To tune the ANN weights using a PSO, the position (x values) of each particle of the swarm is the vector of potential weights for the ANN. The fitness function is determined by the application but normally is the sum of the error over all the training examples. The error is usually either root mean square or absolute relative error [73]. The structure of the network can be indirectly tuned. If the activation function of the network is given by (2.43) then k is an extra variable tuned by the PSO. If, after tuning, k is lower than a threshold, then the node can be removed

as its effect can be reflected by changing the bias of other nodes. If k is above a different threshold, then it can be replaced with a unit step which is computationally less expensive. These thresholds are application dependent [73]

$$y = \frac{1}{1 + \exp k(b + \sum_{n=1}^{N} -x_n)}. \tag{2.43}$$

The PSO will modify the network's weights and structure to best fit the application. In this case, it will optimise the ANN's ability to create clusters of fingerprints where all the examples of the fingerprint will be grouped together.

PSO has $O(pi)$ complexity, where p is the size of the population, and i is the amount of iterations. In the case of network training, this operation is not particularly expensive as it is simple floating point arithmetic.[8] It should also be noted that at each iteration, the entire population can be processed in parallel [85]. Since training can be done centrally, and the final network can be distributed to less powerful processors, this is an important consideration.

A complete example of a PSO application is given in appendix A. This work was originally presented at the International Conference on Swarm Intelligence, Beijing, 2010, by the author [86] of this dissertation. This is presented here in an updated format with permission.

[8] This is arithmetic on rational numbers rather than integers.

Chapter 3

Implementation and Testing

This study seeks to create a complete fingerprint recognition system. It consists of a minutiae extraction process, classifier and matcher. It should be noted that the classifier will need to be trained. All the fingerprints in a database are needed to provide a dataset for testing and development. In practice, once the database is trained it will be deployed. Once deployed, less expensive and powerful systems can be used, as they will not have to redo the training. However, a central processor will still be needed to retrain the classifier with new data and redistributed periodically. Figure 3.1 shows the flowchart of the processes that would be done centrally and which are used in this study. Figure 3.2 shows the flow diagram of what would take place on the smaller systems. Figure 3.1 and Figure 3.2 also indicate where technologies from Chapter 2 will be used.

The flow of Figure 3.2 is works as follows. All the technology proposed in this research is based on minutiae so they will need to be extracted first. Then classification happens to reduce the search space. Then the remaining fingerprints are exhaustively searched. This is only effective if the speed of the classifier is sufficient to reduce the time of the exhaustive search. In other words that time spent classifying is more than the time that would be spent searching all the other fingerprints. Another important consideration is that the classifier does not remove the correct fingerprint from the search, in other words affect the FRR. However, this last is more of a minimisation of the effect because it is impossible that it will have no effect. An exhaustive search is necessary to ensure that the fingerprint is found. The framework given in Figure 3.2 will take advantage of the speed increase of the classifier while still maintaining the strengths of an exhaustive search.

Putting the figures in terms of fingerprint terminology, fingerprint verification is done

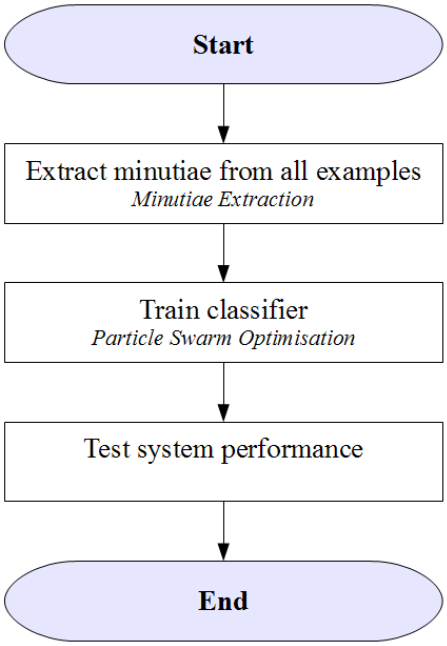


Figure 3.1: Overall Flow Diagram

using graph matching, and fingerprint identification is done by means of an Artificial Neural Network (ANN). The weights of the ANN are trained using Particle Swarm Optimisation (PSO). This all relies on the fingerprints' minutiae being extracted first. Thus the implementation and testing of the minutiae extraction is presented first, followed by graph matching, ANN and PSO.

A dataset needs to be chosen to find results. For this study the FVC2000.4 database was chosen, as the fingerprints were artificially generated [87]. The choice was made because the generation creates a known spread of quality of images. While the FVC2000.3 database would have provided a more difficult benchmark [87], its generally poor quality of images would not be an accurate reflection of the practical deployment.

As previously stated, software was provided by the CSIR [14]. It was decided that it would be prudent to use this software, rather than the unproven extraction method also presented. As a final test of the software, the minutiae of 10 test images were extracted. The image was rotated 90° and the minutiae re-extracted. In both cases the images were marked to show the location of the minutiae. The second image was then rotated and visually compared. If the minutiae were in the same position in both extractions then the software would be at least consistent, which should be sufficient for the purposes of this research.

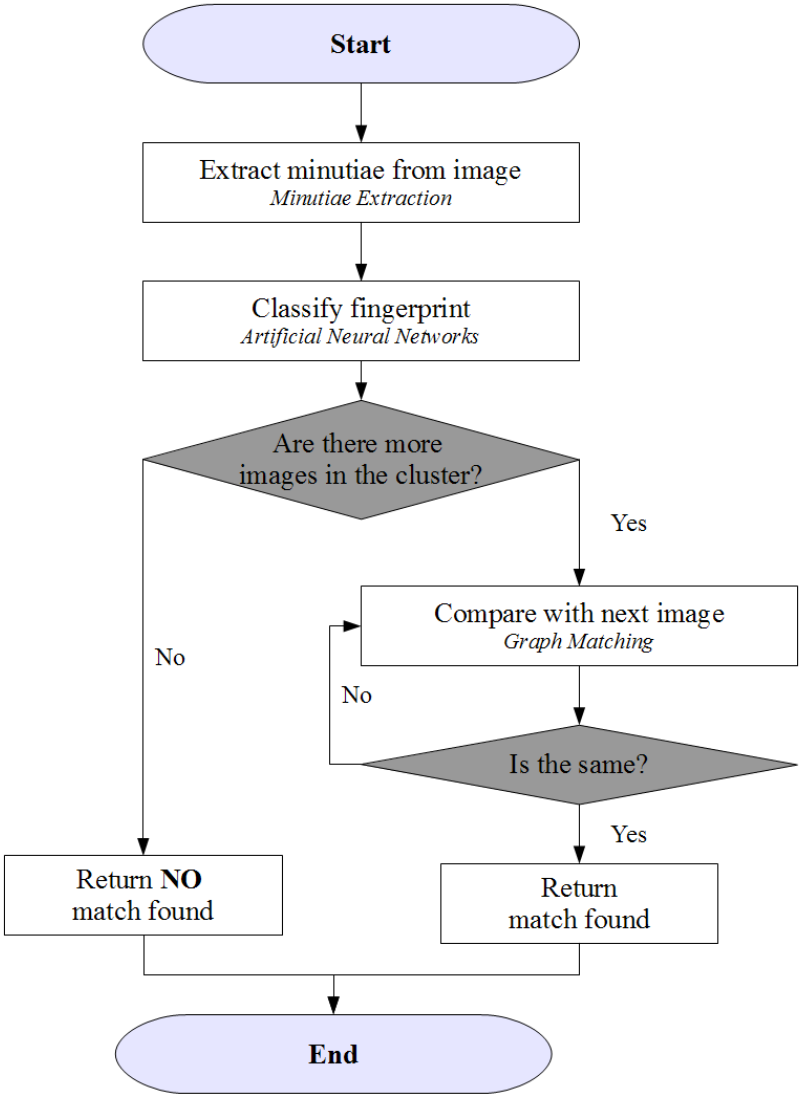


Figure 3.2: Flow Diagram of Distributed System

All implementations were done in C++. For each experiment the following was done:

1. Implement the underling technology, e.g. PSO.
2. Test this implementation.
3. Modify the implementation to be the desired component.
4. Train the component with 70% of the database.
5. Use the remaining 30% to test the performance of the component.

The initial testing of the implementation was to prove that the concepts were understood. It also gave assurance that poor results, if any, would not be attributable to bad coding.

It is clear that there are three distinct parts to this research: Extraction, classification and matching. Each can and should be tested individually. The remainder of this chapter is divided into a section for each. These sections provide both the details of the implementation, and the tests that were carried out on the components individually. The next chapter will give the results and the final chapter will explore implications.

3.1 Minutiae Extraction Algorithm

The minutiae extraction algorithm presented here has one key objective: Minimise time. To do this there are two considerations. First, no orientation or ridge frequency estimations were taken (see § 2.1). Second, processes were chosen that were parallisable. This ensures faster processing time as microprocessors are moving to multi-core designs and Graphics Processing Units (GPUs) (which have many cores already) are also being used for more general processing. To this end the following is used (flowchart given in Figure 3.3): Histogram equalisation, binarisation, cleaning, Region of Interest (ROI) limiting, thinning and location. Details are given below. The original image, used as an example for this process, is given in Figure 3.4.

Histogram equalisation aims to utilise all of the possible intensities of grey equally [88]. Fisher [88] explains that this is finding a mapping function f, which maps an input image $A(x, y)$ to $B(x, y)$ an enhanced image. The density of the intensities, D, can also be written in terms of f

$$D_B = f(D_A), \tag{3.1}$$

which is also termed the transfer law and it is assumed that it is single-valued and monotonically increasing. This allows for the inverse,

$$D_A = f^{-1}(D_B), \tag{3.2}$$

to also be true. Another property is that it will not change the area. Hence, if h is the histogram,

$$h_B(D_B) = h_A(D_A)/\frac{df(x)}{dx} \tag{3.3}$$

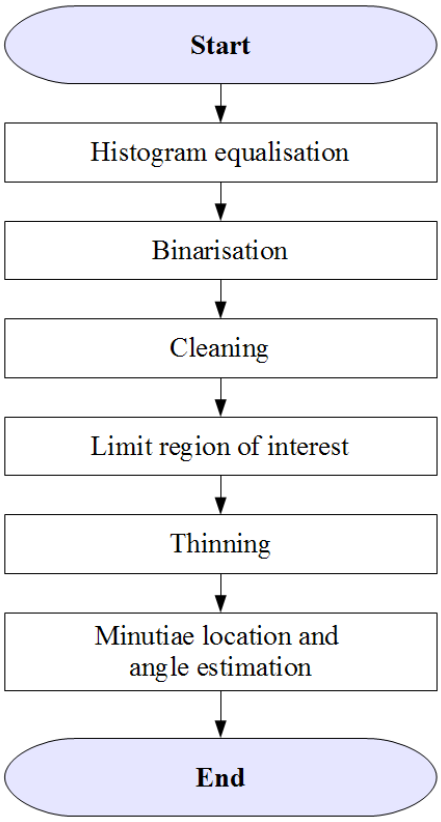


Figure 3.3: Minutiae Extraction Flow Diagram

Figure 3.4: Original Image

Figure 3.5: After Histogram Equalisation

is true. Thus if h is considered to be a continuous probability density function, p, then

$$p_B(D_B) = p_A(D_A)/\frac{df(x)}{dx}. \tag{3.4}$$

If the maximum value of D_A is D_M and the minimum level is 0 then

$$\frac{df(x)}{dx} = D_M p_A(D_A). \tag{3.5}$$

Thus

$$f(D_A) = D_M \int_0^{D_A} p_A(u)du = D_M F_A(D_A) \tag{3.6}$$

is the final result for $F_A(D_A)$ being the cumulative probability distribution. This can be approximated on a digital image with N pixels as

$$f(D_A) = \max(0, \text{round}[\frac{D_M n_k}{N^2} - 1]), \tag{3.7}$$

where n_k is the number of pixels at intensity D_A or less. In summary, the process takes all the pixels at intensity D_A on A and changes them to an intensity of $f(D_A)$ to get B.

Figure 3.5 shows the effect of histogram equalisation on the original image. What can be seen is that the image becomes lighter, but more importantly the fingerprint ridges become much more defined.

The method used for binarisation is as follows:

Figure 3.6: Binarised Image

1. Find the mean intensity. One would expect this to be 0.5 (where zero is black and one is white) after the histogram equalisation but the fact that a discrete approximation was used can change it.

2. Divide the mean by a scaling factor to get the threshold value. Empirically the scaling factor was found to be 0.7.

3. Set all pixels with a lower intensity than the threshold to black and all others to white.

The effect is shown in Figure 3.6. Clearly this is an improved, enhanced, black and white representation of the original fingerprint.

Here cleaning the image means removing isolated pixels. Our approach was as follows: First the number of white neighbours for each pixel is counted. Then, if there are white pixels with entirely black neighbours, they are set to black. Similarly, if there are black pixels with only white neighbours they are set to white. Also, if they are in an 'H' configuration, the interlinking pixel is removed. This last step was intended to able separate ridges that are close together but which are not the same.

The thinning process takes time. To improve performance, it is important to limit the area in which it is done. To this end, the Region of Interest (ROI) is introduced. This is analogous to the segmentation process described in § 2.1. The process by

Figure 3.7: Region of Interest

which the ROI was limited is as follows (this assumes that the coordinates run from (0,0) in the top left corner):

```
1 For n = 1/2 image height to 0
2     Count amount of white on row n
3     If the amount of white is below a threshold
4         Set top limit to n
5         Break
6 Repeat For n = 1/2 image height to image height to find bottom limit

7 For k = 1/2 image width to 0
8     Count amount of white on column k
9     If the amount of white is below a threshold
10         Set left limit to k
11         Break
12 Repeat For k = 1/2 image width to image width to find right limit
```

The effect of the cleaning and ROI limiting is shown in Figure 3.7. In this diagram the borders are shown by the yellow lines. This image shows the crispness that the cleaning process gives. It also shows that when using this ROI limiting technique the ROI contains mostly actual fingerprint ridges and the majority of the fingerprint is in the ROI.

In 2003 [89] proposed what the authors called the "improved parallel thinning

Figure 3.8: Thinned Image

algorithm." It is so named because it is based on a family of algorithms that can be processed in parallel, thus making it a logical choice for this application. While it is intended for character recognition, it was thought that since it is fast and the authors claim that it would not break connected lines it would be suitable for fingerprints. This last point is important as ridges must not be broken due to thinning. The algorithm works by first counting the amount of neighbours (in this case white pixels) each white pixel has. A set of 3x3 pixel templates exist for each number of neighbours of the pixel under inspection. If the pixel and its neighbours match a template then that pixel becomes a candidate for deletion. At this point the candidate pixels are compared against a set of preservation templates, these are used to ensure that the continuity of lines will not be broken. All the pixels are processed in parallel. The process is iterated until there are no more deletions. The effect of the thinning algorithm is given in Figure 3.8. It is limited to the ROI. The templates used are given in [89].

The minutiae are located as per the process described in § 2.1. However, to estimate the angles a 5x5 window is used. In the case of truncations, the 'exit' pixel determines the angle. In the case of bifurcations, the largest angle is found then the two exit points determine the angle. This has the effect of only allowing for a limited set of angles. This loss of information is the trade-off required for improved speed. If there is an incorrect amount of exit points the minutiae are rejected as being false.

This is found to improve performance.

Each component is tested visually as they are implemented. The final algorithm is tested visually. Then its results are compared directly to the minutiae that the CSIR software locates. The visual test means that the researchers manually locate all the minutiae in 10 images and it is checked that the method will find these minutiae. The ratio of spurious (false) minutiae to genuine ones is an important measure that needs to be considered. Apart from their ability to locate minutiae, the speed of the CSIR software and the newly developed method are compared to give an indication if the trade-offs are valuable.

The method given above is simple to understand and to implement. While it is expected that there will be a loss in accuracy, it is hoped that the increase in speed will still make it viable. To this end a comprehensive set of tests will be performed to establish the accuracy, reliability, and increase in speed, compared to established software and humans.

3.2 Graduated Assignment Algorithm for Fingerprint Matching

The first component that will be covered is fingerprint verification. To reiterate, this was done using graph matching. Specifically, the graduated assignment for graph-matching algorithm. This section seeks to explain the details as to how it was implemented for the fingerprint verification problem. It also explains the tests for the implementation and then its performance in the application.

The graduated assignment algorithm for graph matching relies on the use of the 'adjacency matrix.' In the case of the fingerprint graphs this was generated using the distances between the minutiae (see § 2.2). However, to save memory use and to increase computational efficiency, distance was found using (3.8). The absolute function requires less memory because the square will often increase the number of digits required, thus increasing the memory requirements. Also, it was found, through writing code to count clock cycles, that it was faster to calculate the l_1 norm than the l_2 norm, the latter involving square roots. The l_1 norm, on a plane, is defined as

$$l_{ij} = |y_i - y_j| + |x_i - x_j|. \tag{3.8}$$

Table 3.1: Graduated Assignment for Graph-matching Parameters

β_0	Starting value of the β in the softmax assignment
β_r	Ratio by which β increases between iterations
β_f	The final value of β this is the termination condition of the algorithm
I_0	The maximum number of iterations for each β
I_1	The maximum number of iterations during the normalisation process

The graduated assignment for graph-matching algorithm relies on certain parameters. They are given in Table 3.1 [20]. During the implementation and the final tests these needed to be tuned. For the purposes of this study, the tuning was done empirically.

To test the implementation a series of images' adjacency, matrices were changed by a known match matrix by finding

$$X' = PXP^T, \tag{3.9}$$

where X is the original image's adjacency matrix, X' is the new matrix and P is a permutation matrix. As this would mean there is no noise, the program should find the match matrix M such that $M = P$ with 100% accuracy. The parameters of the graduated assignment implementation, given in Table 3.1, are then found empirically. They are chosen to achieve perfect recovery of P in a reasonable amount of time. First the minimum number of iteration was found that would consistantly find the match matrix. It was doubled for future use. This was because it was thought that, based on the results reported in [20], it would take longer when doing real comparisons for which there would be noise. The test was conducted and each test was conducted with a different image selected at random from FVC2000.4.

Noise in the fingerprint images could be from, for example, the slight differences in the extraction from one image to another, and changes in the placement of dust and grease on the scanner. Another important source of noise is caused by different angles at which the fingerprint was scanned.

The next experiment was to test the implementation of the program's ability to resist noise. This should have been in-line with the noise tolerance shown in § 2.2. To test the robustness the images' matrices were subjected to

$$X' = PXP^T + \epsilon \tag{3.10}$$

where ϵ was a noise matrix, and X, X' and P are defined as above. The noise matrix had the same dimensions as the X matrix. Its values were randomly generated on a uniform distribution that had a maximum value of a percentage of the maximum value of X. As explained in § 2.2, X was scaled to the unit interval. Thus ϵ had a maximum value related to one. For the test, the values were run from a zero to two in 0.1 jumps. The test was conducted and each test was conducted with a different image selected at random from FVC2000.4.

To differentiate between a match and a non-match, the objective function of the graduated assignment algorithm (2.11) was evaluated. If the fitness was high, i.e. (2.11) was low, then it was a match, otherwise it was a non-match. This threshold is another parameter that was found empirically.

There are four important measures: False matches, false non-matches, true matches, and true non-matches. Often these are called False Acceptance Ratio (FAR), False Rejection Ratio (FRR), true acceptance, and true rejection respectively [8]. They are commonly expressed as percentages of the test samples. These are all determined by the threshold value set for the objective function and, as such, can be expressed in terms of one another. They are the method by which various algorithms for matching are compared. Thus finding them using FVC2000.4 forms the basis of the testing. The results can then be compared to what is found in industry.

Another important test is to determine the time taken to find the result. If this is too long, even if it is able to separate matches from non-matches, then it is still not a viable solution to the verification problem. If it were a middle ranged value then it could be used for forensics and if it were fast then it could be used for access control as well. Every possible match was computed to keep the test fair as many non-matches as matches were done. The non-matches were chosen completely at random. It was ensured that the same pair of non-matches would not be computed twice.

As has already been stated, there is a need to test the effects of including unary attributes. The first part of it is to choose the attributes that will be included, the second to assign them a weight. The two attributes chosen were:

- The type of minutia: Bifurcation or truncation
- The angle the minutia formed relative to the ridge

While other attributes could have been chosen, such as the quality of the image at

the point of the minutiae (another common feature [11]), the code provided by the CSIR software and the method described in § 3.1 only finds these attributes.

This leads to two comparability functions, $d_1(G_a, g_i)$ and $d_2(G_a, g_i)$. Where $d_1(G_a, g_i)$ given in (3.11) is the comparability of the type of minutiae which has a binary output. The other, which is $d_2(G - a, g_i)$ given in (3.12), computes the comparability of the angles. In these equations, α_1 and α_2 are positive scaling factors that must be tuned. For the purposes of this study they were tuned empirically as part of the process of tuning the parameters of the algorithm.

$$d'\ (G_a, g_i) = \begin{cases} 0 & \text{if type } G_a \neq g_i \\ \alpha_1 & \text{if type } G_a \text{ and } g_i \text{ is same} \end{cases} \tag{3.11}$$

$$d''\ (G_a, g_i) = -\alpha_2 |\angle\ G_a - \angle g_i| \tag{3.12}$$

This means that during the running of the algorithm the update equation changes such that

$$Q_{ai} = \sum_{b=1}^{A} \sum_{j=1}^{I} M_{bj} C_{aibj} + d'(G_a, g_i) + d''(G_a, g_i). \tag{3.13}$$

This is instead of (2.29). The final fitness is

$$E_{wg} = \sum_{a=1}^{A} \sum_{i=1}^{I} d'(G_a, g_i) + d''(G_a, g_i) + \left(\sum_{b=1}^{A} \sum_{j=1}^{I} M_{ai} M_{bj} C_{aibj} \right). \tag{3.14}$$

This is the specific form of (2.30).

During the tuning process, the value that was minimised was the Equal Error Rate (EER). This is the error rate for the threshold when the FAR and FRR are the same. This provides a simple, single reference for the purposes of tuning the parameters. However the parameters also have an effect on the time taken for the program to run. As such, both of these values are plotted and reasonable points of operation were chosen to provide a balance of speed and accuracy. It is assumed that the values' effects on the algorithm are independent of one another. While this was not strictly true, the error in the effects was minimal.

This section explained how the adjacency matrix is generated based on the minutiae, which is the basis for the graph matching. The list of variables, that will need to be tuned empirically, was also presented. Then the tests that will be used for both the implementation and performance of the system were explained. The unary attributes that can be included were explored. The effect on the performance when including unary attributes will be tested in the same manner as the original performance tests.

Table 3.2: XOR Truth Table

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

3.3 Artificial Neural Network Classifier

This section covers the method for solving the fingerprint classification problem. Namely the use of a PSO-trained and structured MLP. First, a brief overview of the tests used to check the implementation is given. This is followed by how the MLP was configured to solve the classification problem. The types of tests on the performance are explained.

Implementation testing for the ANN was done with the Exclusive-Or (XOR) problem. The XOR logic gate is not linearly separable. This makes it a good test [73]. Table 3.2 shows the truth table for the XOR function. In this case x_1 and x_2 are the binary inputs and y is the binary output. The ANN had to achieve a 100% accuracy on this problem to be considered correctly programmed. This concurrently tests the PSO as if the network cannot seperate the problems, then it could be that the PSO may not have worked. Only if both are working will the network give the correct output.

As has already been stated in Chapter 2, Hewavitharana bins are the inputs to the ANN. For the network to give the best possible results, the starting number of hidden nodes is triple the amount of input neurons. The number of input neurons was increased in jumps of two from five to 15. However, if all the neurons are active then the number will be increased. There are many other ways that could have been used to select the network structure, such as [90], but PSO is fast, simple, and reliable [73].

The PSO has its own parameters and these need to be tuned. In the case of this research they were tuned empirically. The parameters are the population size, the maximum number of iterations and the scaling constants for the velocities c_1, c_2, and w, as described in § 2.4. The PSO will be able to find the k and b terms in (2.43). As was explained in § 2.4, the k values will dictate the structure of the network.

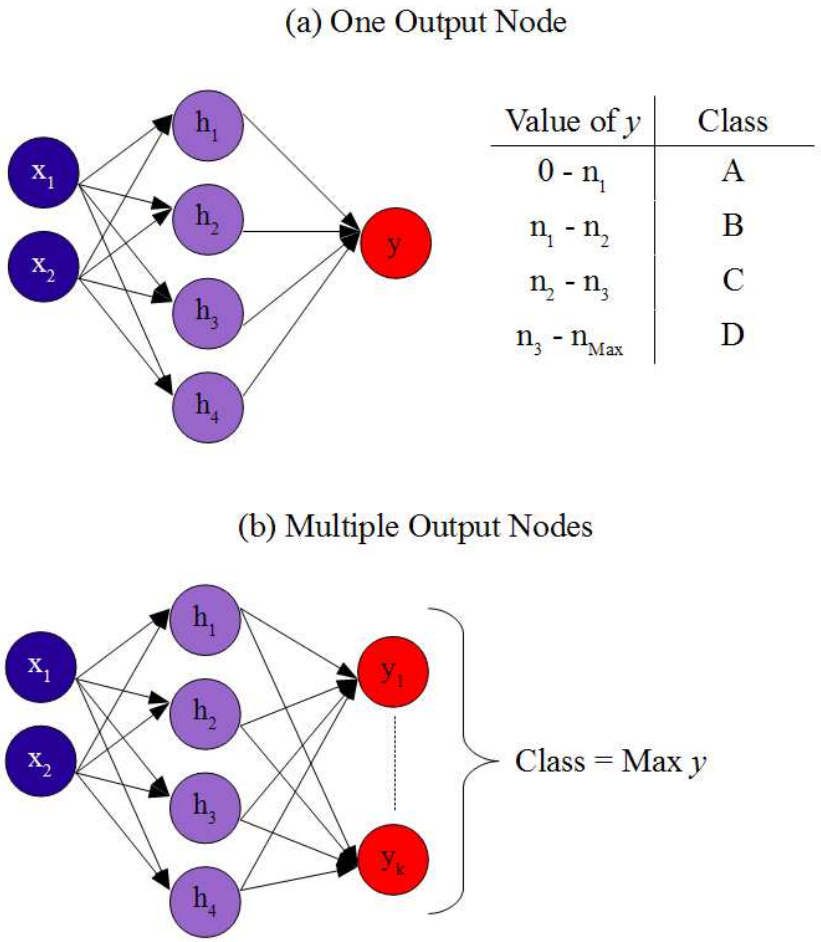


Figure 3.9: Two Options of Generating Classes

The b terms are the network weights and tuning them trains the network to give the highest possible accuracy. A single particle in the swarm contains a complete set of k and b values for all the neurons.

There are two ways the classifier can show which class a fingerprint belongs to. The first is to have one output neuron and its value gives the class. The second is to have multiple output neurons (each one representing a class), then a heuristic (such as which has the highest value) selects the final class. In this case each output node would shows the degree of membership to that class. The two options are illustrated in Figure 3.9. In the second case, the number of output neurons can be slowly increased until the classification accuracy becomes unacceptable for the particular application.

Accuracy is determined by how many times the network places the same fingerprint

in the same class. Accuracy will decrease because the size of the clusters will shrink. This is due to the smaller size of the ranges being more likely to split one person into multiple classes if the first option is used. If the second option is used, it should be noted that the PSO will also restructure the network so the number of hidden neurons, in addition to the number of output neurons, will change. Hence the performance will change. The PSO fitness function must seek to maximise the accuracy and so it is

$$f = \alpha(\text{Nr. fingerprints not in 1 class}). \tag{3.15}$$

In this equation α is a tunable parameter. This function needs to be minimised.

The classifier has two other criteria on which it must be evaluated. The first is how equally across the classes the fingerprints are spread. Ideally, every person will be in their own cluster. The expectation is that this will not be possible. If there are different groups of similar people then the classifier will be able to split these groups up. If, however, everyone is too similar, then they would all be placed in the same class. This would not be useful as the classifier will not improve the time to identification. Because of this criterion for success, the fitness function of the PSO needs a second term, the spread of the amount of people in each class. Thus what (3.15) is trying to minimise is

$$f = \alpha(\text{Nr. fingerprints not in 1 class}) + \beta(\text{variance fingerprints per class}), \tag{3.16}$$

where β is a second tunable weight. Both α and β are evaluated empirically.

The second criterion that will need to be evaluated is the time to classification. If the classifier runs slowly, it will not improve the speed of identification. Clearly all the criteria for success are linked. However, it is the implementation that will have a greater effect on the speed than the optimisation. Thus it is ignored when creating the PSO fitness function.

The design of the ANN is presented. The tests of the accuracy, spread and speed of the implementation of the ANN will give the usefulness of the classifier. The PSO will maximise this usefulness. The FVC2000.4 database is used to provide a large dataset on which to train and test the system. The results should be an accurate indication as to the performance of the system. The database is split so that 77% of the subjects (85 people) are chosen at random from the training set. Of that 30% are used for verification. The remaining 23% are the test set.

In this section the details of the implementation of both the MLP and the PSO were given. This contains a reminder as to how the PSO affects the design of the

structure of the MLP. Also described is the derivation of the fitness function for the PSO. This was done by examining what will make the classifier useful. To train and test the classifier the FVC2000.4 database was used.

Chapter 4

Results and Analysis

When proposing any new methods to solve a problem, the crux of the matter is the performance of the new method. Criteria for judging the performance could be speed or accuracy. This chapter presents the results of the tests described in Chapter 3. Based on these results, an assessment of the utility of the methods is given. This entails a detailed analysis of the results and a comparison to expectations and other solutions. First the CSIR software needs to be evaluated, as the minutiae which were extracted using this software are the basis for the other component tests. The novel approach to minutiae extraction, the fingerprint verification system and the fingerprint classifier will all be subjected to scrutiny in their own sections.

The CSIR software was evaluated visually. The minutiae on the images were marked by the software. The original and rotated images were printed out and compared. A small section of one image is shown for illustration in Figure 4.1. As can be seen, the software places the minutiae in the same area, though not in the exact same place. Sometimes the rotation has the effect of changing the type of minutiae. As this research aims to build a system that does not rely on the type of minutiae, this should not pose a problem. The images show that, while the software is not perfect, it is suitable for this study.

The rest of the chapter is divided into sections for each part of the study. Each section first gives the results, then the analysis thereof. The first section is the minutiae extraction algorithm. Despite not being used for the remainder of the study, it is important to evaluate its performance objectively and compared to the CSIR software. The second section is an evaluation of the graduated assignment for graph matching's ability to perform fingerprint verification. Finally, in the third section, the MLP classifier's usefulness is determined.

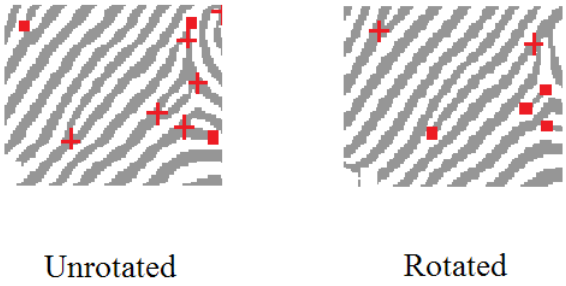


Figure 4.1: Minutiae Marked on Unrotated and Rotated Images

4.1 Minutiae Extraction Algorithm

Minutiae extraction is the first process performed by any system that will use the minutiae. Highlighted in this section are the results of the method proposed in § 3.1. First, the output is shown. This is then compared to the location of minutiae by a human user and the software provided by the CSIR. The quality of the minutiae extracted by the new method is then assessed. Finally, commentary on its usefulness is provided.

An examination of the output of the software provides valuable insight. Figure 4.2 shows a section of a fingerprint that has been extracted. In this image stars reflect bifurcations and diamonds reflect truncations. It is immediately apparent that the thinning algorithm does not protect the continuity of the ridge lines. This results in many spurious (false) minutiae being extracted. It is also clear that the effect of only extracting minutiae with the correct amount of exit points has greatly reduced the amount of false minutiae. The lines indicate the direction of the ridge. It is clear that the direction of the minutiae is not reliably correct.

Figure 4.3 is a fingerprint snippet. The red dots indicate the minutiae as they have been marked by the algorithm. The blue dots indicate the actual minutiae as detected by eye. As has been previously stated, there are many spurious minutiae. What is important to note, however, is that all the real minutiae have been found. This indicates that given the proper filtering techniques it may be possible for the algorithm to be utilised. As it currently stands, the method clearly finds too many minutiae.

The next comparison made is the new method against the software provided by the

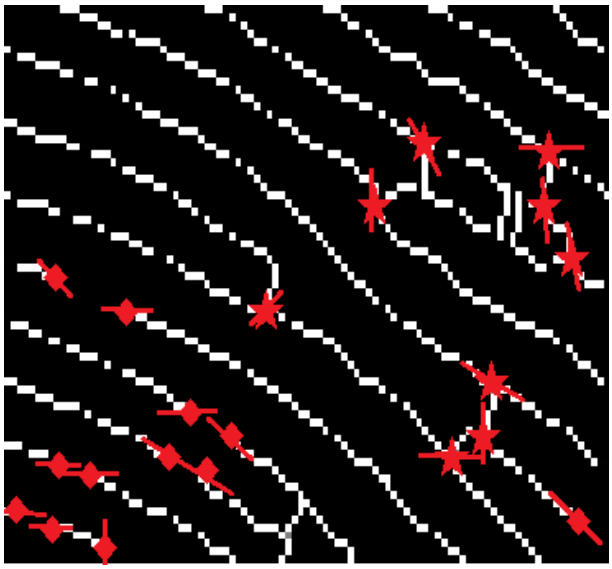


Figure 4.2: Minutiae Marked By New Method

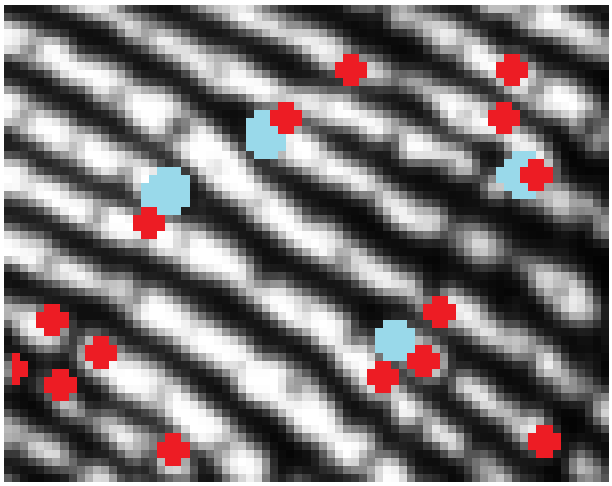


Figure 4.3: Minutiae Marked by New Method Compared to Marked by Eye

CSIR. As before, the red dots in Figure 4.4 indicate the minutiae found by the new method. The blue markings are the ones found by the CSIR. It has been shown before that the CSIR software finds spurious minutiae.

In Figure 4.4 it is clear that the new method finds more spurious minutiae than the CSIR method. In § 2.2 the minimum, maximum and mean number of minutiae extracted by the CSIR codes was 13, 127 and 34.567 respectively. For the new method, those numbers are 24, 382 and 102.607 respectively. Evidently from these numbers alone, it would seem that more spurious minutiae are extracted. However,

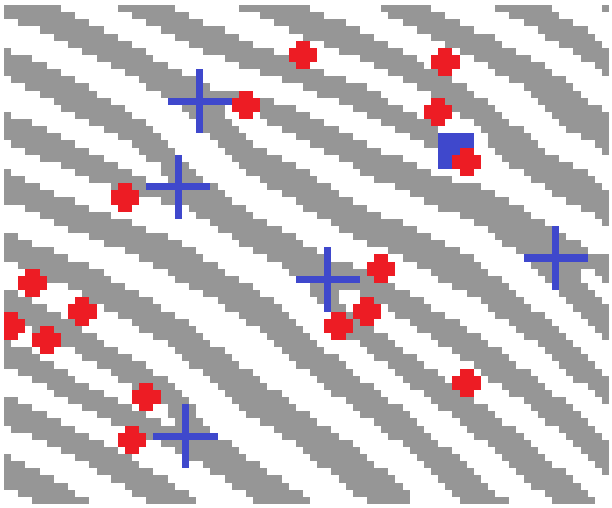


Figure 4.4: Minutiae Marked by New Method Compared to Marked by CSIR

Table 4.1: Time to Extraction on FVC2000.4

	CSIR	Novel
Minimum	1.28s	30ms
Maximum	13.15s	255ms
Mean	3.3s	102ms

based on the graphical evidence, such as that seen in Figure 4.4, the new method does find all the minutiae that the CSIR software does.

The major aim of the new method was to speed up the time taken for an extraction. To find this, a comparison is made with the CSIR software. Table 4.1 gives the results of the time trials. These numbers are generated by extracting the whole FVC2000.4 database. Looking at the table, the new extraction method did achieve its aim of speeding up the extraction time significantly. However, based on the experience of the researchers with doors requiring access control, the CSIR code takes abnormally long to perform an extraction. Exact industry standard averages were not available.

The new algorithm does achieve its aim of speeding up the time taken for an extraction. It also finds all the real minutiae in the fingerprint. However, it finds many spurious minutiae as well. The speed-up in extraction will be offset by the complexity of processing the extra minutiae. Another problem is that the algorithm does not correctly extract the angles, thus limiting the amount of information available to a matching or classification algorithm. Clearly, the shortfalls of the

Table 4.2: Parameters for Graduated Assignment Algorithm

Parameter	Value
β_0	0.5
β_r	1.075
β_f	5
I_0	100
I_1	3

method outweigh its advantages. This method will need additional work before it is feasible.

4.2 Graph-matching Based Fingerprint Verification

Verification that two fingerprints were identical was handled by the graduated assignment for graph matching. This section covers the testing of the implementation of the algorithm. The results of the tests on its effectiveness within the environ of fingerprint matching are then given. Based on these, a conclusion is drawn as to its usefulness. Following this, a similar discussion is given for when attributes are included in the graph-matching process.

The first test done was to check whether the implementation would work with a known change and no noise. This was checked over 100 different samples. The final result showed that the algorithm would find the predetermined match matrix with 100% accuracy. This was achieved with the graduate assignment for graph-matching algorithm's parameters set to the values in Table 4.2. These parameters were found empirically. This was done with the objective of minimising the amount of iterations while maintaining perfect accuracy.

The next step in the testing procedure was to check the resistance to noise. Recalling what was stated in § 3.2 this entails that in

$$X' = PXP^T + \epsilon, \tag{4.1}$$

each element of ϵ is generated randomly. These values are created on a uniform probability distribution. The maximum values increase from zero to two and the minimum from zero to negative two, to represent a signal to noise ratio that varies between 100% and 33%. At each step, the maximum value increased and the

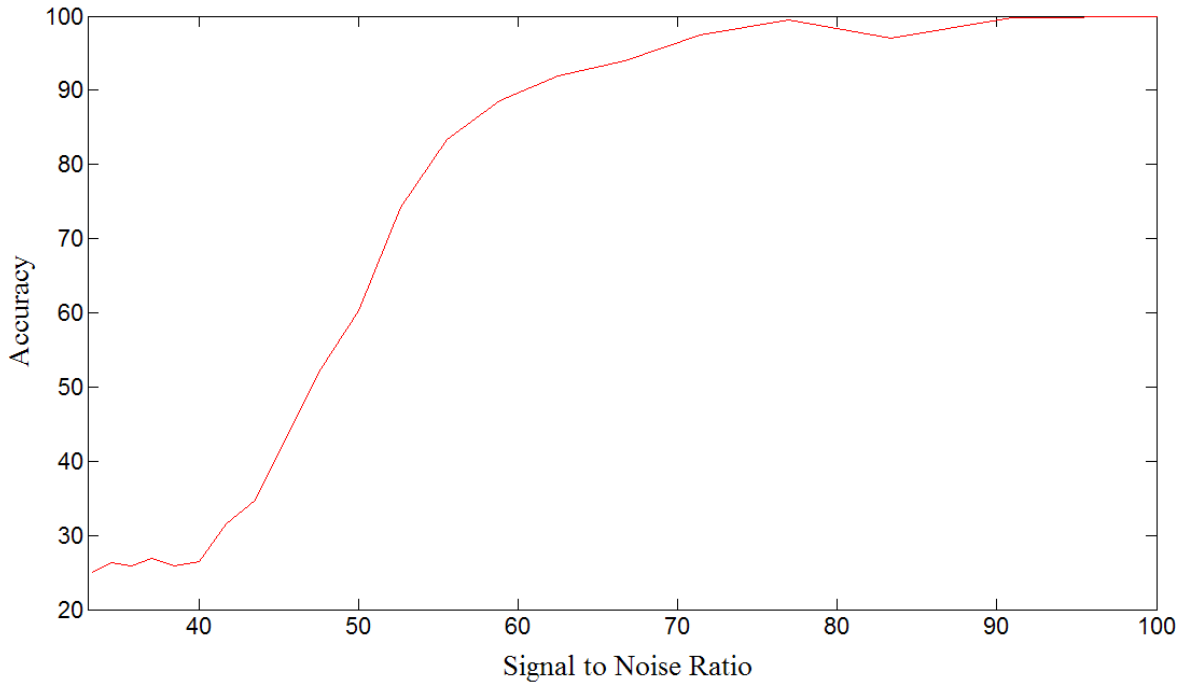


Figure 4.5: Resistance to Noise

minimum value decreased by 0.1. It was then rescaled so that the final value was on the range zero to one. Figure 4.5 shows the results of this test. Clearly the implemenation is resistant to large amounts of noise.

The most important factor for success of any fingerprint verification process is how accurately it can tell the difference between a match and a non-match. The most common measure of this is to look at the False Rejection Ratio (FRR) and False Acceptance Ratios (FAR). In the case of graduated assignment for graph matching, the measurement taken for determining the likelihood of a match is E_{wg}, defined by (2.11). The lower the value of E_{wg}, the better the quality of the match matrix and hence the more likely it is that the two graphs are subgraph-subgraph isomorphic. If E_{wg} falls below a threshold then the fingerprints are the same.

The FRR and the FAR will depend on the threshold. The lower the threshold, the smaller the number of matches declared. This implies that fewer people will be declared the same incorrectly. However, it also means that fewer people who are the same will be declared as the same. The opposite is also true: If more people are determined to be the same incorrectly, then fewer people who are the same will not be marked as the same. Figure 4.6 shows the FRR and FAR relative to the

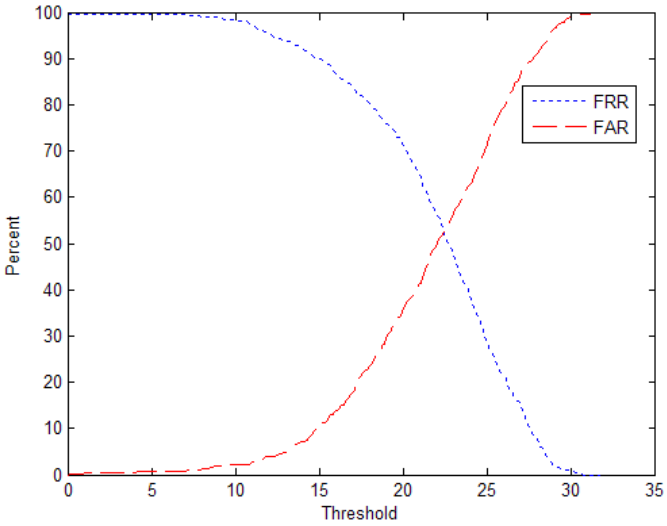


Figure 4.6: FRR and FAR vs. Threshold

Table 4.3: Key Figures for Graduated Assignment Graph-matching Approach

	Match	Non-Match
Min E_{wg}	0	5.6906
Max E_{wg}	36.7460	37.4855
Mean E_{wg}	27.9349	27.2780
Var E_{wg}	24.2802	25.6042

threshold. As can be seen the lower the one, the higher the other. This information can be placed on one plot, as seen in Figure 4.7, which shows the FAR relative to the FRR.

The figures clearly show that the ability to discern between a match and a non-match is low. This is especially true when considering the industry standard is an equal error rate (when the FAR equals the FRR) to be 15% [8]. In Figure 4.8 the values for E_{wg} are given. The blue bars represent the matches and the red, the non-matches. The diagram shows that for every value of E_{wg} there are both matches and non-matches. Though there is a tendency for the matches to have lower E_{wg} values, the difference is not great enough for this method to clearly distinguish between matches and non-matches. Table 4.3 confirms this interpretation of the diagram. The values presented in the table shows that there is not much difference between matches and non-matches.

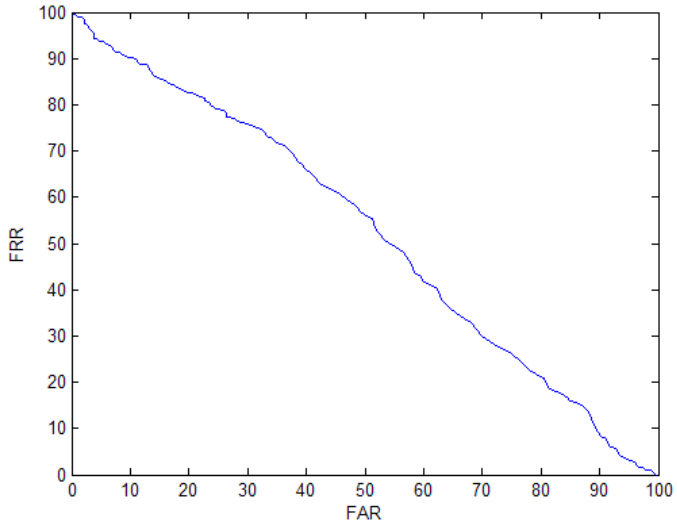


Figure 4.7: FRR vs. FAR

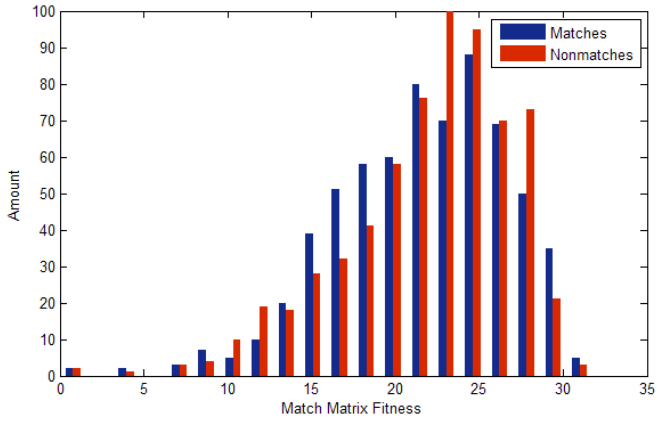


Figure 4.8: Distribution of Fitnesses

Another important criterion is the time taken to find a result. If the algorithm takes too long, it is equally as useless as if it does not find accurate results. Table 4.4 provides the execution times for both matching and non-matching fingerprints. It is immediately apparent that the time taken is too long. Existing algorithms take milliseconds to run. It is interesting to note that it will take more time to obtain a result when two fingerprints do not match. In access control this would be an even bigger problem as most attempts in an access control environment are non-matches. Graphically, the time taken with outliers is shown in Figure 4.10 and Figure 4.11 shows the detail with the outliers removed. It is clear that the bulk of the times are in the $1000ms$ range, which is unacceptably long.

Table 4.4: Times for Graduated Assignment Graph-matching Approach

	Match (ms)	Non-Match (ms)
Min time	69	66
Max time	30239	169018
Mean time	1578.3	2167.3
Var time	1.2047e07	5.9090e07

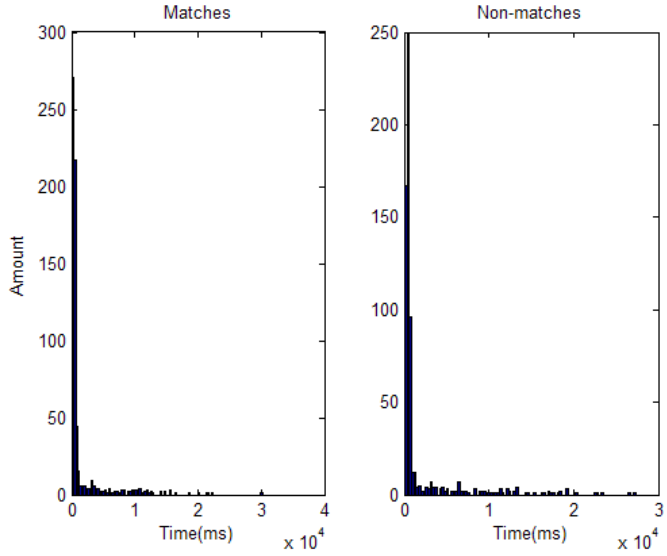


Figure 4.9: Distribution of Runtime

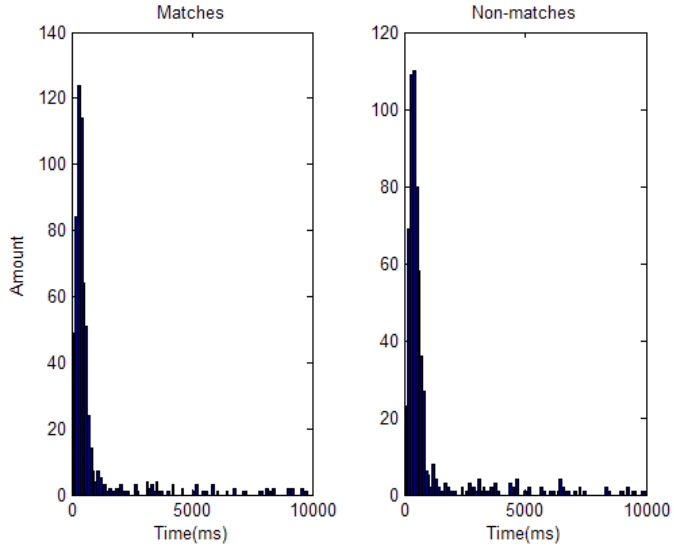


Figure 4.10: Distribution of Runtime without Outliers

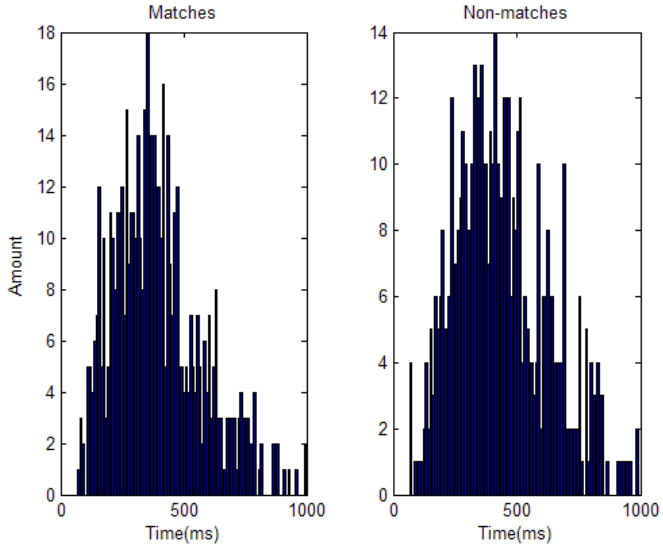


Figure 4.11: Detail of Distribution of Runtime

If the time taken to find a match was consistently low and the time taken to find a non-match was consistently high, then the execution time could be used as an indicator to find if there was a match. However, while there is a difference, it is not great enough to use for this purpose.

Neither the value of E_{wg} nor the runtime of the graph matching have sufficient difference between the values of their values of a match and the values of a non-match to be able to use them to tell the difference. Even if they were capable, the execution time is too long. These results were found when only using the distances between the minutiae as the source of information for the matching. It should still be determined if the use of the unary attributes (type and angle) can overcome these problems.

4.2.1 Graph Matching with Unary Attributes

While the objective of this research was to only use the binary attribute of the distance between the minutiae, the effect of including unary attributes was also investigated. The unary attributes that the CSIR software provides are the type of minutiae (truncation or bifurcation) and the angle of the ridge on which it falls. In § 3.2 it was discussed how they can be taken into account in the graduated assignment for graph-matching algorithm.

Table 4.5: Scaling Factors for Unary Attributes

Parameter	Value
α_1	0.5
α_2	0.07

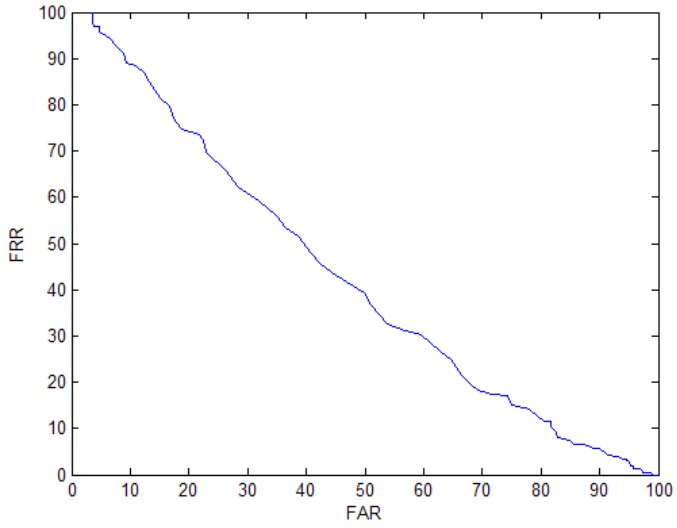


Figure 4.12: FRR vs. FAR with Unary Attributes

The two tests that need to be re-conducted with the inclusion of the unary attributes are the accuracy and the runtime tests. First the weighting of the comparability of the unary attributes needs to be tuned. These are α_1 in (3.11) and α_2 in (3.12). This was done empirically. Only the accuracy needed to be considered in the tuning, as the runtime was unaffected by the change in scaling factors. The optimal parameter values when testing on FVC2000.4 as extracted by the CSIR software are given in Table 4.5.

Figure 4.12 shows the same information as Figure 4.7, but now including the unary attributes. While there is some improvement to the ability to differentiate between non-matches and matches, it is still not enough. The equal error rate still does not approach the 15% industry standard. Like Figure 4.8, Figure 4.13 explains why there is such a poor reliability. As before, there is not enough difference in the distribution of the values of E_{wg} to make a clear distiction between matching and non-matching fingerprint pairs.

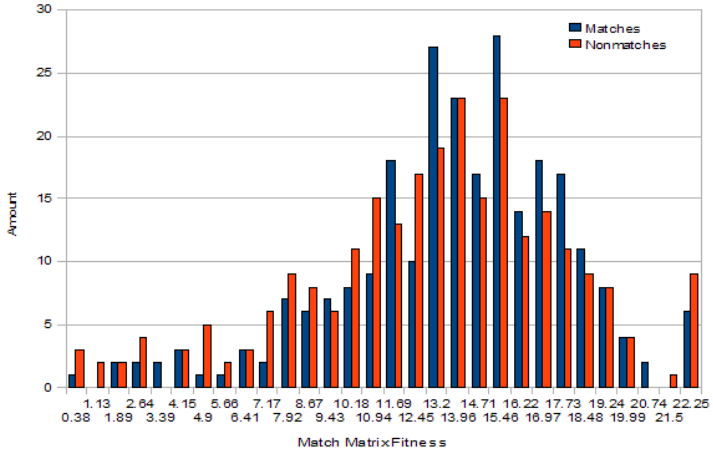


Figure 4.13: Distribution of Fitnesses with Unary Attributes

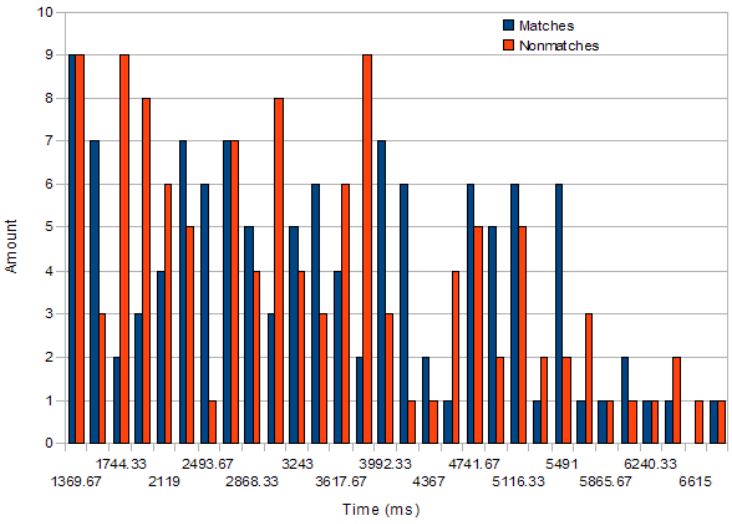


Figure 4.14: Distribution of Runtime without Outliers

The runtime is also of interest. In Figure 4.14 the distribution of runs for the test is given. In this case it is clear that the runtime is significantly longer when using unary attributes. It is also clear that there is no difference in the length of time it takes for a matching and non-matching pair.

The inclusion of the unary attributes did lead to some improvement. However, there was a large increase in the amount of time taken to find a result. Furthermore, the algorithm still could not differentiate between a match and a non-match reliably. Clearly the inclusion of unary attributes has not significantly increased the capability of the graduated assignment for graph-matching algorithm's abilities to verify that

Table 4.6: MLP Structure for X-OR Problem

Min Number of Nodes	3
Max Number of Nodes	17
Mean Number of Nodes	5.1374

two fingerprints are the same.

4.3 Artificial Neural Network Classifier

Like the graph-matching implementation, the MLP needed to be tested. This would also incorporate a test of the PSO. The system was then modified to handle the problem of fingerprint classification. In this section the results of both tests are presented. Following this an analysis of the results is given. Finally a comment on the utility of the MLP is given.

The implementation was tested on a non-linearly separable classification problem. These are more difficult than normal problems as there is no way a line can be drawn in a plot of its parameters that will separate the classes. The problem chosen was trying to correctly predict the output of an exclusive-or (XOR) logic gate.

By the definition of the problem (See § 3.3) there are two inputs and one output. The inputs and the output can either be active or inactive, this representing true or false. However, it could be coded as four inputs and two outputs. In this case each is split into two, one active if it is true, the other if the input or output is false. Combinations of these two approaches to encoding the logic state can also be considered. In this study only the first method above was checked, as they would be sufficient to show that the ANN and PSO worked.

The PSO was able to create a correctly trained ANN. This means that it gave a 100% accuracy in a test set. It was also able to reduce the number of hidden neurons in the structure from a starting point of 100 down to the region of five. Table 4.6 gives the exact results of the capabilities of the PSO to train the ANN. The details of the PSO parameters are given in Table 4.7.

While the XOR logic gate is a relatively simple problem, it can demonstrate that the MLP and PSO have been correctly implemented. For this implementation the

Table 4.7: PSO Details

Parameter	Value
Population Size	30
Max Number of Iterations	50
c_1	1.75
c_1	1.75
w	0.5

results clearly show that this is the case. This means that using the code developed already, a classifier for fingerprint identification can be built.

For fingerprint clustering, the number of input bins that were created varied from 2 to 100. For all experiments there were 1000 hidden nodes to start with, but they were reduced to remove nodes that had a low k value as described in § 2.4. In this case low was defined as less than $k < 1$. This was because it would reduce the number of nodes to the range of, at most, triple the number of input nodes, which brought the processing time to under a millisecond.

The amount of output neurons started at 2 and was increased until there was at most 110 classes. This number was chosen as there were 110 individuals in the FVC2000.4 database. However, once the accuracy became unacceptably low, this test was stopped. To be considered accurate, all the fingerprints from one individual must be placed in the same class. Thus the percentage accuracy was the percentage of individuals who have all their fingerprints in the same class. It was expected that, as the amount of classes increased, the accuracy would decrease. In the practical tests for all stated numbers of input and output neurons, the system achieved a 100% accuracy. This was because it places all fingerprints in the same class. The following was tried to change this result:

- Vary k threshold
- Change the initial number of hidden neurons to 2, 10, 50, 100, and 10 000
- Change the distributions for the initial weight estimations
- Change the parameters of the PSO

The implementation had already been proven to be correct. It also did not seem

that the choice of parameters of either the MLP or the PSO had an effect. Thus another reason was sought for the poor performance.

A possible reason is visible when viewing the source data. Figure 4.15 shows the histograms, which represent the values of the bins, from the first subject in the FVC2000.4 database. Figure 4.16 gives the histograms from the second. These are given for 10 Hewavitharana bins. Clearly there is no humanly discernible difference in the pattern between the two and the implementation of the ANN shows it is unlikely that there is a computer discernible difference. This is because the histograms change too much between different scans of the same fingerprint and change too little between two different fingerprints.

The MLP and PSO were correctly implemented for an XOR logic gate. Despite this, the system was unable to create more than one class of fingerprint. This means that no matter how fast it ran, it will not be able to decrease the time taken until identification. Thus it is not a feasible approach to classification.

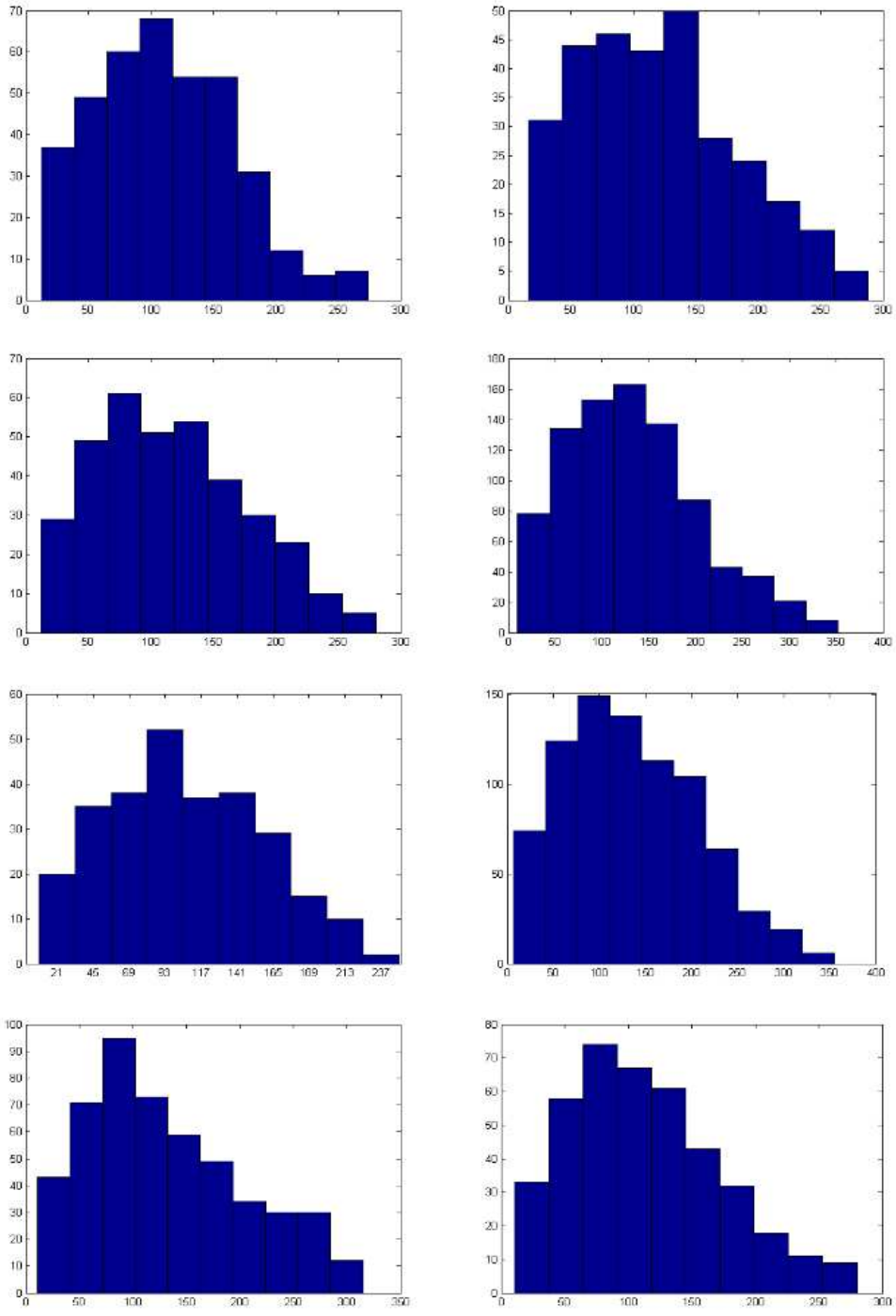


Figure 4.15: Histograms of Distances from First Fingerprint

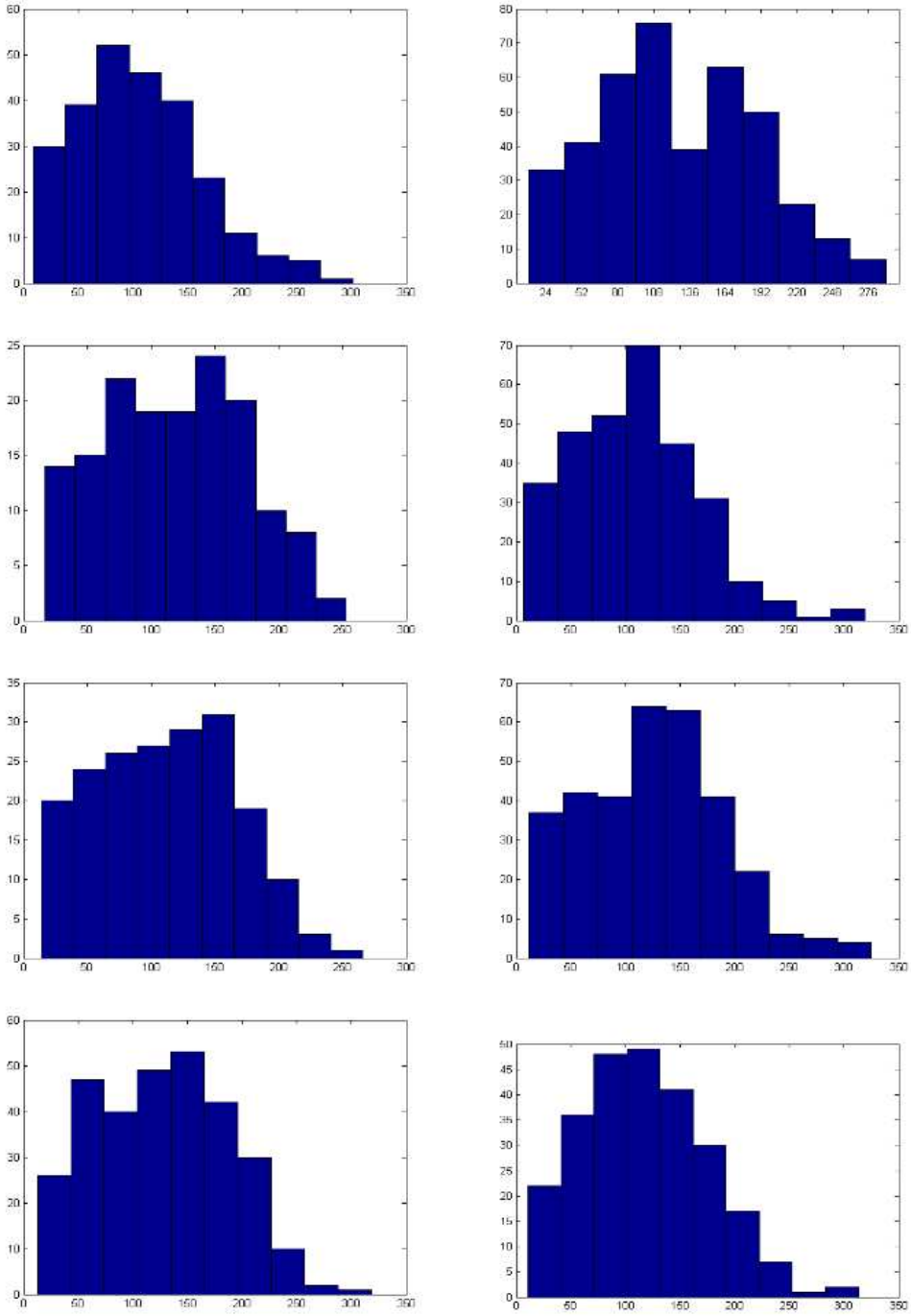


Figure 4.16: Histograms of Distances from Second Fingerprint

Chapter 5

Conclusions and Recommendations

Since the realisation of the uniqueness of a person's fingerprints in ancient times, they have been a method by which individuals have been identified. Today they are used in access control such as on doors or laptops. They are also used in forensics to find criminals and to determine all of the aliases of an individual. For both applications there are two major problems. The first is to find to whom a fingerprint belongs. The second is to ensure that two fingerprints are the same.

A common tool used to solve these problems is to locate minutiae on the ridge pattern of the fingerprint. They are where a ridge splits (bifurcation) or ends (truncation). The location of these points and the angle of the ridge is unique for each person. This study aimed to use these points for solving the problems of identification and verification of fingerprints in a new way. This new way was to only consider the distances between the minutiae as important information.

First the fingerprints had to be extracted. This was done using software provided by the Council for Scientific and Industrial Research (CSIR), South Africa. However, a new approach for extraction was also tried. This new method omitted detecting the orientation of the ridges in an effort to speed up the extraction process.

The new method was much faster than the CSIR software. However, it found many spurious (false) minutiae. As time to extraction was not important for the rest of the study, the CSIR software was used. While it too extracted false minutiae, it found fewer than the new software. Due to the number of false minutiae the new method extracted, it is not considered to be viable.

To attempt to solve the verification of fingerprints problem, fingerprints were considered to be a graph. The binary attributes of the graph were the distances between

the minutiae. The unary attributes are the type and angle of the minutiae. Finding out if two fingerprints are the same becomes the problem of determining if their graphs are sufficiently similar. This new problem is difficult to solve. However, there are methods by which it can be approximated. In this study, the graduated assignment for graph matching was used.

This method seeks to find a matrix that will show how the two graphs are related. The quality of the solution is then evaluated. This is used to determine if the fingerprints are the same. First this method was tried with only the binary attributes (distances between minutiae), then with unary attributes (type, angle) as well.

The algorithm was unable to determine if the two fingerprints were the same or not. It also took an unacceptably long time to run. Thus it is not a viable solution. The cause of this could be the spurious minutiae. It could also be that different minutiae were observed on different scans of the finger. This could have caused too much change. Another possibility is that the algorithm was too resistant to noise and would find a good match regardless.

By dividing the fingerprints up into classes, fewer fingerprints needed to be compared to find an individual. The approach proposed for fingerprint identification is based on methods from character recognition. First the distances were divided into a histogram. The size of the bins was the input for a Multilayer Perceptron (MLP) which was trained using Particle Swarm Optimisation (PSO). The output neurons of the MLP determined which class the fingerprint belonged to. Ideally each person would belong to their own class.

The MLP would only put everyone in the same class. While this maintained a 100% accuracy, it did not improve on the time taken to identify a person. Like the graph-matching problem, this could be due to the spurious minutaie, partial observability or unsuitablity of an MLP for this problem.

While several techniques were developed during the course of this research, none of them met all the criteria that they were being evaluated on. It is important to determine the source of this as they could be made to be viable with modifications. The next section seeks to provide a guide as to how this could be accomplished.

5.1 Recommendations

There are two distinct parts to the recommendations for future work. The first relates to the minutiae extraction process and the second to the fingerprint verification and identification problems.

To make the minutiae extraction process viable, the amount of spurious minutiae need to be decreased. This could possibly be done with a form of contextual filtering. It will be important to ensure that the time taken for the filtering does not increase the runtime beyond existing methods.

The fingerprint verification and identification problems have similar possible reasons for their failure. These will need to be investigated. Namely, the effect of the spurious minutiae and the partial observations. It is important to note that these reasons are not necessarily the problems for either or both. However, they are the most likely explanation.

The initial attempts proposed in this dissertation were not successful. However, it is possible that they still may be made viable. In the case of minutiae extraction, it is possible that a filter could be used. In the fingerprint identification and verification problem, the causes could be found for the poor results and thus they could be made to work. Hopefully the issues will be resolved in future work.

Appendix A

Particle Swarm Optimisation Example

At functions, events, and gatherings it is necessary to determine who will sit at the same table. There will be certain constraints such as personal dislikes or trying to keep delegates from the same organisation from sitting together. Other examples include seating people of similar ages together and ensuring that the correct number of people are seated at each table. It would be ideal if this process was automated. One technology capable of doing so is PSO. This example shows how the solution would be implemented for this problem.

The assignment problem [91] is analogous to how workers can be paired with jobs. This is a one-to-one relationship. Table allocation is similar to the assignment problem, only it is a many-to-one assignment. Also, inherent in the problem is that the costs are not related to the job (or table) but rather to the interaction of the workers. The formal definition is given in [86] and is:

> DEFINITION: To assign all people to tables in such a way as to minimize costs and satisfy rules.

[86] also stated that a "No Table" allocation can be done to represent the possibility that people can be unassigned to a table.

The greater the amount of types of costs the greater the difficulty of the problem. The use of PSO will allow for the easy modification of the cost matrix. This is due to the fact that the implementation will not change apart from the fitness function. The weighting of each cost is even simpler to change, as it is just a matter of balancing numbers. Regardless, there will not be much change to the computational efficiency, unless a new type of cost is difficult to compute.

The intuitive response would rather be to use linear or quadratic programming. However, it may be the case that the cost function is exponential, a higher order polynomial, or any other general form. PSO has an advantage in that its fitness can depend on any function. PSO has an easy to implement update, discussed below. Other population-based approaches could have been taken, such as a Genetic Algorithm (GA) [73], but a cross-over function was not readily apparent. PSO has another advantage – memory – which is advantageous in this problem.

The heuristic, known as the No Free Lunch Theorem [92], states that all optimisation techniques can be used for any given problem, however, that some are better suited to the given problem. To prove that PSO would be suited to table allocation, it is compared to random generation of solutions. To see how much the PSO will improve over this randomisation process as time progresses, it has been decided that the following would be a fair comparison: First, just as many random seating arrangements are generated as the size of the PSO's population. Then the best is chosen. The best is recorded and kept. This is one iteration. Thus, as many solutions are created for the random generation as in the PSO. Both methods are averaged over 100 trials. If the PSO cannot outperform random selection, then it cannot be considered a reasonable method.

Each particle is an entire seating arrangement, rather than a table. Each particle is initialised with a random, valid arrangement. Movement within the particles is performed by swapping people between tables. So movement towards optima is done by swapping people to be at the same place as they are in the optimum in question. Empirically, it was found that the best results were given when updating between iterations were done in this order:

- Towards Global Best
- Towards Local Best
- At random based on inertia

Better results are found when the values of each velocity are stored separately, rather than finding a joint velocity. A single movement is done by moving one person from one table to another. It could be that the capacity of the destination table will be exceeded, or the source table would have too few people after the movement. In that case, rather than a simple move, a person at the destination will be chosen at random and moved to the source table. This will maintain the capacity requirements. The detailed algorithm is as follows:

```
1 Find the amount of tables
2 Randomly initialise all particles with seating arrangement and velocity
3 Do until convergence or iteration limit
4     For all particles
5         Calculate global best, local best and inertia velocities
6         For global best velocity
7             Select a table at random
8             Select a person at the table at random
9             Find the position of person in global optimum solution
10             If that person is already at the global optimum solution
11                 Increase the fail counter
12                 If the fail counter is past its threshold
13                     Stop moving to global optimum
14                 Else find a new table and person
15             Else move the person to the table
16             If the table is now overfull
17                 Move a random person back to the first table
18         For local best velocity
19             Select a table at random
20             Select a person at the table at random
21             Find the position of person in local optimum solution
22             If that person is already at the global local solution
23                 Increase the fail counter
24                 If the fail counter is past its threshold
25                     Stop moving to loca                 Else find a
new table and person
27             Else move the person to the table
28             If the table is now overfull
29                 Move a random person back to the first table
30         For inertial velocity
31             Select a table at random
32             Select a person at that table at random
33             Select a different table at random
34             Move the person to the table
35             If the table is now overfull
36                 Move a random person back to the first table
37         Update fitness of the particle
38         Update global and local best
```

It is possible that the system will fail to find a valid move and go into an infinite loop. To prevent this scenario, failure thresholds are introduced.

Based on the example application the swarm could, within five iterations, converge on a local optimum. This occurs when initialisation of velocities is done with random numbers generated uniformly. It happened in 3% of the trails. If, however, a normal distribution with a mean of 20, and a variance of 5 is used, this problem stops. This is due to the fact that the initial speeds do not start near zero. Other distributions could have a similar effect. [84] provides a comparison of various types of distributions. It would seem that the exact values of the mean and variance are application-specific. The values given above were found empirically during the example. They gave both reasonable results and runtime.

The initialisation of the particles is embarrassingly parallel. The update is almost also embarrassingly parallel, with the exception that after each update the global solution must be shared. To accommodate this sharing, when implementing the optimisation, it is efficiently achieved by flagging particles that improve on the global optimum as candidates. The fitness of the candidates is then bubble sorted with only one iteration so the fittest is known[85]. This also makes it easy to use the multi-elitist modification to PSO proposed in[93]. Because the example is to be used as a benchmark, the modification was not implemented.

ICON and UPCON are examples of Role Playing Game (RPG) conventions held in South Africa. During test conventions there are many people that have to be assigned to a table. There are two types of participants: Player and Game Master (GM). A person may enter as either a player, GM or either. Players may also register for the convention in teams of seven people or as individuals. Each convention consists of several rounds of games.

Ideally during each round:

- Players are not placed with players they were with during a previous round
- Team mates and family members are kept apart
- There is mix of experience at a table
- There is an ideal ratio of gender
- There is an ideal ratio of players to GMs

Table A.1: Cost Table

Description	Cost
Rejoining of the players with each other	15 per Round
Rejoining of the players with a GM	35 per Round
Family members at the same table	100 per violation
Team mates (players) at the same table	300 per violation
Players at the same table as a team GM	1000 per violation
Incorrect ratios of players to a GM	15 per Player
Incorrect ratio of genders	5 per Player
Experience variation	0.05

Each criterion has a different level of importance. As the conveners learn more about the system, they may change in relation to one another. Thus, it could be represented as a linear combination, where each factor is given a weight. This can be represented as a cost matrix such as the one given in Table A.1. The experience costs represent a multiplier for the standard deviation. A negative multiplier then favours grouping players together who have similar experience.

The test conditions for the example are as follows:

1. There are 65 participants
2. 9 participants are registered as GMs, 8 male, 1 female
3. 12 participants have registered to either play or be a GM, 8 male, 4 female. During the allocation process their role becomes fixed. There are no players that are concurrently GMs.
4. There are 10 female players and 34 male players
5. There are 4 full teams of 7
6. There have been 2 previous rounds (which were allocated at random)
7. There are 4 families of which the sizes are 2, 3, 4, and 5
8. The size of the table must be between 4 to 6 players, and there must be a single GM
9. Ideally there are 6 players, 2 female, 4 male at each table
10. The experience was uniformly randomised for each player between 0 and 100

Table A.2: PSO and Random Times

	PSO	Cost
Initialisation	5.8s	5.8s
Iteration	4.15s	5.8s
Total Runtime	213.3s	290s

The characteristics of the PSO where:

- Population = 30
- Iteration limit = 50
- Global optimum scaling factor = 0.9
- Local optimum scaling factor = 0.3
- Inertial scaling factor = 0.1
- Initial speed = Normally distributed random number (mean 20; variance 5)
- Failure threshold = 50

The swarm size and iteration limit are typical of PSO [73] and kept to a reasonable run time. The scaling factors are chosen using [83] as a guide and finalised empirically. The initial speed and failure threshold is found empirically.

The trial system was run on an Intel 2GHz Core2 Duo notebook. The algorithm was implemented in Stackless Python 2.5.4. As can be seen from Table A.2, it, in fact, took longer to generate random valid solutions than to update the PSO. Thus the PSO ran in 71% of the time of the random generation.

In Figure A.1 a lower fitness is better. This diagram illustrates how the fitness improves over the running time of the algorithms. What is important to note is that the PSO outperforms and converges faster than the random generation approach.

Summarised in Table A.3 are the vital statistics. Initially they have the same fitness, but in other respects the PSO is the better performer. Normally there is a large margin of improvement. As an indication as to just how much better the PSO does, the mean final fitness of the PSO is better than the best run of the random generation. While this at first glance would seem to mean that the PSO will

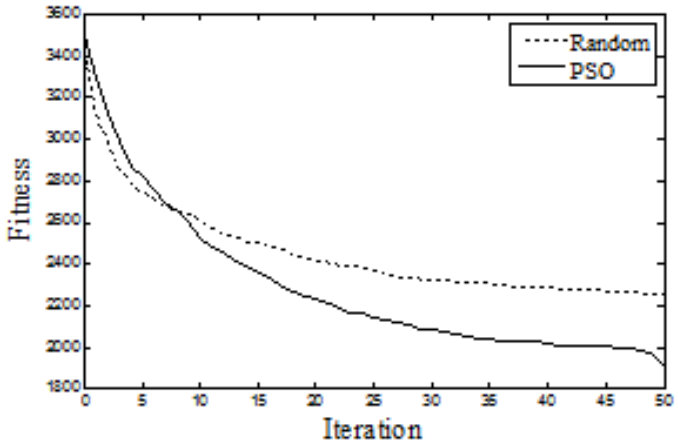


Figure A.1: Fitness vs. Iteration

Table A.3: Fitness Comparisons

	Cost	PSO
Mean Initial Fitness	3498.2	3465.43
Minimum Initial Fitness	2088	2011
Mean Final Fitness	1903.93	2243.36
Minimum Final Fitness	1201	1925
Time per Iteration	4.15s	5.8s
Total Run Time	213.3s	290s

give better results than random selection could, this is not the case. Any solution generated by the PSO could, in theory, be generated randomly.

More importantly, the histogram shown in Figure A.2 indicates the reliability of PSO over that of random generation. The majority of the PSO results are the same or better than the random approach. In fact, a mere 5% of the PSO results are worse than the bulk of the randomly generated allocations.

As a final note, the table allocation problem and the assignment problem are very similar. The Particle Swarm Optimisation (PSO) approach developed here could be adapted for the assignment problem. In this example PSO outperforms randomised results in every criterion. Thus PSO is considered to be a valid approach to solving the table allocation problem.

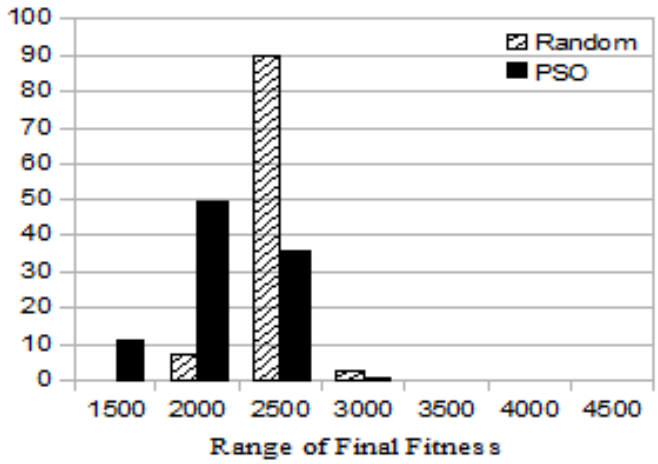


Figure A.2: Histogram of Final Fitness

References

[1] G. Biometrics, "History of Biometrics," 2008. http://www.griaulebiometrics.com/page/en-us/book/understanding-biometrics/introduction/history.

[2] R. Technologies, "Biometrics Fingerprint Technology," 2009.

[3] J. D. Woodward, N. M. Orlans, and P. T. Higgins, *Biometrics*. McGraw-Hill, 2003.

[4] H. T. F. Rhodes, *Alphonse Bertillon, Father of Scientific Detection*. Greenwood Press, 1968.

[5] enotes.com, "Vucetich, Juan," 2010. http://www.enotes.com/forensic-science/vucetich-juan.

[6] J. Leyden, "Fingerprinting of UK School Kids Causes Outcry," *The Register*, 2002.

[7] D. Holley, "Lawyer Unjustly Jailed Working Toward "Normal"," *Portland Tribune*, March 26 2009.

[8] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. Springer, second ed., 2009.

[9] E. R. Henry, *Classification and Uses of Finger Prints*. Printed for H.M. Stationery Office, by Darling & Son, Ltd., third ed., 1900.

[10] B. für Sicherheit in der Informationstechnik, "Evaluation of Fingerprint Recognition Technologies - Biofinger," 2004. Public Final Report.

[11] I. S. Orginisation, "ISO 19794 - 2: Information Technology - Biometric Data Interchange Formats - Part 2: Finger Minutiae Data," 2007.

[12] M. U. Munir and M. Y. Javed, "Fingerprint Matching Using Gabor Filters," *National Conference on Emerging Technologies*, pp. 147 – 151, 2004.

[13] J. Feng, "Combining Minutiae Descriptors for Fingerprint Matching," *Pattern Recognition*, vol. 41, no. 1, pp. 342 – 352, 2008.

[14] F. Nelwamondo. Personal Communication, CSIR, Information Security, fnelwamondo@csir.co.za.

[15] A. M. Bazen, G. T. B. Verwaiijen, S. H. Gerez, L. P. J. Veeelenturf, and B. J. van der Zwaag, "A Correlation-Based Fingerprint Verication System," *ProRISC 2000 Workshop on Circuits, Systems and Signal Processing,*, pp. 1 – 8, 2000.

[16] L. Coetzee and E. C. Botha, "Fingerprint Recognition with a Neural-Net Classifier," *Proceedings of South African Workshop on Pattern Recognition*, vol. 1, pp. 33 – 40, 1990.

[17] B. Moayer and K. Fu, "A Tree System Appoach for Fingerprint Pattern Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 3, pp. 376 – 388, 1986.

[18] M. Kawagoe and A. Tojo, "Fingerprint Pattern Classification," *Pattern Recognition*, vol. 17, pp. 295 – 303, 1984.

[19] D. Maio and D. Maltoni, "A Structural Approach to Fingerprint Classification," *Proceedings of the International Conference on Pattern Recognition*, vol. 3, pp. 578 – 585, 1996.

[20] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for Graph Matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 377 – 388, April 1996.

[21] L. Hong, Y. Wan, and A. K. Jain, "Fingerprint Image Enhancement Algorithm and Performance Evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 777 – 789, 1998.

[22] Y. Wang, J. Hu, and F. Han, "Enhanced Gradient-Based Algorithm for the Estimation of Fingerprint Orientation Fields," *Applied Mathematics and Computation*, vol. 185, pp. 823 – 833, 2007.

[23] S. M. Viola, S. L. de Oliveira Gonzaga, and A. Conci, "On The Line Width Influence in DIrectional Field Determination for Fingerprint Images," *12th International Workshop on Systems, Signals and Image Processing*, pp. 313 – 316, 2005.

[24] B. M. Mehtre and B. Chatterjee, "Segmentation of Fingerprint Images – A Composite Method," *Pattern Recognition*, vol. 22, no. 4, pp. 381 – 385, 1989.

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

[26] V. Espinosa-Duro, "Fingerprints Thinning Algorithm," *IEEE Aerospace and Electronic Systems Magazine*, vol. 18, pp. 28 – 30, September 2003. Issue: 9.

[27] D. Miao, Q. Tang, and W. Fu, "Fingerprint Minutiae Extraction Based on Principal Curves," *Pattern Recognition Letters*, vol. 28, no. 16, pp. 2184 – 2189, 2007.

[28] F. Zhao and X. Tang, "Preprocessing and Postprocessing for Skeleton-Based Fingerprint Minutiae Extraction," *Pattern Recognition*, vol. 40, no. 4, pp. 1270 – 1281, 2007.

[29] H. Fronthaler, K. Kollreider, and J. Bigun, "Local Features for Enhancement and Minutiae Extraction in Fingerprints," *Image Processing, IEEE Transactions on*, vol. 17, no. 3, pp. 354 – 363, 2008.

[30] C. C. Chen and D. C. Chen, "Multi-Resolutional Gabor Filter in Texture Analysis," *Pattern Recognition Letters*, vol. 17, no. 10, pp. 1069 – 1076, 1996.

[31] A. K. Jain and S. K. Bhattacharjee, "Address Block Location on Envelopes Using Gabor filters," *Pattern Recognition*, vol. 25, no. 12, pp. 1459 – 1477, 1992.

[32] A. K. Jain, N. K. Ratha, and S. Lakshmanan, "Object Detection Using Gabor Filters," *Pattern Recognition*, vol. 30, no. 2, pp. 295 – 309, 1997.

[33] S. Bow, *Pattern Recognition and Image Processing.* Marcel Dekker, second ed., 2002.

[34] T. Y. Young, *Handbook of Pattern Recognition and Image Processing.* Academic Press, 1994.

[35] J. A. Bondy and U. S. R. Murty, *Graph Theory.* Springer London, 2008.

[36] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty Years Of Graph Matching In Pattern Recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, pp. 265 – 298, 2004.

[37] J. R. Ullmann, "An Algorithm for Subgraph Isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31 – 42, 1976.

[38] B. D. McKay, "Practical Graph Isomorphism," *10th. Manitoba Conference on Numerical Mathematics and Computing*, vol. 30, pp. 45 – 87, 1981.

[39] M. A. van Wyk, T. S. Durrani, and B. J. van Wyk, "A RKHS Interpolator-Based Graph Matching Algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 988 – 995, 2002.

[40] M. Fernández and G. Valiente, "A Graph Distance Metric Combining Maximum Common Subgraph and Minimum Common Supergraph," *Pattern Recognition Letters*, vol. 22, pp. 753 – 758, 2001.

[41] W. D. Wallis, P. Shoubridge, M. Kraetz, and D. Ray, "Graph Distances Using Graph Union," *Pattern Recognition Letters*, vol. 22, pp. 701 – 704, 2001.

[42] M. Neuhaus and H. Bunke, "A Graph Matching Based Approach to Fingerprint Classification Using Directional Variance," *Audio- and Video-Based Biometric Person Authentication, Lecture Notes in Computer Science*, vol. 3546, pp. 455 – 501, 2005.

[43] E. W. Weisstein, "Complexity Theory," 2010. http://mathworld.wolfram.com/ComplexityTheory.html.

[44] NIST/SEMATECH e-Handbook of Statistical Methods, "Beta Distribution," August 2009. http://www.itl.nist.gov/div898/handbook/eda/section3/eda366h.htm.

[45] R. V. Hogg and E. A. Tanis, *Probability and Statistical Inference*. Prentice Hall, sixth ed., 2001.

[46] G. M. Clarke and D. Cooke, *A Basic Course in Statistics*. Arnold, fourth ed., 1998.

[47] P. Foggia, C. Sansone, and M. Vento, "A Performance Comparison of Five Algorithms for Graph Isomorphism," *3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 188 – 199, 2001.

[48] D. H. Ballard, *Computer Vision*. Prentice-Hall, 1982.

[49] R. D. Boyle and R. C. Thomas, *Computer Vision: A First Course*. Blackwell Scientific Publications, 1988.

[50] K. R. Castleman, *Digital Image Processing*. Prentice Hall, 1996.

[51] S. Watanabe, *Methodologies of Pattern Recognition*. Academic Press, 1969.

[52] R. Plamondon and S. N. Srihari, "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 63 – 84, January 2000.

[53] S. Hewavitharana, H. Fernando, and N. Kodikara, "Off-line Sinhala Handwriting Recognition Using Hidden Markov Models," *Proceedings of the Indian Conference on Computer Vision , Graphics and Image Processing*, pp. 266 – 269, 2002.

[54] V. Märgner, H. E. Abed, and M. Pechwitz, "Offline Handwritten Arabic Word Recognition Using HMM - a Character Based Approach without Explicit Segmentation," *CIFED*, 2006.

[55] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338 – 353, June 1965.

[56] S. N. Wood, *Generalized Additive Models: An Introduction with R*. Chapman & Hall, 2006.

[57] R. J. Elliott, L. Aggoun, and J. B. Moore, *Hidden Markov Models: Estimation and Control*. Springer-Verlag, 1995.

[58] A. T. Bharucha-Reid, *Elements of the Theory of Markov Processes and Their Applications*. McGraw-Hill, 1960.

[59] D. L. Isaacson and R. W. Madsen, *Markov Chains Theory and Applications*. John Wiley & Sons, 1976.

[60] I. L. MacDonald and W. Zucchini, *Hidden Markov and Other Models for Discrete-valued Time Series*. Chapman & Hall, 1997.

[61] J. Hu, M. K. Brown, and W. Turin, "HMM Based On-Line Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 1039 – 1045, October 1996.

[62] Y. Katayama, S. Uchida, and H. Sakoe, "A New HMM for On-Line Character Recognition Using Pen-Direction and Pen-Coordinate Features," *19th International Conference on Pattern Recognition*, pp. 1 – 4, 2008.

[63] H. Bourlard and C. J. Wellekens, "Links Between Markov Models and Multilayer Perceptrons," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 1167 – 1178, 1990.

[64] K. Gurney, *An Introduction to Neural Networks*. CRC Press, 1997.

[65] C. T. Leondes, *Neural Network Systems Techniques and Applications*, vol. 3: Implementation Techniques. Academic Press, 1998.

[66] R. C. Eberhart and Y. Shi, *Computational Intelligence: Concepts to Implementations*. Morgan Kaufmann, 2007.

[67] N. Warakagoda, "Hidden Markov Models," May 1996. http://jedlik.phy.bme.hu/ gerjanos/HMM/node2.html.

[68] B. Widrow and M. A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proceedings of the IEEE*, vol. 78, pp. 1415 – 1442, September 1990.

[69] C. T. Leondes, *Neural Network Systems Techniques and Applications*, vol. 1: Algorithms and Architectures. Academic Press, 1998.

[70] E. W. Weisstein, "Sigmoid Function," 2010. http://mathworld.wolfram.com/SigmoidFunction.html.

[71] C. T. Leondes, *Neural Network Systems Techniques and Applications*, vol. 2: Optimization Techniques. Academic Press, 1998.

[72] A. P. Engelbrecht, *Fundimentals of Computational Swarm Intelligence*. John Wiley & Sons Ltd, 2005.

[73] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*. Morgan Kauffmann Publishers, 2001.

[74] C. L. Wilson, G. T. Candela, and C. I. Watson, "Neural Network Fingerprint Classification," *Journal of Artifical Neural Networks*, vol. 1, no. 2, pp. 203 – 208, 1993.

[75] J. K. Gupta and R. Kumar, "An Efficient ANN Based Approach for Latent Fingerprint Matching," *International Journal of Computer Applications*, vol. 7, no. 10, pp. 18 – 21, 2010.

[76] J. H. Hong, J. K. Min, U. K. Cho, and S. B. Cho, "Fingerprint Classification Using One-vs-All Support Vector Machines Dynamically Ordered with Naïve Bayes Classifiers," *Journal of Pattern Recognition*, vol. 41, pp. 662 – 671, 2008. Issue 2.

[77] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti, "FingerCode: A Filterbank for Fingerprint Representation and Matching," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2187 – 2194, 1999.

[78] A. Senior, "A Hidden Markov Model Fingerprint Classifier," *Conference Record of the 31st Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 306 – 310, 1997.

[79] H. Guo, "A Hidden Markov Model Fingerprint Matching Approach," *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, vol. 8, pp. 5055 – 5059, 2005.

[80] R. A. Krohling and E. Mendel, "Bare Bones Particle Swarm Optimization with Gaussian or Cauchy Jumps," *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 3285 – 3291, May 2009.

[81] B. Liu, L. Wang, Y. H. Jin, F. Tang, and D. X. Huang, "Improved Particle Swarm Optimization Combined with Chaos," *Chaos, Solitons & Fractals*, vol. 25, pp. 1261 – 1271, September 2005. Issue 5.

[82] G. K. Venayagamoorthy and G. Singha, "Quantum-Inspired Evolutionary Algorithms and Binary Particle Swarm Optimization for Training MLP and SRN Neural Networks," *Journal of Computational and Theoretical Nanoscience*, vol. 2, pp. 561 – 568, December 2005.

[83] Y. Shi and R. C. Eberhart, "Parameter Selection in Particle Swarm Optimization," *Lecture Notes in Computer Science*, vol. 1447, pp. 591 – 600, 1998.

[84] R. A. Krohling and L. dos Santos Coelho, "PSO-E: Particle Swarm with Exponential Distribution," *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 1428 – 1433, July 2006.

[85] A. Hunt and D. Thomas, *The Pragmatic Programmer*. Addison-Wesley Professional, 1999.

[86] D. A. Braude and M. A. van Wyk, "PSO Applied to Table Allocation Problems," *Lecture Notes in Computer Science*, vol. 6145, pp. 206 – 213, 2010.

[87] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "FVC2000: Fingerprint Verification Competition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 402 – 412, 2002.

[88] R. Fisher, S. Perkins, A. Walker, and E. Wolfart., "Histogram Equalization," 2003. http://homepages.inf.ed.ac.uk/rbf/HIPR2/histeq.htm.

[89] L. Huang, G. Wan, and C. Liu, "An Improved Parallel Thinning Algorithm," in *Proceedings of the 7th International Conference on Document Analysis and Recognition*, pp. 780 – 783, IEEE Computer Society, 2003.

[90] D. C. Psichogios and L. H. Ungar, "SVD-NET: An Algorithm That Automatically Selects Network Structure," *IEEE Transactions of Neural Networks*, vol. 5, pp. 513 – 515, 1994. Issue: 3.

[91] D. B. Shmoys and E. Tardos, "An Approximation Algorithm for the Generalized Assignment Problem," *Mathematical Programming*, vol. 32, pp. 461 – 474, February 1993.

[92] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67 – 82, April 1997.

[93] S. Das, A. Abraham, and A. Konar, "Automatic Kernel Clustering with a Multi-Elitist Particle Swarm Optimization Algorithm," *Pattern Recognition Letters*, vol. 29, pp. 688 – 699, 2008.