Fingerprint recognition

by

Louis Coetzee

Submitted as partial fulfilment for the degree

Master of Engineering

in the Faculty Electronics and Computer Engineering

University of Pretoria

Pretoria

February 1992

Acknowledgements

I wish to thank my advisor, Liesbeth Botha for her unfailing support and expert guidance and Etienne Barnard for the many fruitful technical discussions. Finally I wish to thank my parents. Without their support and understanding this work would never have reached it's completion. Thanks.

Abstract

In this thesis methods of automatic fingerprint recognition are investigated. An introduction to fingerprint recognition as well as previous research done in this field are presented. This is followed by descriptions of the approaches we have followed while investigating the subject. These include a feature-based approach and a correlation-based approach. Two methods of data generation are discussed. The first is the well known paper-and-ink method, while the second is an optical data generation method which overcomes many of the problems of the paper-and-ink method.

Preprocessing algorithms which reduce the effect of poor quality data and which subsequently improve classifier performance were developed and are presented. The effect of these algorithms on classifier performance in the feature-based and correlation-based approach is measured and presented.

In the feature-based approach features are extracted from fingerprint images (e.g. the directional image, which is a numerical description of the fingerprint) or transformations of the images (e.g. the Fourier transform) to use with various classifiers. These classifiers include the neural-net, linear and nearest-neighbour classifiers. 100% classification performance was obtained from features extracted from the frequency domain of smoothed binary images. The best classification performance obtained from features extracted from the directional image was 90% using the neural-net classifier.

In the correlation-based approach fingerprint images are classified using a correlation classifier, without any feature extraction. The best classification result obtained using the correlation classifier and preprocessed images was 95%.

We show simulation results obtained for both approaches. From these results the best method of fingerprint recognition for this type of data is identified and we conclude with a discussion of possible extensions to some of the approaches.

Uittreksel

In hierdie verhandeling word outomatiese vingerafdruk herkenningsmetodes ondersoek en beskryf. 'n Inleiding tot die veld van vingerafdruk herkenning sowel as 'n beskrywing van werk wat voorheen gedoen is, word verskaf. Daarna word die twee benaderings wat ons ondersoek het beskryf, naamlik 'n kenmerk-gebaseerde benadering en 'n korrelasie-gebaseerde benadering. Twee metodes van data-generasie word beskryf. Die eerste metode is die bekende papier-en-ink metode, terwyl die tweede 'n optiese datagenerasie metode is, wat baie van die probleme wat bestaan met die papier-en-ink metode oorkom.

Voorverwerkingsalgoritmes wat die effek van swak gehalte data verminder en dus klassifikasie resultate verbeter is ontwikkel en word beskryf. Die invloed van hierdie algoritmes word gemeet en verduidelik.

In die kenmerk-gebaseerde benadering word eienskappe uit die vingerafdruk (bv. 'n rigtingbeeld, wat 'n numeriese beskrywing van die vingerafdruk is) en uit transformasies van die vingerafdruk (bv. die Fourier transformasie) onttrek vir gebruik met verskeie klassifiseerders. Hierdie klassifiseerders is onder andere die neurale-netwerk klassifiseerder, 'n lineêre klassifiseerder, asook 'n naasliggings klassifiseerder. 100% toets klassifikasie is verkry van eienskappe wat onttrek is uit die frekwensievlak van vergladde binêre vingerafdrukke. Die beste klassifikasie wat verkry is deur eienskappe van die rigtingbeeld te gebruik, was 90%. (Dit is verkry deur 'n neurale-netwerk klassifiseerder te gebruik.)

In die korrelasie benadering word vingerafdrukke geklassifiseer met 'n korrelasie klassifiseerder, sonder dat eienskappe uit die vingerafdruk onttrek word. Die beste vertoning verkry met hierdie benadering was 95%.

Resultate verkry uit beide benaderings word beskryf. Uit hierdie resultate word die beste metode van vingerafdruk herkenning vir hierdie tipe data gekies, voordat ons afsluit met 'n bespreking van moontlike verbeterings aan sommige van die benaderings.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Fingerprint comparison is a fundamental method for the identification of people. Fingerprint identification is based on the immutability and the individuality of fingerprints. Immutability refers to the permanent and unchanging character of the pattern on each finger, from before birth until decomposition after death. Individuality refers to the uniqueness of ridge details across individuals. No two persons, even identical twins, have been found to have identical fingerprints.

Because of the large collections of fingerprints and recent advances in computer technology, there has been increasing interest in automatic classification of fingerprints.

The two main uses for the automatic classification of fingerprints are identification and verification. In this thesis we investigate identification. Identification is the process whereby an unknown print is matched against all the prints in the database. The unknown print is identified as belonging to one of the classes of that database.

Verification is the matching of the print from a known class to the prints of the same class in the database. If this matching between the prints is not good enough the print cannot be verified as belonging to that class.

Reliable fingerprint recognition normally requires extensive preprocessing of the fingerprint image to transform it from a greyscale image into a usable binary image. Features can then be extracted from this binary image to use with various classifiers.

Several approaches for preprocessing and fingerprint recognition have been investigated. Preprocessing for fingerprint classification was implemented by Ch and Rao [1]. The main functions of this preprocessing were the filling of isolated holes, removing noise, bridging gaps and finally thinning the image.

Moayer and Fu [2] developed a *syntactic tree system* approach to represent and classify fingerprint images. The fingerprint patterns are subdivided into sampling squares which are preprocessed for feature extraction. A set of regular tree languages is used to describe the fingerprint patterns and set tree automata are used to recognize the coded patterns.

A method for *ridge detection* in a fingerprint was investigated by Verma, Majumdar and Chatterjee [3]. This method of ridge detection divides the print into sampling matrices for which the dominant ridge direction is determined. The collection of directions obtained from all the sampling matrices is known as the *directional image.* Verma and Chatterjee [4] classify partial fingerprints by using a regular grammar and the dominant direction of the sampling matrices as features.

Structural matching is used by Hrechak and McHuch [5] to identify fingerprints. Structural matching is based on local relations among features. That is the identification of identical characteristics that are identically positioned.

We have investigated several of these approaches and found them inadequate for application in a fingerprint recognition system. A possible fingerprint recognition system should contain the following attributes:

- It must be automatic,
- use dedicated hardware,
- be fast
- and very reliable.

The preprocessing [1] is not effective enough for an automatic fingerprint recognition system. The same can be said for the various approaches to classification. We feel that better results can be obtained by the use of a new generation classifier such as the neural-net classifier and features which describe the fingerprint in a more reliable manner. Hrechak et al. [5] obtained 100% on a training set, but classification performance dropped using the test set. (A test set was generated by deleting regions from the training set.) Verma et al. [4] achieved a classification performance of 75% using the directional image.

In this thesis we describe the approaches we have followed in solving the fingerprint recognition problem. We describe the preprocessing implemented, the different types of feature vectors extracted and the different classifiers used. The classifier performance for the various approaches is presented as well as a comparison between the different approaches.

Two approaches were followed in classifying fingerprints. They are feature-based and correlation-based classification. Feature-based classification uses features which are extracted from the greyscale or binary image, or a transformation of these images. Various preprocessing algorithms are applied to these images to improve the quality of the image before features are extracted. In some instances, the images are transformed into another domain before an attempt is made to extract features which describe the original image. (An example of such a transformation is the Fourier transform.) These features are then used to classify the fingerprint. Linear, nearest-neighbour, neural-net or any other classifier can be employed.

Correlation-based classification is an attempt to classify fingerprint images by means of a correlation classifier. A correlation classifier determines the best match for an unknown test class. Preprocessing algorithms applied to the images before the correlation can improve the classification. Such preprocessing techniques were developed and are presented in this thesis. They include a recursive-greyscale edge follower to binarize the greyscale image and an adaptive window concept for the smoothing of the binary image.

We now give a short description of the topics that are discussed in each chapter of this thesis.

Chapter 2 discusses ways in which the fingerprint recognition problem can be solved. The classical approach normally used for fingerprint recognition is discussed and the pattern recognition approach followed in this thesis is presented. We also discuss the various types of classifiers used in our experiments, i.e. the nearest-neighbour, linear, neural-net and correlation classifiers. The advantages and disadvantages of each of these are discussed.

In the next chapter (Chapter 3), two preprocessing algorithms devised by the author are discussed. These are the recursive-greyscale edge follower in Section 3.2 and the algorithm for smoothing of the binary image in Section 3.3. A third algorithm developed by Orit [6] is used to extract the skeleton from the smoothed binary image. These preprocessing algorithms can be applied in the feature-based as well as the correlation-based classification schemes. Examples of results of the preprocessing algorithms are also given. The effect on classification performance by the application of these algorithms is discussed in Chapters 4 and 5.

In Chapter 4 feature-based fingerprint recognition is investigated. Features can be extracted from the fingerprint image in the spatial domain as well as the frequency domain. The reliability of these features is improved by the application of the preprocessing algorithms described in Chapter 3. Different types of features are extracted from fingerprint images. These features include:

1. feature vectors extracted from the directional image (a numerical description of the ridge flow of a fingerprint),
2. feature vectors extracted from the frequency domain and
3. feature vectors built by extracting minutiae from the fingerprint image.

The performance of three different classifiers (a nearest-neighbour, linear and neural-net classifier) are measured on these different types of feature vectors. The effect of the application of the preprocessing algorithms of Chapter 3 is measured and presented.

Correlation-based classification is discussed in Chapter 5. In this approach no features are extracted from the fingerprint image. Classification is obtained by a "direct" comparison of the fingerprint images. Preprocessing algorithms such as those described in Chapter 3 play an integral part in the successful use of the correlation classifier. By good preprocessing the matching of fingerprint images is improved, which means better classification results are obtained. The effect of the preprocessing is shown in Chapter 5.

In our final chapter (Chapter 6) we compare the feature-based and correlation-based approaches. The "best" type of feature in the feature-based classification scheme, as well as the reasons for the good performance are discussed. We conclude with a discussion of possible extensions to some of the approaches we have investigated.

Chapter 2

Methods of fingerprint recognition

In this chapter we discuss two methods of fingerprint recognition: the classic approach used by the police and our own automated pattern recognition approach. These two were chosen because the first is the most common and well known method and because the second has some new and innovative approaches not used in the approaches described in Chapter 1.

2.1 The classic approach

There are three basic patterns in fingerprints, namely *arches, loops* and *whorls*. They can be subdivided into eight types: *plain arch, tented arch, ulnar loop, radial loop, plain whorl, central pocket loop, double loop* and an *accidental* pattern as shown in Fig. 2.1.

1. The *plain arch* is the simplest pattern. The ridges enter on one side, rise to form a wave in the center, and exit smoothly on the opposite side.
2. The *tented arch* is a variation of the plain arch. Ridges at the center are thrust upward in a more abrupt manner.
3. The *radial loop*, which is a subset of the shown *plain loop*, is a pattern in which one or more ridges enter on the side toward the thumb (the side on which the radius bone on the forearm lies), recurve, and then exit on the same side.

4. The *ulnar loop* (also a type of plain loop) is a pattern in which one or more ridges enter on the side toward the little finger (the side on which the ulna bone of the forearm lies), recurve, and then exit toward the same side.

5. The *plain whorl* is a pattern in which one or more of the ridges form a complete revolution around the center.

6. The *central pocket loop* is a variation of the plain whorl pattern. Some ridges tend to form a loop pattern, which recurves and surrounds a whorl at the center.

7. The *double loop* is another type of whorl. In it two separate loop formations are present and may surround each other.

8. The *accidental* is used to describe those patterns that do not conform to any pattern previously described. It is quite rare.

In Fig. 2.1 only a plain loop is shown, as the difference between the ulnar loop and the radial loop is only applicable for a specific hand.

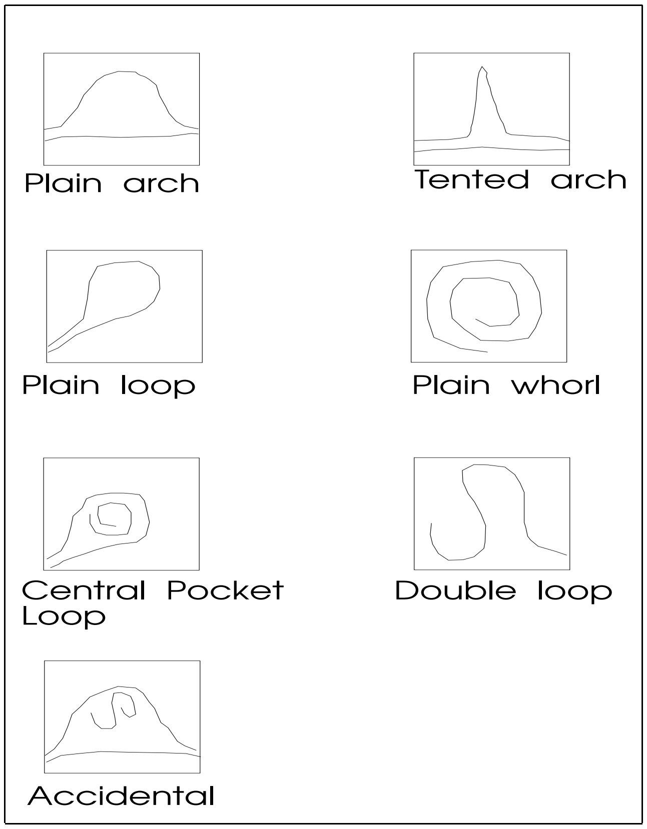


Figure 2.1: Basic fingerprint patterns

Two more features of a print are used in classification: the *core* which is the approximate center of the pattern and the *delta* which is the outer, terminal point of the pattern. Examples of these are shown in Fig. 2.2.

The fingerprint pattern also contains other features that can be useful in classification of fingerprints. These features, known as *minutiae* are those small unique patterns on the fingerprint such as ridge endings and forks. Examples of minutiae as well as the role they play in classification are described in Section 4.3.

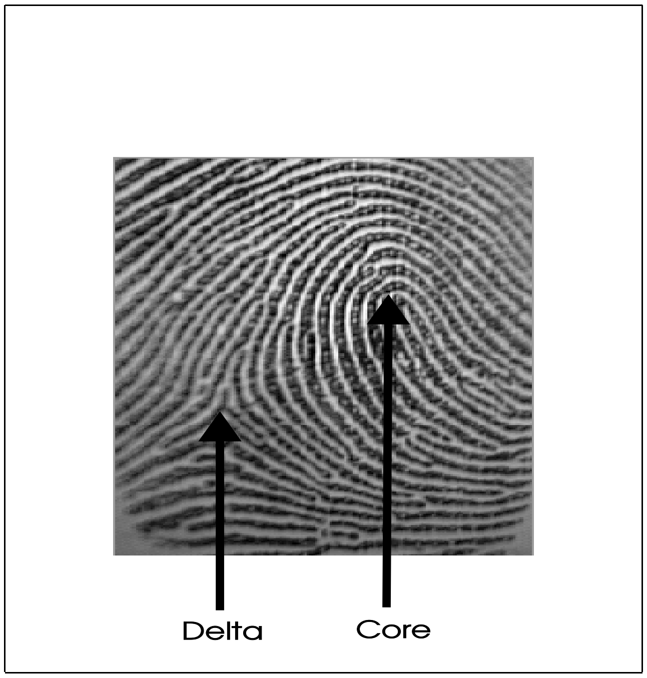


Figure 2.2: The core and delta of a fingerprint pattern

The goal of the classic fingerprint classification system is to assign a working formula to a set of fingerprints which will enable the set of prints to be classified or located in a file. The formula consists of numerical values which are assigned to fingerprint patterns (the values differ from finger to finger). These values are then summed to form a numerical description of the set of fingerprints, which are used in conjunction with the type of pattern appearing in the index fingers and numerical values computed from the ridge counts of various fingers.

One such method is the **Henry system** [7]. This system makes use of the complete set of 10 fingerprint patterns of both hands to classify a person. The complete modified Henry system which is an extension of the original Henry system is described in [7].

2.1.1 The Henry System

We briefly describe the major steps in the original Henry system, as the modified Henry System is similar, but requires a few more classification steps.

- The first step in classification using the Henry system is *primary classification*. Different numerical values are assigned to whorl patterns of different fingers. The right-hand thumb having the highest value and the left-hand little finger the smallest value. The sum of these numerical values is computed for odd and even fingers. This sum is then used in the primary classification.
- *Secondary classification* follows primary classification and is based on the types of patterns appearing in the index fingers.
- Next is *subsecondary classification* which is based on the ridge count (the number of line crossings from the core to the delta) for the index, middle and ring fingers.
- *Final classification* follows the subsecondary classification. It is based on the ridge count of the loop pattern on the right little finger. If the right little finger does not have a loop pattern, a ridge count is made of the left little finger.

All these values obtained from the different steps are then used to form a description of the set of fingerprints for that specific person.

There are other classification systems, but most are merely modifications and extensions to Henry's original system. The only other system worth mentioning is the **Battley System** [8] which is used to file and retrieve single prints. This system determines the general pattern type, after which the core is located and classified. Delta and ridge counting in loops and ridge following in whorls are employed to describe a single fingerprint pattern. The police verifies the identity of a person by matching two prints of the same finger by a system known as the seven point system. One print is normally retrieved from a data base, while the other is that lifted from a crime scene. These two prints are then compared by matching seven or more minutiae. If seven corresponding features are identified on the prints, the person is positively identified.

In our approach the idea is to classify single prints in an automatic manner using pattern recognition ideas. Various solutions exist for solving the fingerprint recognition problem using this approach. In our next section a functional blockdiagram of a typical classification system as well as the methods used in the approach are presented.

2.2 The pattern recognition approach

A pattern recognition solution to the fingerprint problem is shown in Fig. 2.3. The system consists of five subsystems. Data generation is the first step. It transfers the 3-dimensional print into a usable digitized greyscale image. The image is used in the second subsystem which performs preprocessing, such as the binarization etc. as described in Chapter 3. Feature extraction follows the preprocessing. This subsystem tries to generate a unique feature vector for the data which was generated in the first step. Feature extraction is followed by classification. In this subsystem a classifier is used that was trained on the vectors generated during the feature extraction phase. The result of classification is the identity of the fingerprint. The final step is the postprocessing stage where the results of the classifier are evaluated.

Different methods and algorithms can be employed in the pattern recognition approach. The methods we investigated in this thesis are summarized in the graphical representation in Fig. 2.4. It shows a tree with two main branches that represent the approaches that we investigated, namely feature-based and correlation-based recognition .

The branch on the left side represents the feature-based recognition approach. We divided this approach into two sub-branches. The first sub-branch consists of the directional image calculation, feature extraction from the directional image and finally classification. The other sub-branch consists of preprocessing, before we split it up once again. In this division minutiae is extracted before classification, while a Fourier transformation transforms the image into the frequency domain from which features are extracted by using a wedge-ring detector. These features are then used for classification purposes.

The other approach we investigated is represented by the right branch. In this approach we performed preprocessing before classification with the correlation classifier.

In the solution as presented in Fig. 2.3, various algorithms can be employed for data generation through to the classification. Different classifiers as well as different types of feature vectors can be used. We discuss the classifiers that were used in our experiments, as well as their main advantages and disadvantages, in the following section. The different types of features are discussed in Chapters 4 and 5.

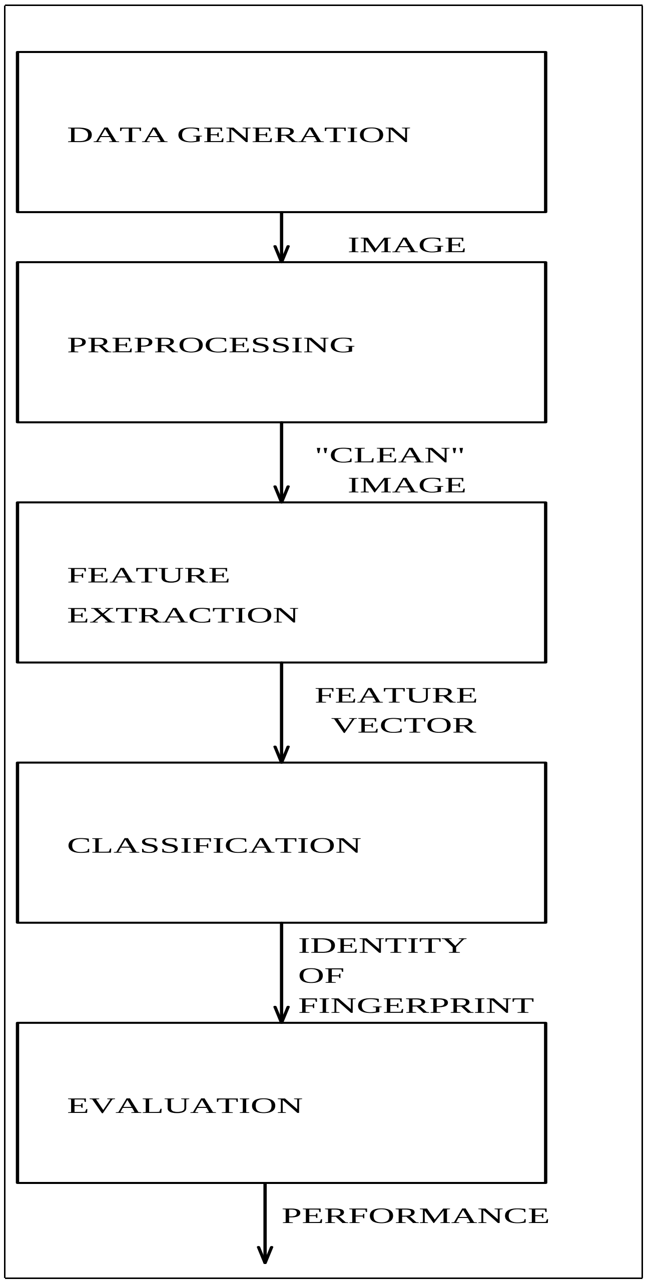


Figure 2.3: Functional blockdiagram

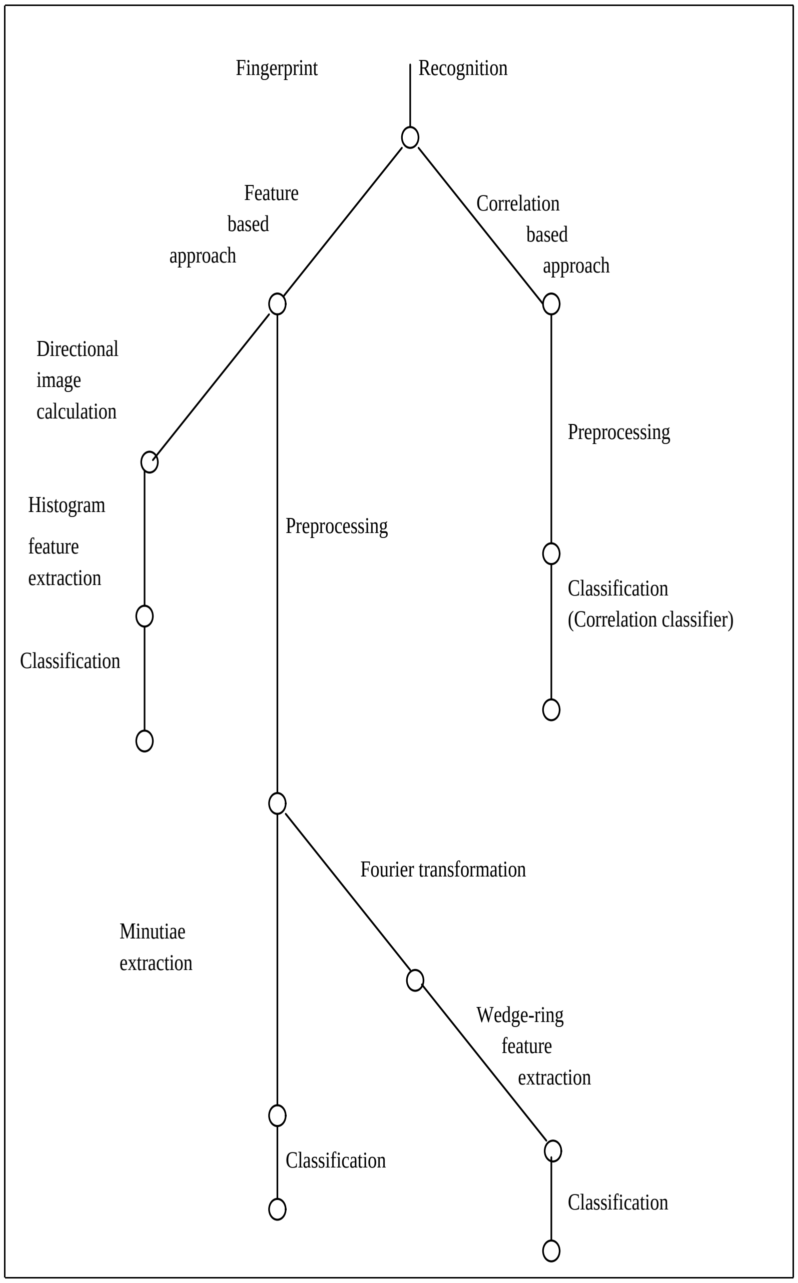


Figure 2.4: Methods in fingerprint recognition

2.3 Classifiers

2.3.1 The Nearest-Neighbour Classifier

This classifier is the most common and simplest classifier. It consists of a database containing the set of labelled training feature vectors. An unknown test vector is classified in the following manner: the distance between the test vector and every one of the training vectors is computed. The unknown test vector is labelled as belonging to that class for which the distance is the smallest. As distance measure we used the euclidian distance E computed with the following equation:

$$E_{i,j} = \sum_{d=1}^{D} \mid \mathbf{T}_d^i - \mathbf{U}_d^j \mid$$

with D the dimension of the feature vectors, $\mathbf{T}^i$ the i^{th} test vector and $\mathbf{U}^j$ the j^{th} training vector.

One advantage of the nearest-neighbour classifier is that no training is associated with it. This means that the classifier is ready to use the moment labelled feature vectors are available. Another advantage is its ability to generate extremely complex decision boundaries, which enables the classifier to generalize very effectively.

This classifier has a few disadvantages. It is extremely noise sensitive. This means a test vector can be labelled as belonging to a certain class because the distance between the test vector and an outlying vector of that training class was the smallest. Two more disadvantages are the extensive computation time for classification (the euclidian distance between the test vector and every one of the training vectors has to be calculated for each classification) and the large storage space needed because all the training vectors have to be stored for use in calculations for each classification.

2.3.2 The Linear Classifier

The linear classifier use linear discriminant functions. Such a linear discriminant function can be written as

$$\mathbf{g}(\mathbf{x}) = \mathbf{w}^{\mathbf{t}}\mathbf{x} + w_0$$

where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ the feature vector and w_0 the threshold weight. According to [9]

- a linear discriminant function divides the feature space by a hyperplane decision surface,

- the orientation of the surface is determined by the normal vector $\mathbf{w}$,
- the location of the surface is determined by the threshold weight w_0,
- the discriminant function is proportional to the signed distance from $\mathbf{x}$ to the hyperplane.

Determining the weight vector $\mathbf{w}$ and the threshold weight w_0 is a field of study on its own. Several procedures have been investigated. A popular method of finding $\mathbf{w}$ is to define a criterion function $\mathbf{J}(\mathbf{w})$ that is minimized if $\mathbf{w}$ is a solution vector. This reduces the problem to minimizing a scalar function, and can often be solved by a gradient-descent procedure. Starting with an arbitrary weight vector $\mathbf{w_1}$, computing the gradient of $\mathbf{J}$ with respect to $\mathbf{w}$ at this point $\mathbf{w_1}$, and moving a small distance from $\mathbf{w_1}$ in the direction of the steepest descent, we obtain the next estimate of $\mathbf{w}$. In general then:

$$\mathbf{w_{k+1}} = \mathbf{w_k} - \rho_{\mathbf{k}} \bigtriangledown \mathbf{J}(\mathbf{w_k})$$

Hopefully this sequence of weight vectors converges to a solution minimizing the criterion function.

Several criterion functions have been suggested [9]. These include:

- The perceptron criterion function,
- a minimum squared-error criterion function,
- the *Widrow-Hoff* criterion function
- and the *Ho-Kashap* algorithm.

The linear classifier we used in our experiments uses the minimum squared-error criterion function. The linear classifier draws linear decision boundaries in feature space. This is the main disadvantage if the data is not linearly separable, in which case no meaningful test classification can be obtained since the net does not have the ability to distinguish between the different classes. However, the linear classifier is very fast and good classification can be obtained if the data is linearly separable. In our experiments we found that some features are linearly separable as the test classification performance was 100%.

2.3.3 The Neural-Net Classifier

The neural-net classifier used in our experiments is a three-layer Perceptron-type trained with the conjugate-gradient algorithm [10].

A 3 layer neural-net classifier is shown in Fig. 2.5. The first layer is the input layer whose outputs are the values of the elements in the feature vector to be classified. Thus the number of input neurons is equal to the dimension of the feature vector plus one. The second layer is the hidden layer while the third layer is known as the output layer. Each neuron in the second layer computes the sum of all its inputs. This sum is then used in conjunction with a transfer function to determine that neuron's output. The number of hidden neurons is problem specific and must be determined by means of repeated experiments with different numbers of hidden neurons. The output layer has one neuron for each class. Each output neuron performs the same calculation as the hidden layer neurons. We used a sigmoid transfer function for both hidden and output neurons. If a specific neuron in the output layer has a high output value for an input feature vector, the feature vector is labelled as belonging to that class for which the output neuron is high.

Each connection between the the input layer and hidden layer, as well as each connection between the hidden layer and the output layer has a certain numerical value associated with it. All these numerical values are contained in the weight matrix **W**. The training of the neural net consists of determining the best weight matrix **W** for that specific problem. The initial values of the weight matrix are determined randomly but must adhere to specific rules. These rules state that the values must not be too big or too small and that they must be distributed randomly [11].

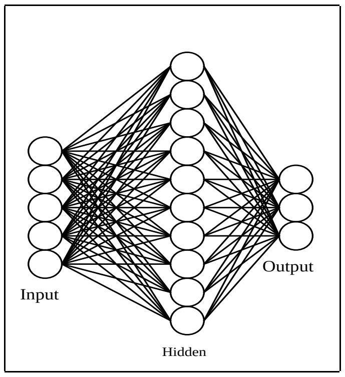


Figure 2.5: 3-layer Neural-network classifier

The neural net has several distinct advantages. The foremost is its ability to generalize very well due to its ability to generate very complex decision boundaries in feature space. If overtraining occurs, this can be a drawback if too complex decision boundaries are drawn. Another advantage is that the neural net does not require a huge amount of memory during classification. The number of computations required during classification is also very small.

2.3.4 The Correlation Classifier

This classifier also contains a database of labelled training images. To classify, it determines the correlation between the test image and all the training images. The test image is then labelled as belonging to the class of the training image that gives the largest correlation value anywhere in the correlation function.

Correlation is normally implemented in the spatial domain. For the functions f and g the correlation is defined as

$$f(x,y) \otimes g(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) g(x+m, y+n)$$

with $x = 0, 1, 2, ..., M-1$ and $y = 0, 1, 2, ..., N-1$ [12].

However, this is computationally expensive. This is a big disadvantage of the correlation classifier.

For this reason we implemented the correlation in the frequency domain. For the functions f and g the following correlation theorem holds:

$$f(x,y) \otimes g(x,y) \leftrightarrow F(u,v) G^*(u,v)$$

where * represents the complex conjugate, F and G the Fourier transforms of f and g respectively and $\otimes$ the correlation in the spatial domain.

The Fourier transform F of a function f can be computed with the following equation:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp[-j2\pi(ux/M + vy/N)]$$

for $u = 0, 1, 2, ..., M-1$ and $v = 0, 1, 2, ..., N-1$.

The computational burden is still considerable when implementing the correlation using the frequency domain, as the Fourier transform has to be calculated for each test image and each training image, followed by an inverse Fourier transform calculation of the product of these Fourier transforms.

We demonstrate the computational burden of both these approaches with an example. A normal implementation in the spatial domain requires the following number of multiplications and additions:

$$(M \times N)^2$$

with M and N the x and y dimensions of the images respectively. This is an upper limit in the correlation calculation. For 256$\times$256 images the computational expense for this implementation is 4.3×10^9 calculations.

A more suitable implementation would require the following number of multiplications and additions:

$$2\sum_{j=1}^{N-1}\sum_{i=1}^{M-1}(M-i)(N-j)+M\times N$$

where M and N are the same as described previously. This implementation requires 2.1×10^9 calculations.

The computational expenses associated with correlation implemented in the frequency domain is of the order [12]:

$$2\times M\times M\times M\log_2 M+M\times M$$

where we assume that the images are square with dimension M. For a 256×256 image the value of $M=512$ to prevent aliasing. This implementation requires 1.7×10^9 multiplications and additions, which is less than the calculations required for the spatial implementation.

This concludes our description of the different classifiers and different approaches to fingerprint recognition. This chapter was meant as an introduction into the basic fingerprint methods. Our next chapters give more detail of the solution to the fingerprint recognition problem. The next chapter (Chapter 3) deals with methods of data generation as well as preprocessing algorithms.

Chapter 3

Preprocessing

Preprocessing plays an integral part in any classification system. Good preprocessing techniques reduce the effect of poor quality data and this results in better classification performance. In this chapter we discuss two methods of data generation, namely the paper-and-ink method and an optical data generation method, and two preprocessing algorithms developed by the author specifically for fingerprint recognition. We also present some results obtained by the application of these algorithms, as well as a skeleton extraction algorithm, to a fingerprint image. The application of preprocessing algorithms such as the skeleton extraction algorithm enables us to improve classifier performance. This is demonstrated in Chapters 4 and 5.

3.1 Data generation

In our system fingerprint images are digitized by means of a video camera and a framegrabber. The video camera's output is an analog signal of the fingerprint image. This analog signal is then quantized into 256 discrete levels by means of the framegrabber. The framegrabber stores this quantized image as a two-dimensional array of bytes.

Two main methods of data generation are known. The first method is the well known ink-and-paper method. The finger is rolled in ink and then rolled on paper to transform the three-dimensional finger into a two-dimensional print that can be photographed by the video camera and quantized by the framegrabber.

This method is not very reliable because the ink can smear and blur the whole image or part of the image. Plastic distortion plays a role as well, because the finger being rolled on the paper is not pressed equally hard each time the print is taken. It is also time consuming and very dirty. This makes it unsuitable for application in any automatic fingerprint classification system, but can be used in experimentation as well as manual methods such as those employed by the police.

The second method is a more efficient and reliable optical data generation system. The system as shown in Fig. 3.1 consists of a prism and a uniform light beam that transforms the three-dimensional data into two-dimensional data which can be photographed and quantized.

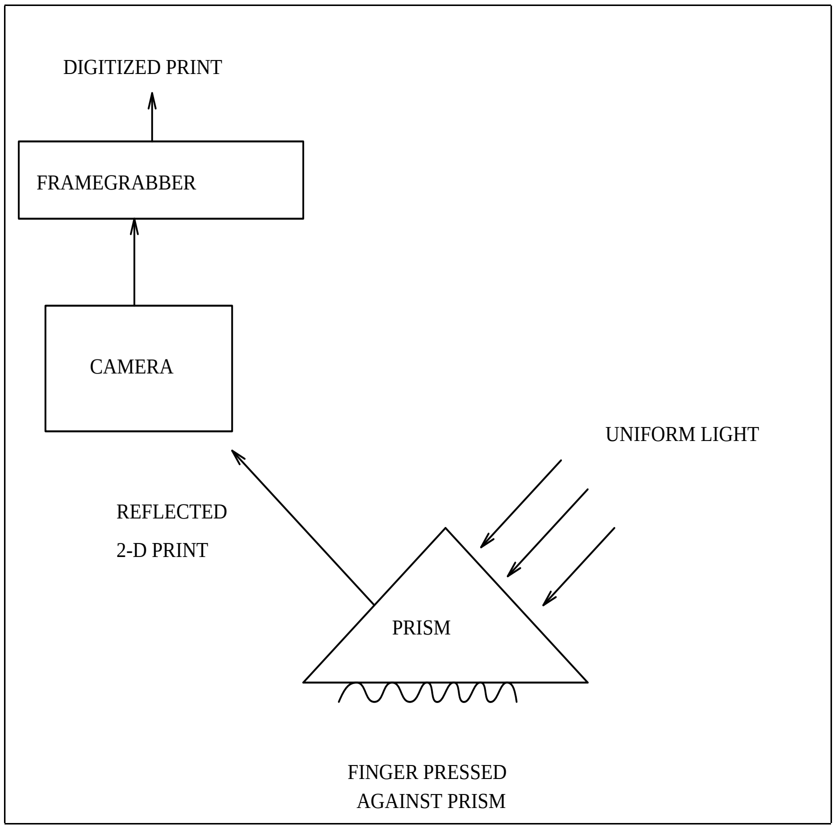


Figure 3.1: Optical data generation

This system makes use of the total internal reflection obtained in a 90° prism. The uniform light beam is shone into the prism. Normally all this light is reflected out of the prism by the 45° surface of the prism. However, when a finger is pressed on this surface, the reflection coefficient of that surface is changed at the places where the ridges of the finger touch the prism's surface. At the places where the refraction index is changed, no light is reflected. This means that the ridges of the print are not reflected and appear dark in the image, while the background is of a high intensity. This image of the fingerprint consisting of reflected light and unreflected light is photographed by the video camera and quantized by the framegrabber.

This method of data generation is not perfect either. The contrast and focus of the image obtained from the optical method are poor. Most images have a continuous slope in greyscale values superimposed on the finer detail, which means that binarization techniques based on a global threshold are unsuitable. However the advantages outweigh the disadvantages. The method is clean and very fast and most of the problems can be overcome by good preprocessing techniques such as greyscale-to-binary conversion as described in the next section.

3.2 Binarization

The algorithm used in our system for the conversion of the greyscale image into a binary image is based on a recursive procedure for line thinning by line following [6]. We converted this line-following algorithm for use with greyscale images [13].

3.2.1 The algorithm for binarization

We now describe the greyscale-to-binary image conversion algorithm.

The edges of the greyscale fingerprint are extracted using the Marr-Hildreth algorithm [14]. (We refer to the image obtained by the application of the Marr-Hildreth algorithm as the "gradient image".) The Marr-Hildreth algorithm extracts a binary outline (the edges) of the ridges from the original greyscale image. These edges in the gradient image are one pixel wide. This image is then used in conjunction with the greyscale image to extract the binary image from the original greyscale image.

The binarization is performed by means of adaptive windows. Two windows are used in each step of a recursion process to determine the binary image. The windows are the gradient window built in the gradient image, and the greyscale window built in the greyscale image.

The algorithm is based on six steps which we summarize here and illuminate further in subsequent sections:

- The *first step* in the algorithm is the determination of a suitable starting point for the recursion. The greyscale image is used to determine this starting point. The current lowest greyscale value in the image is found. (Assume this point is X). This location is used as the reference point around which the gradient and greyscale windows are built.

- The *next step* is the building of two small windows (the greyscale and gradient windows) in their respective images. The location of these windows is either the point X determined in step 1, or the location determined in step 6. The determination of the dimensions of these windows are described in Section 3.2.4. Fig. 3.2 shows part of the greyscale image with the small greyscale window superimposed on the image.

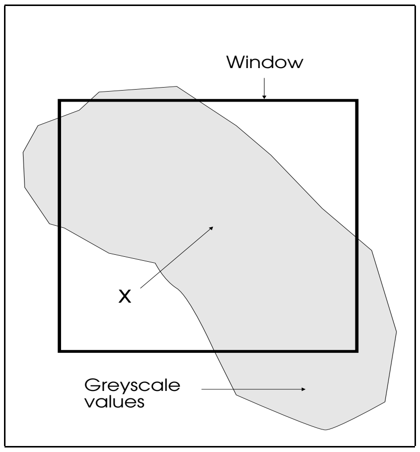


Figure 3.2: Greyscale image with window

- The *third step* is the determination of transition regions on the boundaries of the greyscale and gradient windows. A transition region is that part on the boundary of a window which has been identified as containing useful information such as ridges of a fingerprint. This process of determinating transition regions is explained in Sections 3.2.2 and 3.2.3. The transition regions of a window determine the next location of the windows in the recursion process.

- The *fourth step* in this algorithm is the evaluation of the number of transition regions which were calculated for each window. If the number of transition regions calculated for both greyscale and gradient windows is the same, the transition regions of the greyscale window are used in the following steps of the algorithm. If the number of transition regions differ, we know that a problem occurred due to the poor quality of the image at that location. Then we use the window with the smallest number of transition regions in the following steps of the recursion process. We use the smallest number of transition regions because a false edge can be created if the bigger number is used. As a result of this choice the image is much clearer as the noise is suppressed. If either one of the windows has zero transition regions, the other window's transition regions are used in the following steps. If both windows have zero transition regions we know that we have reached the end of a ridge whereupon the recursion returns to a previous level. Fig. 3.3 shows a window with two transition regions.

- The *fifth step* is the binarization of the greyscale image. This binarization is conducted for each step in the recursion. Both the greyscale and the gradient windows are used in the binarization

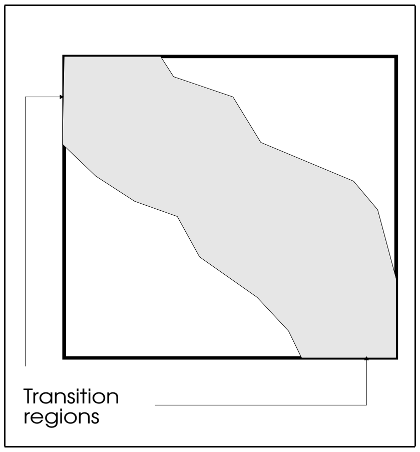


Figure 3.3: Window with two transition regions

process. The binary image is determined for each of the windows as described in Section 3.2.5. The "OR"ed binary image of the two windows is the binary image of the fingerprint.

- The *sixth step* is the determination of the location of the windows in the next step of the recursion. The transition regions of the window which was chosen in step four are used. The endpoints of one the transition regions of the window are used as the reference points around which the greyscale window in the greyscale image and the gradient window in the gradient image are built. Consecutive steps in the recursion process are shown in Fig. 3.4. The current window as well as the next window in the recursion process are shown. If the chosen window has more than one transition region the first transition region's endpoints are used as the location around which the following windows will be built (step 2). The first ridge is thus followed by the building of successive windows until no more transition regions are found for any of the two windows. The recursion then returns to that depth of recursion where more than one transition regions were found. It then follows the other ridge by using that transition region's endpoints as the location for step 2. In this way all the ridges connected to each other are followed to their respective ends.

Once the location of the next windows have been determined the process starts again with step 2, which is the building of the windows in the gradient image and the greyscale image.

Steps 2 through 6 are repeated until no transition regions exist for either one of the windows. After this the process starts again with step 1 which is the determination of the lowest greyscale value in the greyscale image for the first location for the two windows.

We now describe the methods by which the transition regions are determined for the greyscale and gradient windows, and the method by which the dimensions of the small window is calculated for a

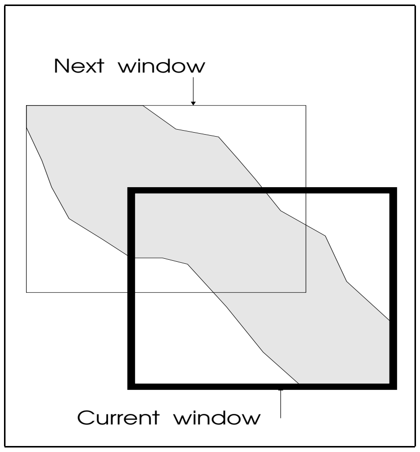


Figure 3.4: Next step in recursion

step in the recursion.

3.2.2 Determination of transition regions for the greyscale window

To determine the transition regions on the boundary of the greyscale window, the boundary of the window is searched for a maximum value (in our case ridges, the foreground, have the lowest greyscale values while the background has the higher greyscale values). Next the current greyscale window's lowest value on the boundary is found and is compared with the maximum value. If the difference between these two is larger than a certain percentage of the maximum we assume that the window is situated on a ridge. If not, and no transition regions are found on the boundary of the gradient window, we know that the lowest greyscale value was due to spot noise in the image. We then return to step 1 to determine a new starting location for the windows.

Assume that the difference between the maximum and the minimum was larger than the required value. (We used 0.6 of the maximum.) The boundary of the current greyscale window is then searched for occurrences of values less than the threshold set by the percentage of the maximum. All these elements which are less than our threshold and which are on the boundary of the greyscale window is known as transition region elements. A collection of this transition region elements forms a transition region. Fig. 3.3 shows the window with two transition regions on the boundary of the window.

Next we describe the determination of the transition regions for the gradient image.

3.2.3 Determination of transition regions for the gradient window

The algorithm to determine the transition regions for the gradient image window is based on a blobcolouring routine [15] which tries to colour the ridges in the gradient window. The algorithm searches for the smallest greyscale value in the greyscale window which corresponds to an edge pixel in the gradient window. This value is assumed to be the largest greyscale value that could be part of a ridge and is used as an upper threshold in the blobcolouring routine. The blobcolouring routine searches for a pixel with the background colour in the gradient image and colours pixels connected to this, but cannot cross the edge pixels of foreground colour which forms the gradient image. When a pixel with the background colour in the gradient image is found, the corresponding greyscale value in the greyscale window is compared with the upper threshold determined earlier. If the greyscale value is lower than the upper threshold, the blobcolouring routine colours the pixel because it is assumed to be part of the ridge. If the greyscale value is higher than the upper threshold we know that the blobcolouring routine is probably outside the edges of the gradient image (and this could possibly result in incorrect transition regions.) Because of this the pixel is not blobcoured.

This procedure prevents the blobcolouring routine from failing if the edge of the gradient image is not a solid line. The transition regions are then determined by counting the areas in which the blobcolouring colour occurs on the boundary of the gradient window.

3.2.4 Determination of window dimensions

Assume that the window must be built with location X and Y as the reference points. The window is built around these two points with a buffer of d pixels on each side. This is shown in Fig. 3.5. The dimensions of the window can therefore be calculated as :

$$DIMX = \mid X.x - Y.x \mid + 2 * d$$

and

$$DIMY = \mid X.y - Y.y \mid + 2 * d$$

where $X.x$ is the x-coordinate of point X, $Y.x$ the x-coordinate of point Y and similarly $X.y$ the y-coordinate of point X and $Y.y$ the y-coordinate of point Y.

A value of two was used for d in all our experiments. Step 2 of Section 3.2.1 builds the small windows in the gradient and greyscale windows. If step 2 follows step 1, X=Y; if step 2 follows step 6, X and Y are the endpoints of a chosen transition region.

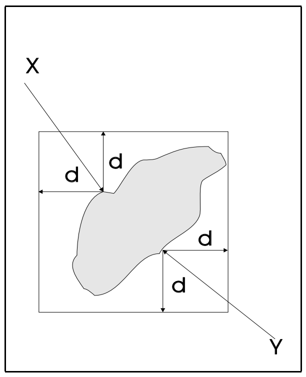


Figure 3.5: Window with *d* pixel buffer

3.2.5 Calculating the binary image from the windows

The binary image for a set of windows is formed by the logical "OR"ing of a binary image determined from the greyscale window and a binary image determined from the gradient window.

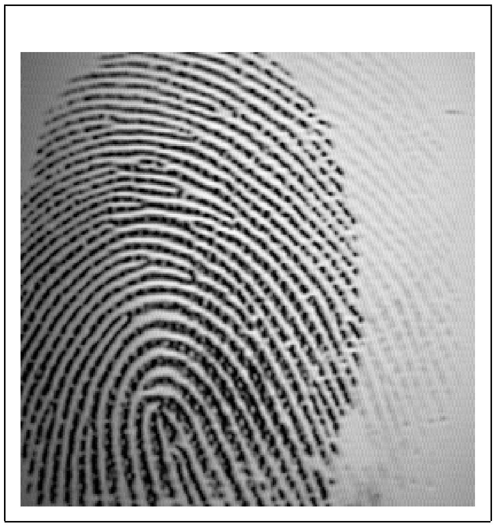
The binary image for the greyscale window is formed by thresholding the greyscale window using a certain percentage of the maximum (60 % in our case) as threshold. The elements below the threshold are retained. The binary image of the gradient window consists of all the elements in the window which were blobcoloured.

The elements in the "OR"ed binary image are deleted from the original greyscale image as well as the gradient image to prevent the recursion from retracing its path.

This algorithm works very well for images with a continuous slope in greyscale values. An example of the application of this algorithm on the greyscale image of Fig. 3.6(a) is shown in Fig. 3.6(b). One problem of this algorithm is that the binary image is not a very smooth image. It has holes and gaps in the ridge parts of the image. One of the eventual goals is to extract the skeleton and the existence of these holes and gaps deforms the skeleton considerably, as can be seen in Fig. 3.15. Fortunately this problem can be overcome by the algorithm described in the next section. It improves the binary image by removing the holes and gaps in the binary image and is known as smoothing.

3.3 Improvement of binary image

The improvement of the binary image algorithm is based on an adaptive window concept. The first part of the algorithm locates possible isolated holes or gaps. Next the isolated holes or gaps in the


(a) Greyscale image

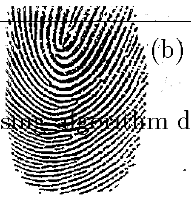

(b) Binary image

Figure 3.6: Binarization of greyscale image using algorithm described in Section 3.2

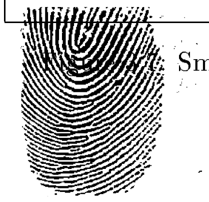


Figure 3.7: Smoothed binary image

edge part of the fingerprint are removed. This is done by using the adaptive window and a recursive blobcolouring routine.

3.3.1 Finding possible isolated holes or gaps

The binary image is searched from top left to bottom right for any possible isolated holes or gaps. An isolated hole is defined as any island of background colour in the image completely surrounded by the foreground colour. Fig. 3.8 shows an isolated hole in the ridge. Gaps are defined as regions of background colour not completely surrounded by foreground colour, which are not part of a merge or branch point. Fig. 3.9 shows a gap in the ridge. To find holes or gaps, the image is searched for a transition from the foreground colour to the background colour, left to right and top to bottom. When such a transition is found the position is marked (location X). The image is then searched for a transition from the background colour to the foreground colour. If such a transition is found (location Y) the distance between these two points is computed. If this distance is smaller than a chosen length the algorithm continues with the filling of the hole or the bridging of the gap by using the window. If the computed distance is too big the whole process of finding a transition starts again.

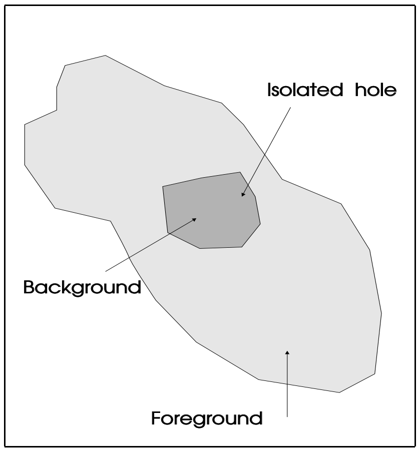


Figure 3.8: Isolated hole in ridge

3.3.2 Removing holes

Once a region has been identified as containing a possible isolated hole or gap the algorithm verifies whether it is a isolated hole or a gap. This is done by building a window containing the region's data. The window is built using the transition points (X and Y) as reference points for the calculation of the window dimensions as described in Section 3.2.4. This window therefore contains the possible isolated hole or gap as well as the surrounding data as shown in Figures. 3.10 and 3.11.

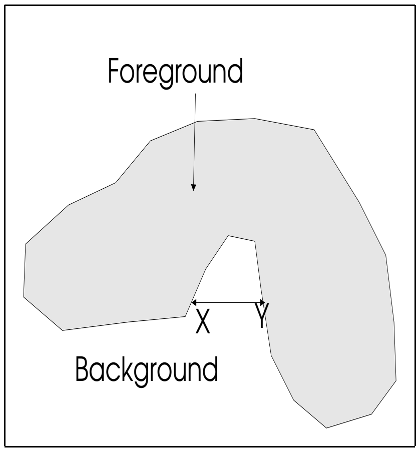


Figure 3.9: Gap in ridge

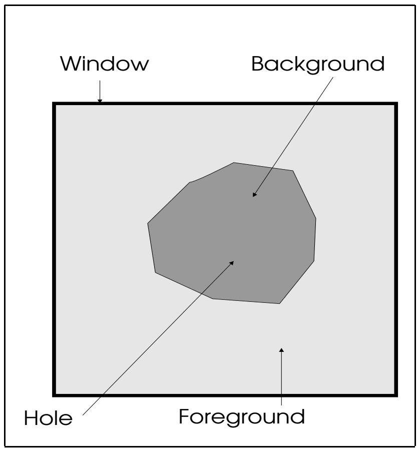


Figure 3.10: Isolated hole surrounded by a window

A 4-connected blobcolouring routine is then used to decide if the possible isolated hole is a definite isolated hole. The window is blobcoloured with the starting point in the region between the initial transition regions, X and Y. The blobcolouring routine seeks the background colour and colours it's path in colour A.

Next the boundary of the window is inspected for any occurrences of the colour A. If none of the boundary elements is of colour A we know that the possible isolated hole is completely surrounded by foreground colour. Therefore we know that it is a definite isolated hole. Such a case is shown in Fig. 3.10. The isolated hole is removed by colouring all the elements in the window of colour A to the foreground colour in the original data. The algorithm then starts again by searching for possible transition regions (X→Y)as described earlier.

If some of the boundary elements were of colour A the hole is not completely surrounded by the foreground colour. Therefore it cannot be filled as it might be a possible merge-branch point or a gap. The boundary elements of the window containing colour A form transition regions. These regions are used in the determination of possible gaps, as described in the next section.

3.3.3 Removal of gaps

This part of the algorithm bridges gaps in the edge part of the fingerprint. An example of a gap surrounded by a window is shown in Fig. 3.11.

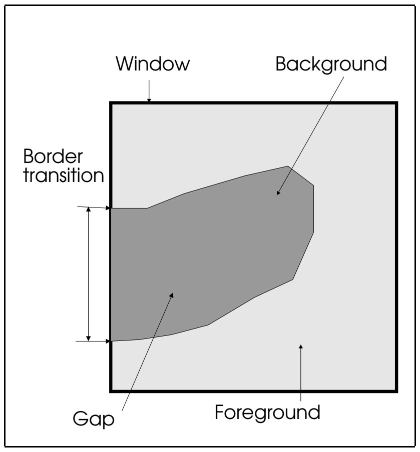


Figure 3.11: Gap surrounded by a window

First the number of transition regions as shown in Figures. 3.10, 3.11 and 3.12 are counted. Fig. 3.10 has no transition regions on the boundary. Fig. 3.11 has one transition region, while Fig. 3.12 has two transition regions. In order to bridge gaps in the binary image the size of each transition region

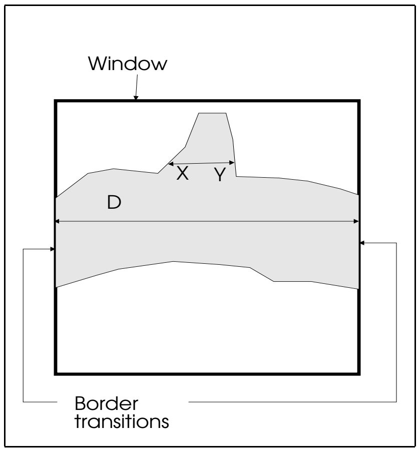


Figure 3.12: Example of window with 2 transition regions

is also calculated. If a window only has one transition region and if the size of the transition region is greater than a user defined length, it is a gap and not part of a possible merge-branch point. This is true if we make the assumption that a true merge-branch point has a very sharp "v". Therefore if the size of the transition region is large we define it as a gap and not as a merge-branch point.

The original gap is then bridged by drawing a line from the two points on the edge where the gap started. This transforms the gap into an isolated hole. The hole is then filled with the foreground colour by the recursive blobcolouring routine.

If the size of the transition region along the boundary of the window was less than the user defined length, or if there was more than one transition region on the boundary of the window, a bigger window is built around the original window. The background is again coloured by the blobcolouring routine. Once again the number of transition regions along the boundary of the window is counted. A heuristic rule determines if it is a gap of not: If the number of transitions regions is less than or equal to two and if the distance (D in Fig. 3.12, if two transition regions are present) between the regions on the boundary of the window is greater than the user defined length, the gap is bridged. All other occurrence counts are ignored as being noise as it is highly unlikely that more than two transition regions will exist on a window boundary for a clear fingerprint.

The application of this algorithm to a binary image smooths it. This smoothed binary image is then more suited for the extraction of the skeleton of the binary image.

3.4 Thinning the binary image

When the binary image has been smoothed the skeleton is extracted from the binary image. The skeleton is a thinned binary image of the original binary image which is only one pixel thick throughout the image.

We implemented an adaptive line thinning by line following algorithm [6] to extract the skeleton from the binary image. This algorithm uses an adaptive window to recursively follow a ridge. The skeleton of the ridge is extracted by connecting the center points of adjacent windows in the recursion process.

3.5 Preprocessing results

To evaluate the performance of our algorithm, we tested it on various images of fingerprints. Typical results are shown in Figures. 3.6(b), 3.7 and 3.13 to 3.15. All these results were obtained from the same original image (Fig. 3.6(a)), which was produced by means of an optical data generation method as described in Section 3.1.

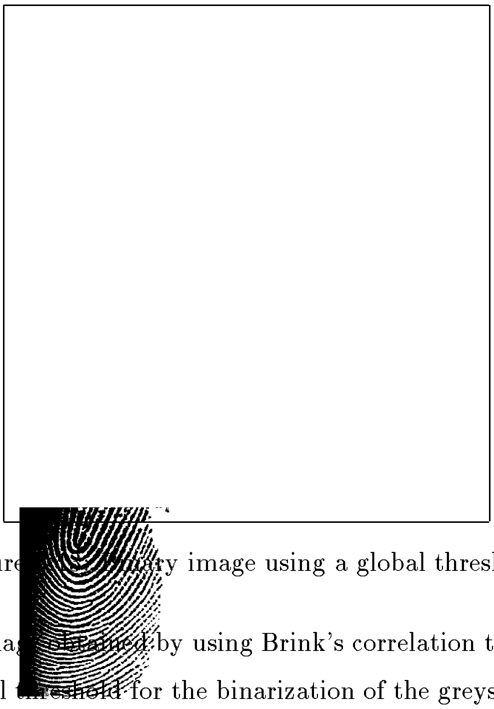


Figure 3.13: Binary image using a global threshold

Fig. 3.13 shows the binary image obtained by using Brink's correlation threshold algorithm [16]. This algorithm determines a global threshold for the binarization of the greyscale image. This threshold is selected by maximizing the correlation between the original greyscale image and a thresholded binary image. As can be seen this method is quite inadequate for our application - the leftmost region is washed out completely. The reason for this is the continuous slope in greyscale values superimposed on the greyscale image.

Fig. 3.6(b) shows the same image obtained by using the recursive greyscale edge follower described in Section 3.2. Although this image is not perfect, it clearly demonstrates the improvement obtained by the application of our edge follower. No region of the image is completely spoilt, and improved ridge continuity has resulted.

The smoothed binary image obtained from the image in Fig. 3.6 is shown in Fig. 3.7. This was obtained by application of the smoothing algorithm described in this thesis. Most of the noise which one would expect to disrupt the binary edge follower has been removed.

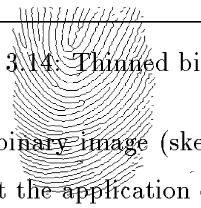


Figure 3.14: Thinned binary image of smoothed image

Fig. 3.14 shows the thinned binary image (skeleton) of the smoothed version. Fig. 3.15 shows the same image's skeleton without the application of the smoothing algorithm. Comparison of these two images clearly demonstrates the improvement in the quality of the skeleton of the image which was smoothed. The unsmoothed image in Fig. 3.15 is disrupted by noise and no reliable features can be extracted from this image. On the other hand, reliable features can probably be extracted from the skeleton in Fig. 3.14, as will be shown in Chapter 4.

In this chapter data generation, preprocessing algorithms and results obtained by the application of these preprocessing algorithms on data that was generated by the optical data generation method were discussed. In the next chapter one branch of the methods in the fingerprint recognition tree in Fig. 2.4 is discussed. This branch is the feature-based recognition approach. Different types of features used are discussed as well as results obtained from using these features with various classifiers.

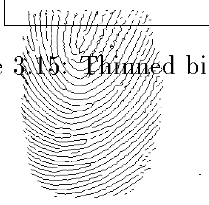


Figure 3.15: Thinned binary image without smoothing

Chapter 4

Feature-based recognition

In this chapter we investigate the feature-based recognition approach of Fig. 2.4. The features used are those extracted from the directional image [17], from the frequency domain [18] and minutiae extracted from the prints themselves. We used these features in experiments with different classifiers. The classifiers used were the neural-net classifier (Section 2.3.3), the linear classifier (Section 2.3.2) and the nearest-neighbour classifier (Section 2.3.1).

We used the same data set in all our experiments. This data set was generated with the optical data generation method described in Chapter 3. The data set consisted of 60 training images and 20 test images from 20 classes. That is, each class consisted of three training images and one test image. Each class was generated from a thumb of a person. The resolution of each image was 256×256 pixels.

In the next section we describe experiments conducted on features extracted from the directional image. Following that we describe experiments conducted on features extracted from the frequency domain and in the final section experiments conducted using the minutiae as features are discussed.

4.1 The directional image

4.1.1 Description of directional image

In this section experiments conducted using the directional image as basis for feature extraction are described. We describe the manner in which the directional image is calculated, as well as the manner in which we extract features from this directional image.

In order to calculate the directional image, the image is subdivided into small blocks. (We chose blocks of 16×16 pixels in our 256×256 images.) The directional image consists of the dominant directions of each of the small blocks. The dominant direction of a block is calculated in the following manner:

The possible directions are quantized and each of these is called a subdirection. Refer to Fig. 4.1 where a quantization of four is illustrated. The four directions are labelled 0 to 3. Numerical values for each of the subdirections are calculated from the pixel intensities, after which the dominant direction is chosen from these. The value of a subdirection is equal to the sum of the absolute differences between the average value and the greyscale values along the subdirection. The subdirection with the smallest value (i.e. smallest variation) is the dominant ridge direction.

Refer to Fig. 4.1 for an example of this calculation. The small block shown is of dimension 5×5. Subdirection 0 consists of greyscale values 5,6,7,8,9 with an average of 7, subdirection 1 of 10,20,7,40 and 50 (average of 25.4), subdirection 2 of 10,15,7,25 and 30 (average of 17.4) and subdirection 3 of 10,20,7,40 and 50 (average of 25.4). Consider subdirection 2: the average value is $(10 + 15 + 7 + 25 + 30)/5 = 17.4$. The absolute values of the differences are: $\mid 10 - 17.4 \mid = 7.4$, $\mid 15 - 17.4 \mid = 2.4$, $\mid 7 - 17.4 \mid = 10.4$, $\mid 25 - 17.4 \mid = 7.6$ and $\mid 30 - 17.4 \mid = 12.6$. The sum of these values is 40.4. In a similar fashion the sum of differences is computed for subdirection 0 (resulting in a value of 6), 1 (78.4) and 3 (also 78.4). For this example subdirection 0 is the dominant direction as it has the smallest sum of differences value (6 compared to 78.4 and 40.4).

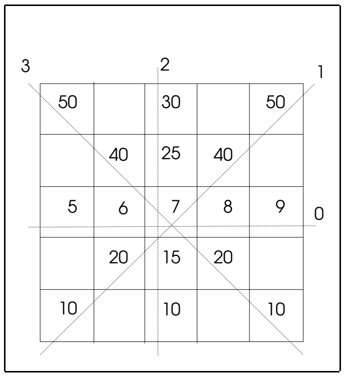


Figure 4.1: Block quantized into 4 subdirections

As the directional image is a numerical description of the fingerprint pattern, different feature vectors can be extracted from the numerical values. In the following subsections we describe the types of features we implemented in our experiments.

4.1.2 Feature vectors

Histogram feature vectors

Our first type of feature vector is based on the histogram of the dominant ridge directions. Each bin in the histogram is used as an element of the feature vector.

We demonstrate this with an example. In Figures 4.2 and 4.3 a small portion of a fingerprint pattern which has been divided into small blocks is shown. There is a total of 25 blocks in this portion of the print: 5 rows and 5 columns. Each block is quantized into 4 subdirections for the calculation of the dominant ridge direction. This means the dominant direction can be any direction from 0 through to 3. Fig. 4.2 shows the dominant direction for each small block while Fig. 4.3 shows the numerical value assigned to that dominant direction in a small block. Because the original small blocks were quantized into 4 subdirections the histogram consists of 4 bins, one bin for each quantized subdirection. For the directional image in Fig. 4.2 the histogram is shown in Fig. 4.4. The feature vector for the directional image in Fig. 4.2 is the same as the values in the bins of Fig. 4.4 i.e. (6,4,9,6).

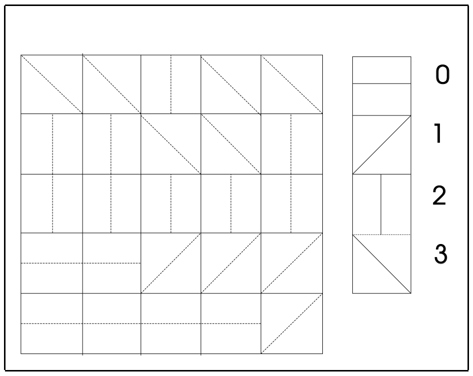


Figure 4.2: Directional image

3	3	2	3	3
2	2	3	3	2
2	2	2	2	2
0	0	1	1	1
0	0	0	0	1

Figure 4.3: Numerical values of directional image

Variations on this type of feature vector can be obtained by excluding some of the small blocks from

Bin 0	Bin 1	Bin 2	Bin 3
6	4	9	6

Figure 4.4: Histogram for Fig. 4.3

contributing to the histogram. We define two types of feature vectors: the *large* feature vector where all the small blocks of the directional image contribute to the histogram and a *small* feature vector where the outer ring of small blocks does not contribute to the histogram. The idea with the small type of feature vector being that noise on the edge of the image is ignored in the calculation of the feature vector. The small feature vector for our example is constructed when row 1 and 5 as well as column 1 and 5 are ignored in the calculation of the histogram. The small feature vector in this case is (1,2,4,2).

Feature vectors for direct comparison

This type of feature vector is constructed using the directional image as basis. The feature vector's dimension is equal to the number of small blocks in the directional image. The numerical values of the blocks in the directional image are elements of the feature vector. That is, the dominant direction of the x^{th} block is inserted into the x^{th} element of the feature vector. For the directional image of Fig. 4.2 the corresponding feature vector is (3,3,2,3,3, 2,2,3,3,2, 2,2,2,2,2, 0,0,1,1,1, 0,0,0,0,1). This type of feature vector is defined as a *large* type of feature vector.

The *small* feature vector is again constructed by ignoring the outer ring of numerical values. That means all the elements that corresponds to a block situated on the outer ring receives the value 0. That is, each fingerprint image receives the same direction on the outer ring. The dimension of this type of feature vector is not reduced to 15 as the outer ring elements have a direction i.e. 0. The small feature vector for Fig. 4.2 is (0,0,0,0,0, 0,2,3,3,0, 0,2,2,2,0, 0,0,1,1,0, 0,0,0,0,0).

Feature vector with variance bins

When a direction for a small block is calculated, a certain amount of uncertainty about this direction exists. This uncertainty can be utilized in classification by calculating the variance of the dominant direction for that block.

The variance is computed perpendicular to the dominant direction for that block and in a similar manner as the sum of differences value (Section 4.1) which is used to determine the dominant direction from the subdirections. The calculation of the variance is shown in Fig. 4.5. For each position in the dominant direction the sum of differences is calculated along a line perpendicular to the dominant direction. In Fig. 4.5 the dominant direction is represented by the thick line with the seven thin lines representing the area where each calculation of the sum of differences value perpendicular to the dominant direction is performed. These seven values are then summed to form the variance for that block.

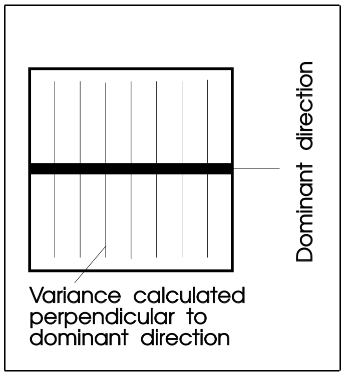


Figure 4.5: Calculation of the variance for a block

The feature vector is constructed by using the variance as a means to increase the number of bins in the histogram of directions. For instance: each direction has 3 bins associated with it, a low, medium and high variance. When the dominant direction is calculated for a block, the variance determines which bin of that direction (in the histogram of directions) is incremented. Fig. 4.6 shows this type of construction with the various variance bins for each quantized level.

Different feature vectors can be constructed by adjusting the number of bins for the variance and by adjusting the number of quantization levels for the dominant direction. Once again *large* and *small* feature vectors are defined. Large feature vectors correspond to the case where all the small blocks contribute to the histogram of variance bins i.e. 256 small blocks in the 256×256 images. Small feature vectors are vectors constructed similarly, but the outer ring of small blocks is ignored.

4.1.3 Simulation results

Each type of feature vector used in our directional experiments is labelled with a number to facilitate graphical representation. This association between the type and its label is shown in Table 4.1. The performance of the different classifiers for different feature vectors based on the directional image can be seen in Figures 4.7 to 4.16. The performance of a specific type of feature vector can be determined by reading its label from the figure and cross referencing it with Table 4.1.

Bin #	Variance
0	Low
	Medium
	High
1	Low
	Medium
	High
2	Low
	Medium
	High
3	Low
	Medium
	High

Figure 4.6: Variance type of feature vector

Label	Type of feature vector	Dimension of feature vector
0	Large histogram	Number of quantized levels
1	Small histogram	Number of quantized levels
2	Large direct	Number of blocks
3	Small direct	Number of blocks
4	4 variance bins, large	Number of quantized levels $\times$ 4
5	4 variance bins, small	Number of quantized levels $\times$ 4
6	8 variance bins, large	Number of quantized levels $\times$ 8
7	8 variance bins, small	Number of quantized levels $\times$ 8
8	12 variance bins, large	Number of quantized levels $\times$ 12
9	12 variance bins, small	Number of quantized levels $\times$ 12
10	16 variance bins, large	Number of quantized levels $\times$ 16
11	16 variance bins, small	Number of quantized levels $\times$ 16

Table 4.1: Label associated with each type of feature vector

Classification results using the neural-net classifier

All experiments were performed with the directional image as the basis for the feature vectors. Quantization levels of 8, 16, 24 and 32 were used.

The neural-net classifier was a 3-layer *perceptron* with a varying number of hidden neurons (1,3,5,...,15). A least-squares criterion function was used with conjugate gradient optimization to determine the weights [10].

Each experiment with a specific type of feature vector was repeated with 10 different randomly chosen initial sets of weight values. The greyscale images were divided into small blocks of dimension 16×16 pixels i.e. 256 small blocks in the 256×256 images. No preprocessing was done on any of the images prior to the extraction of the directional image.

The classification performance for the different types of feature vectors are shown in Figures 4.7 to 4.11. The average as well as the maximum classification performance over all trails for training and testing are shown.

Figures 4.7 to 4.11 and Table 4.1 show that the best classification (90%-maximum for the test set in Fig. 4.7) is obtained for a quantization level of 8 and by using the variance type of feature vector (type 5). The worst classification performance is obtained from the feature vector for direct comparison. The average test classifier performance in Fig. 4.7 starts near 60% for type 0 features (large histogram feature vectors), drops sharply for the large and small direct comparison feature vectors before climbing to a maximum for type 5 (4 variance bins, small feature vector). From this point to type 11 the average test classifier performance stays between the 60% and 70% levels. The average training performance was good (100%) except for the direct comparison feature vectors where the neural-net classifier was unable to draw meaningful decision boundaries in feature space. (This is illustrated by the bad classifier performance for types 2 and 3 on the test sets.) Because of the bad training, we assume that the direct comparison feature vector does not contain meaningful information which can be used by the neural net.

Fig. 4.8 shows the performance (maximum test and average test) of type 5 feature vector (4 variance bins, small type of feature vector) vs. the number of hidden neurons. The best performance is obtained for 9 and 15 hidden neurons. In Fig. 4.8 the classifier performance is very low for 1 hidden neuron. It then increases to a maximum of 90% (for 9 hidden neurons) as the number of hidden neurons is increased.

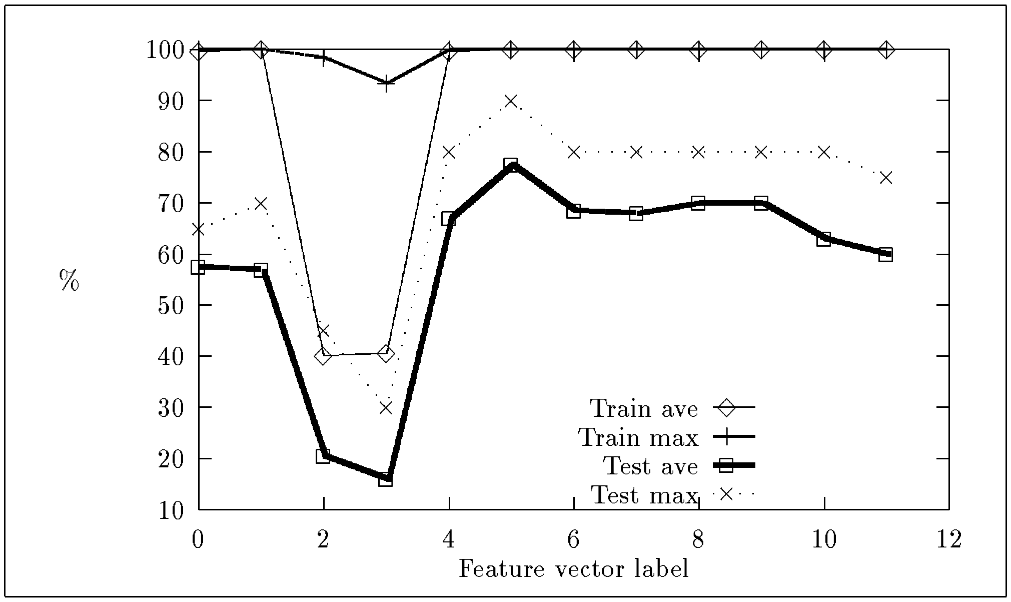


Figure 4.7: Neural-net classifier performance - 8 subdirections

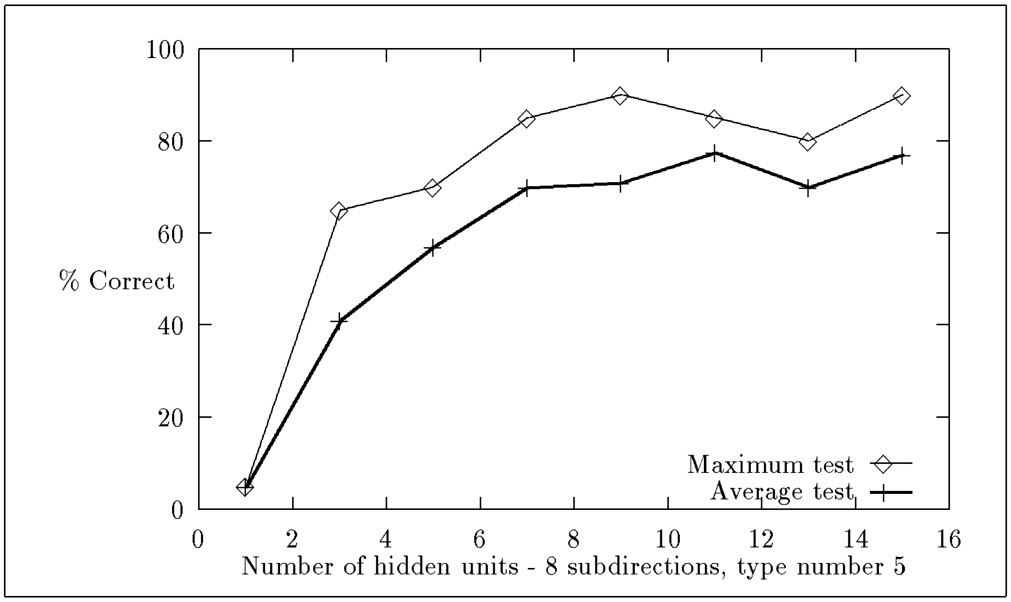


Figure 4.8: Neural-net classifier performance vs. number of hidden neurons

Sixteen subdirections were used to construct the feature vectors whose performance is shown in Fig. 4.9. The average test classifier performance again starts at 60% before dropping very low for the direct comparison type of feature vectors (type 2 and 3). From types 4 to 11 the average performance varies between 60% and 70%. Once again the neural net was able to find suitable minima during training, except for the direct-comparison types of feature vectors.

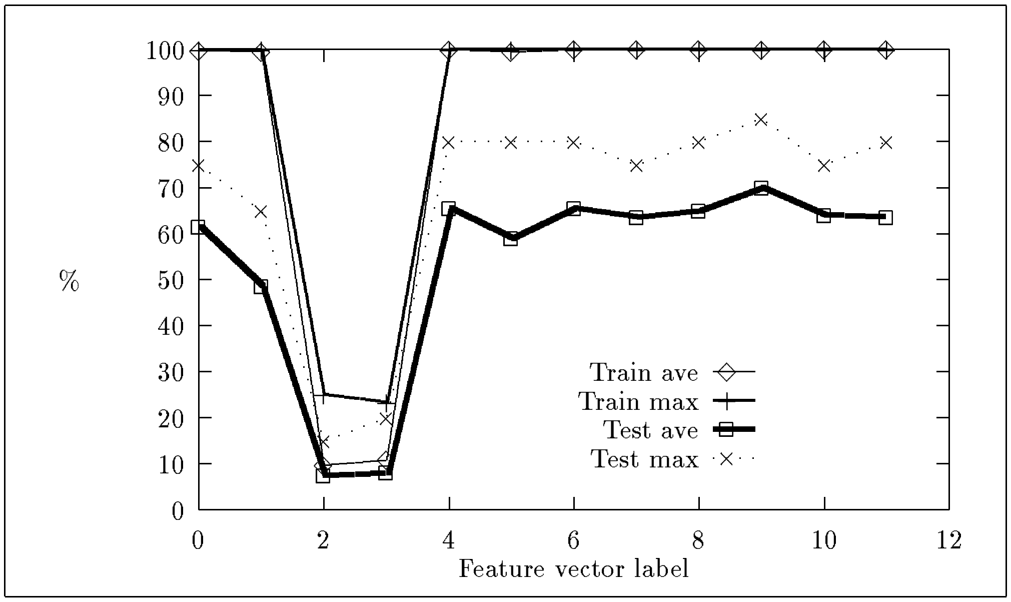


Figure 4.9: Neural-net classifier performance - 16 subdirections

The average test classifier performance for feature vectors constructed from directional images which were quantized into 24 levels is shown in Fig. 4.10. The classifier performance is similar to that obtained from a directional image constructed from 16 quantized levels. Once again the neural-net classifier fails on features using the direct comparison type of feature vector.

The neural-net classifier performance for features extracted from a 32 level quantized directional image is shown in Fig. 4.11. The direct comparison type of feature vector fails once again, the histogram-based feature vector's performance is near 50% and the variance-bins type of feature vector between 50% and 70%. In general the small type of feature vector performs worse than the large vector of the same type.

The best classifier performance obtained using the neural-net classifier and the directional image was 90%. In general the variance-based feature vector performed better than the histogram-based feature vector. The direct-comparison feature vector failed on all accounts. It is meaningful that the variance-based feature vectors perform better than the histogram-based feature vectors as the uncertainty that exists in the determination of the dominant direction is now used to improve the classifier performance.

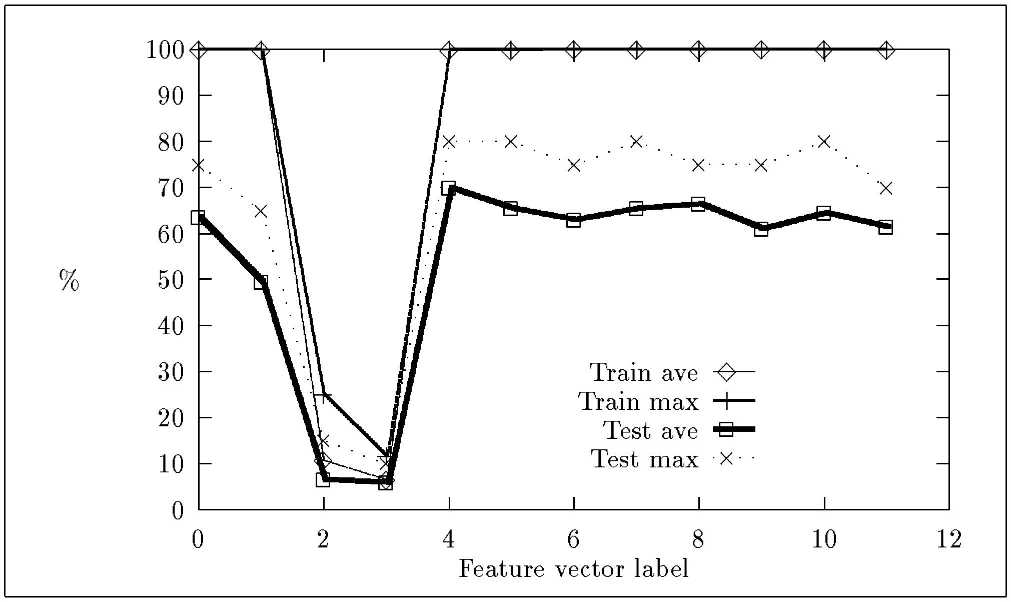


Figure 4.10: Neural-net classifier performance - 24 subdirections

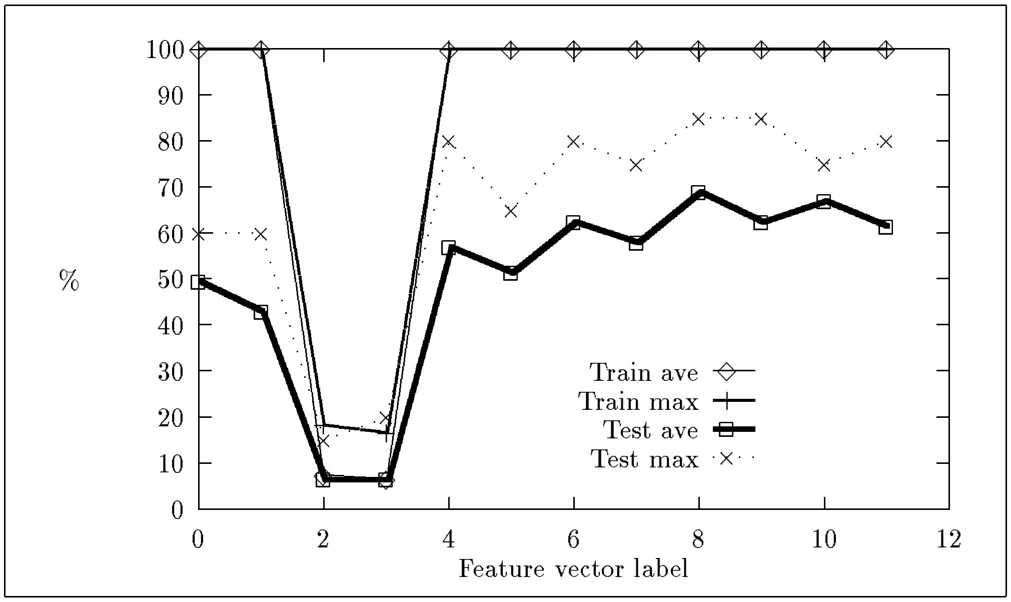


Figure 4.11: Neural-net classifier performance - 32 subdirections

Classification results using the linear classifier

Figures 4.12 to 4.15 show the classification obtained by using the linear classifier (Section 2.3.2) and the features as described in Table 4.1. The experiments were performed on feature vectors generated by using different quantized levels of 8,16,24 and 32.

Fig. 4.12 shows the linear classifier performance for each type of feature. The feature vectors for direct comparison fail, with the histogram-based between 40% and 50% and the variance-based between 50% and 70%.

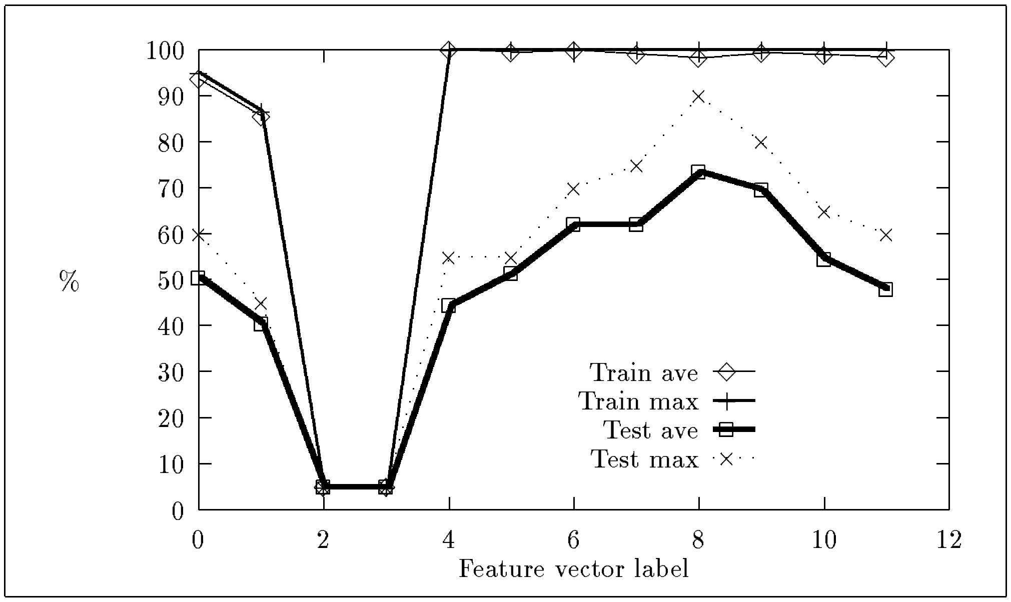


Figure 4.12: Linear classifier performance - 8 subdirections

In Figures 4.13 and 4.14 the performance for features extracted from a directional image obtained from 16 and 24 quantized levels are shown. The performance of the feature vectors for direct comparison is similar, failing on both accounts. In both cases classifier performance for features using the small type of construction is lower than that using the large type. This shows that meaningful information is lost by ignoring the outer ring of blocks in the construction of the feature vector.

Fig. 4.15 shows the classifier performance for directional images constructed from 32 quantized levels. Again, classifier performance is lower for all the small types of construction than those obtained from the large type. The direct comparison feature vectors failed again.

The best classification performance was 90% on the variance type of features with 8 quantized levels (type 8). The worst classification was obtained from the feature vectors using the direct comparison

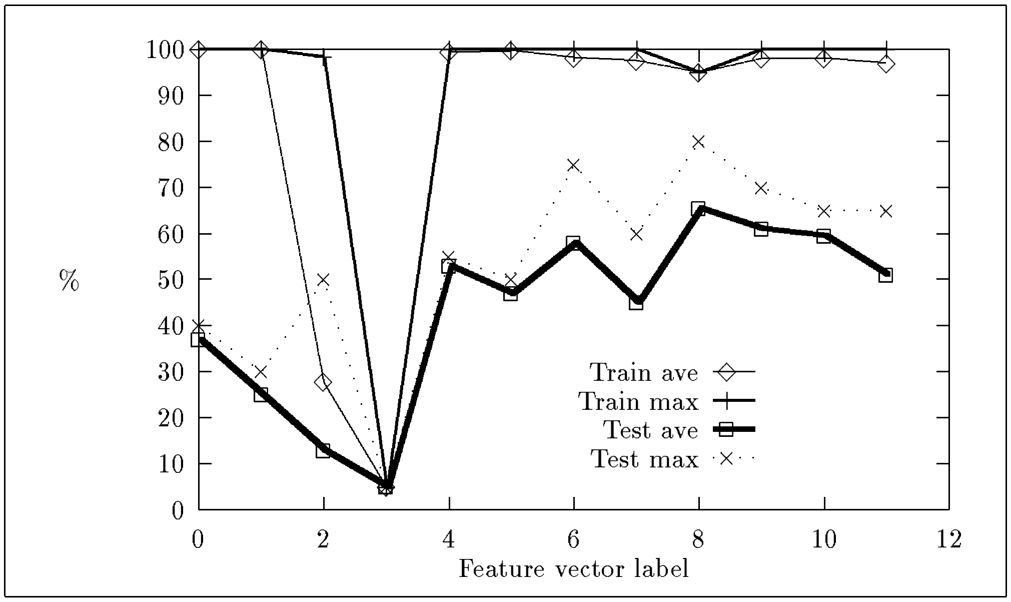


Figure 4.13: Linear classifier performance - 16 subdirections

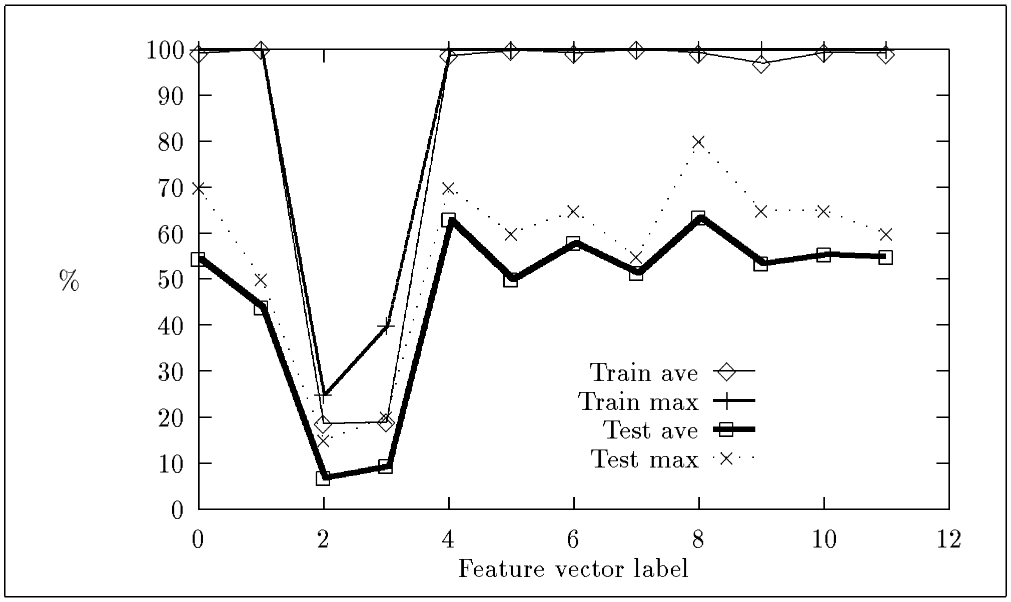


Figure 4.14: Linear classifier performance - 24 subdirections

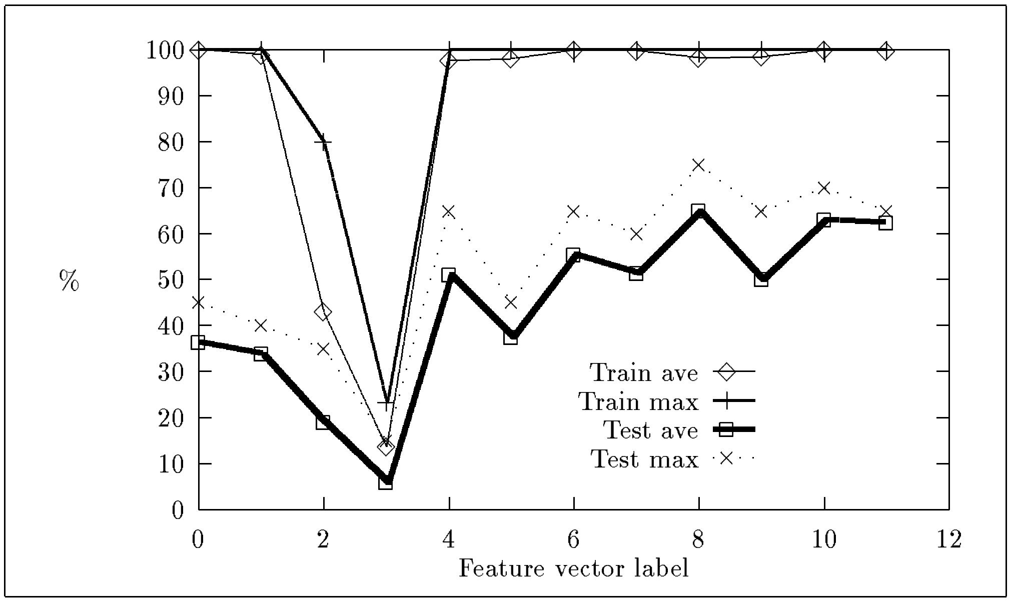


Figure 4.15: Linear classifier performance - 32 subdirections

type of feature vectors. For features extracted from the higher quantized directional images, the small type of construction performed worse than the large type of construction, which reinforces the hypothesis that meaningful information is lost by using the small type of construction with a linear classifier. We assume that the directional image is more noise sensitive at higher quantized levels, which leads to the drop in classifier performance.

Classification results using the nearest-neighbour classifier

We repeated the experiments using the features as in Table 4.1, using the nearest-neighbour classifier as described in Section 2.3.1. The results are shown in Fig. 4.16.

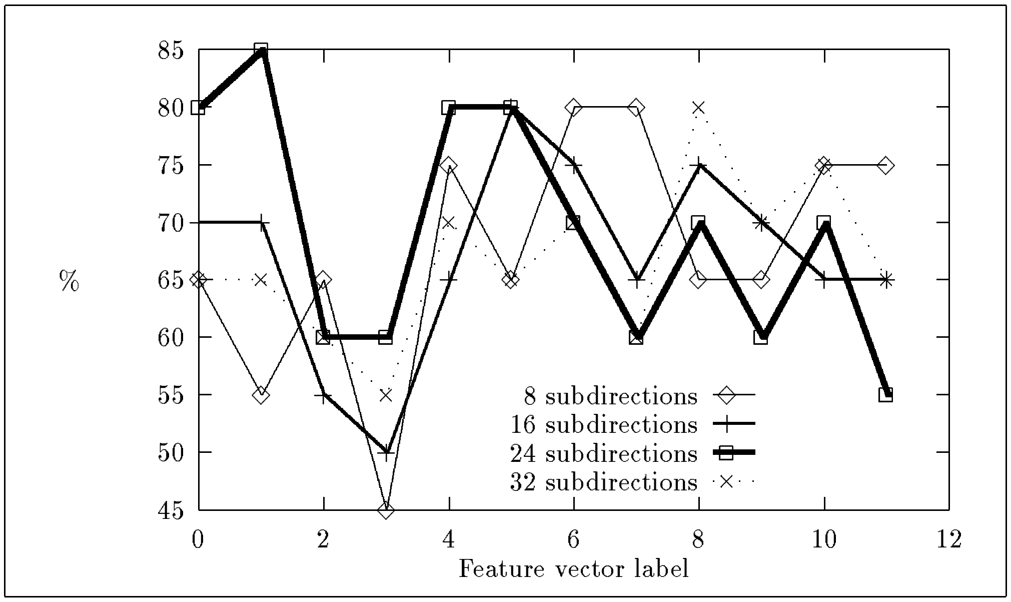


Figure 4.16: Nearest-neighbour classifier performance 8-32 subdirections

It can be seen from Fig. 4.16 that the best classification was 85% for 24 quantized levels using the small histogram feature vector (type number 1). This is not as good as the 90% classification which was obtained with the neural-net and linear classifiers. The worst classification was again on the direct-comparison type of feature vector. In general classifier performance was centered around the 70% mark with variations to either side. The best classification performance was on features using the small type of construction but no definite trend can be seen.

4.2 The Fourier transform and wedge-ring detector

4.2.1 Description of features

All the features that have been extracted thus far have been from the spatial domain. Unique features can also be obtained from the images transformed to the frequency domain [18].

The image is transformed from the spatial domain into the frequency domian by the application of a two-dimensional Fourier transform. The image is centered around the origin in the frequency domain by the addition of phase terms in the spatial domain. The mathematical operation for this is:

$$\exp^{(j\pi t)} f(t) \leftrightarrow F(w-\pi)$$

Features are extracted from the frequency domain by using a wedge-ring detector [19] whose center corresponds to the origin. An example of a wedge-ring detector with 4 rings (labelled 1,2,3 and 4) and 4 wedges (labelled A,B,C and D) is shown in Fig. 4.17. The summation of the elements (integration of the energy) in each ring or each wedge forms one element of the feature vector. Therefore the feature vector constructed from the wedge-ring detector in Fig. 4.17 is:

$$(\sum_A F(w_i), \sum_B F(w_i), \sum_C F(w_i), \sum_D F(w_i), \sum_1 F(w_i), \sum_2 F(w_i), \sum_3 F(w_i), \sum_4 F(w_i))$$

with $\sum_X F(w_i)$ the summation of all the elements in the frequency domain for wedge or ring X. By varying the number of rings and wedges the dimension of the feature vector is adjusted.

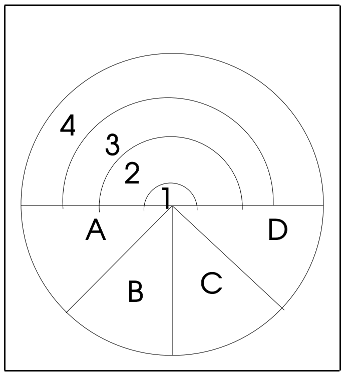


Figure 4.17: Wedge-ring detector

Manual fingerprint classification normally uses the minutiae. Minutiae in a fingerprint image is associated with high frequency components, while the ridges are at lower frequencies. Because of this, the wedge-ring detector is able to extract meaningful information from the frequency domain, as the high frequency components will differ from class to class because of differences in minutiae.

4.2.2 Simulation results

Experiments were performed with a varying number of wedges and rings. Each of these feature vectors with a different number of wedges and rings is labelled. This relationship between the different feature vectors and their respective labels are given in Table 4.2.

Experiments were performed on greyscale images, unsmoothed binary images, smoothed binary images and thinned binary images extracted from the smoothed binary images. Refer back to Fig. 3.6 for an

Label	Number of wedges	Number of rings	Dimension of feature vector
0	2	6	8
1	2	12	14
2	2	18	20
3	2	24	26
4	4	6	10
5	4	12	16
6	4	18	22
7	4	24	28
8	6	6	12
9	6	12	18
10	6	18	24
11	6	24	30
12	8	6	14
13	8	12	20
14	8	18	26
15	8	24	32

Table 4.2: Labels associated with variations in wedges and rings

example of an unsmoothed binary image, Fig. 3.7 for the smoothed version of Fig. 3.6 and Fig. 3.14 for the thinned binary images extracted from the smoothed binary image of Fig. 3.7.

Classification performance using the neural-net classifier

The same experimental setup was used in these experiments as in the experiments using the directional image and the neural-net classifier (Section 4.1.3). Each experiment was repeated with 10 different initial sets of weight values. Once again the number of hidden neurons was varied. We show the average classification performance as well as the maximum performance for each different feature vector on the test set.

Fig. 4.18 shows the classification performance on features extracted from the Fourier transform of the greyscale images. The maximum classification performance achieved was 95% for type 15. Table 4.2 shows that type 15 has 8 wedges and 24 rings to give the feature vector a dimension of 32. Fig. 4.19 shows the performance versus the number of hidden neurons for type 15. The best performance is obtained with 11 hidden neurons.

The average test performance for each feature vector which has 6 rings (types 0,4,8 and 12) is much lower than the performance obtained with more rings (12,18 and 24), although classifier performance drops for a high number of rings and wedges (24 rings with 6 wedges and 24 rings with 8 wedges).

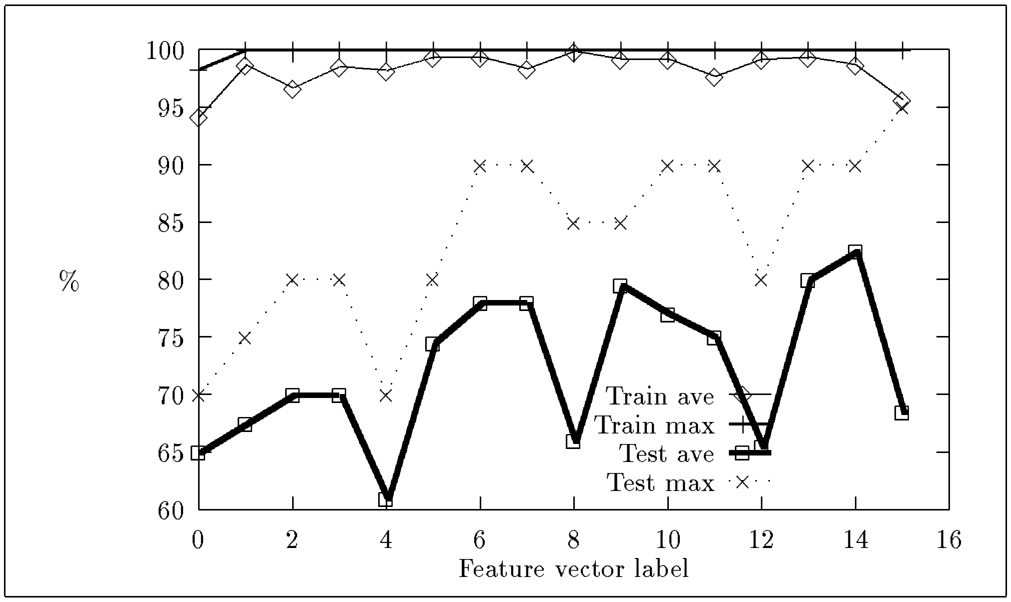


Figure 4.18: Neural-net classifier performance - greyscale images

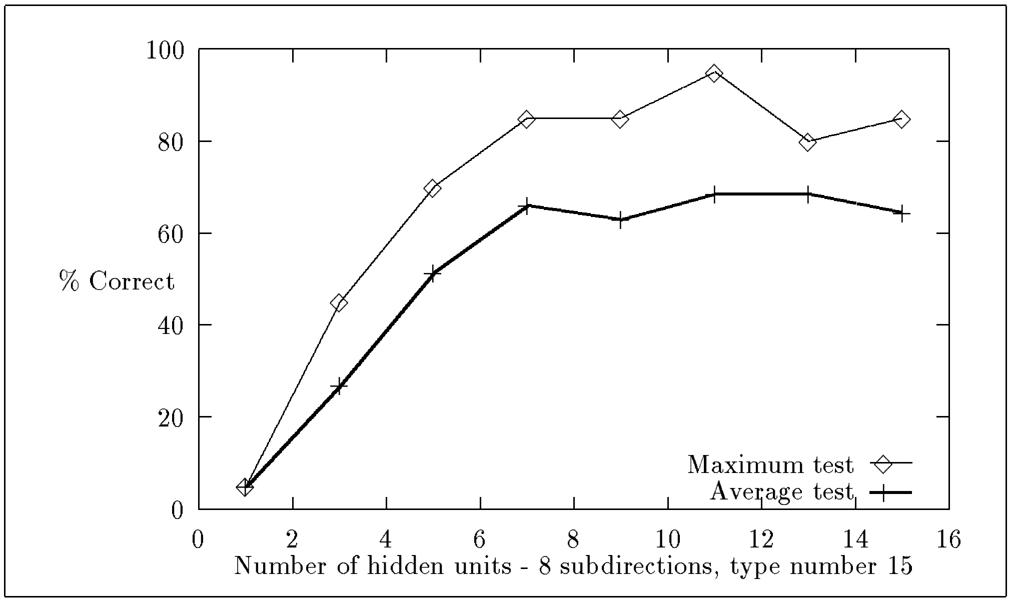


Figure 4.19: Neural-net classifier performance vs. hidden neurons

The experiments were repeated on features extracted from the frequency domain of the unsmoothed binary images. The maximum classification shown in Fig. 4.20 is again 95%. This is not higher than the maximum classification achieved on the greyscale features, but more feature vectors reach this classification performance. They are types 3,9,10,11,13 and 14. The average test performance shown in Fig. 4.20 shows that performance for 6 rings in the wedge-ring detector is again lower than the performance with the higher number of rings for each number of wedges.

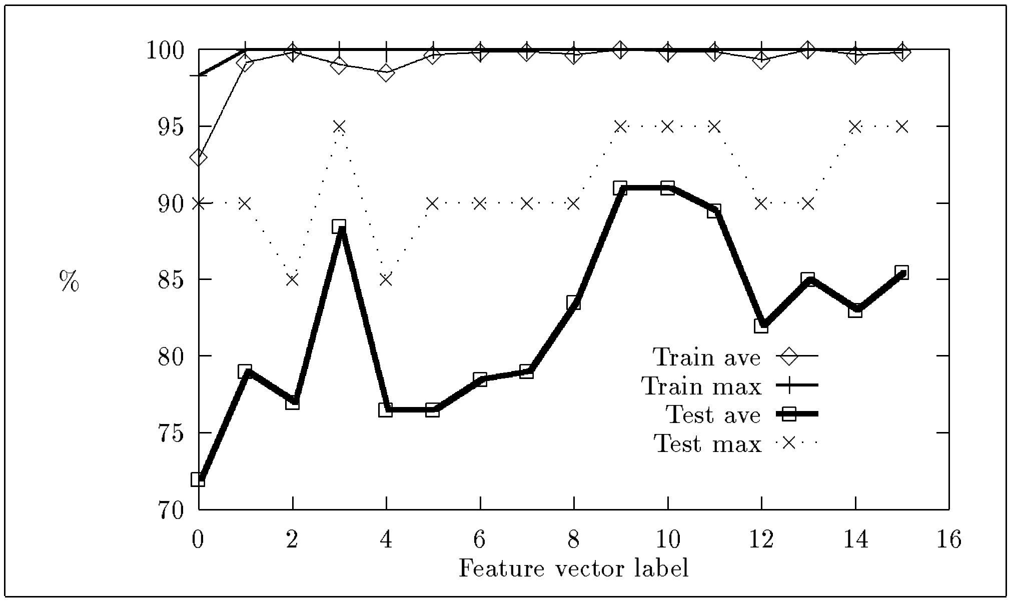


Figure 4.20: Neural-net classifier performance - unsmoothed binary images

In the next experiment features were extracted from the Fourier transform of the smoothed binary images. The results of our experiments are shown in Fig. 4.21. The classification performance obtained in these experiments are notably higher than those obtained from the greyscale images and the unsmoothed binary images. The best classification performance was 100% for types 1,2,3,9,10,11,13 and 14. The average test performance is the highest for features using 6 wedges. Once again the average test performance is low for a small number of rings, but increases as the number of rings is increased.

Our final experiment used features extracted from the frequency domain of the thinned binary images. The classification performance for these features are shown in Fig. 4.22. The overall classification performance dropped from the 100% achieved in our previous experiments to a best performance of 95% for type numbers 9,10 and 11.

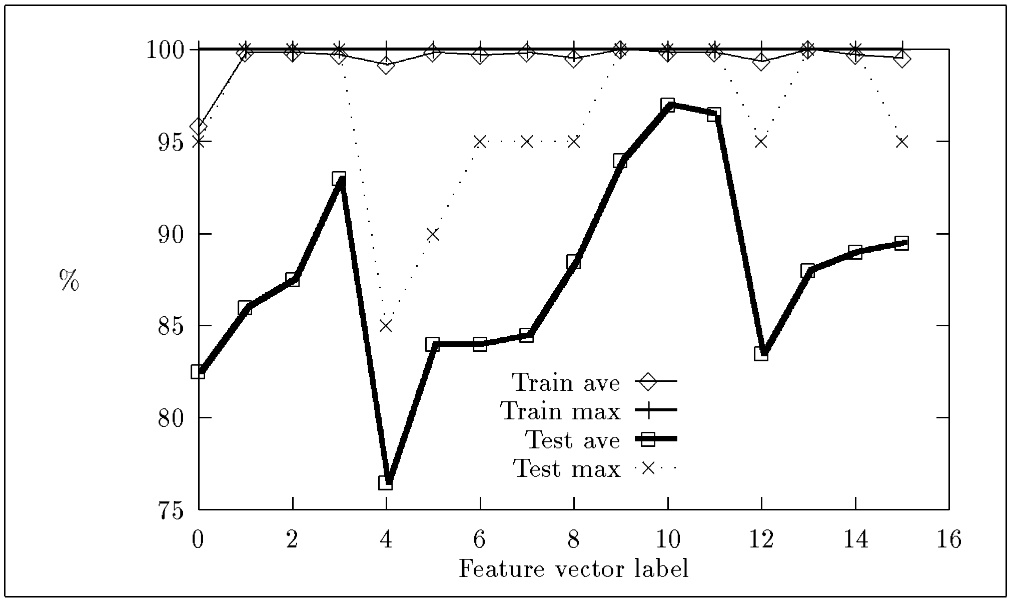


Figure 4.21: Neural-net classifier performance - smoothed binary images

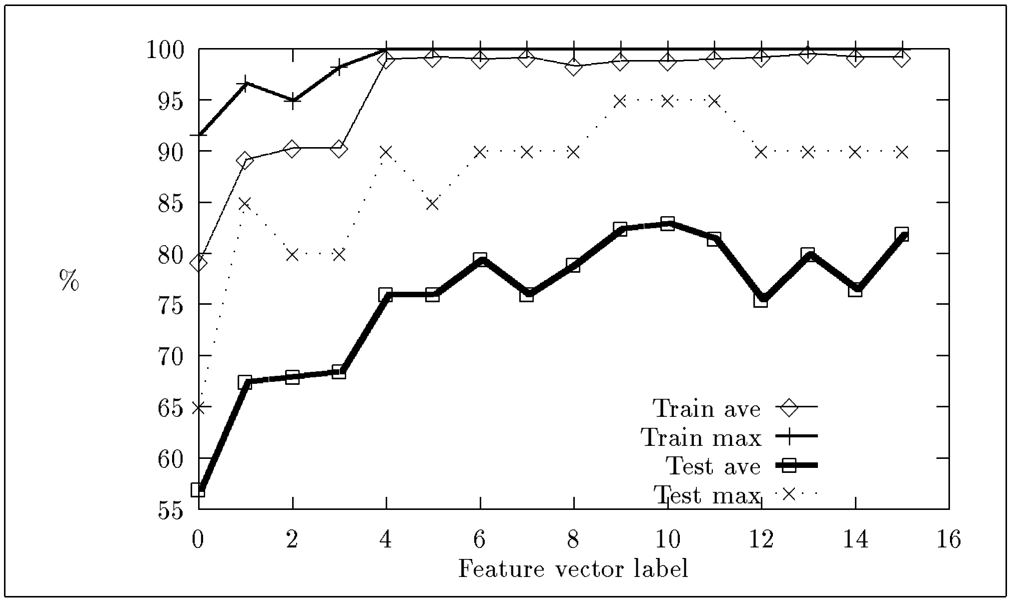


Figure 4.22: Neural-net classifier performance - thinned binary images

Classification results using the linear classifier

The classifier performance obtained on the features of Table 4.2 extracted from the Fourier transform of the greyscale images and by using the linear classifier is shown in Fig. 4.23. The best classification performance is 90% for types 13 and 14. Average test performance increases for the variation in the rings for a specific number of wedges.

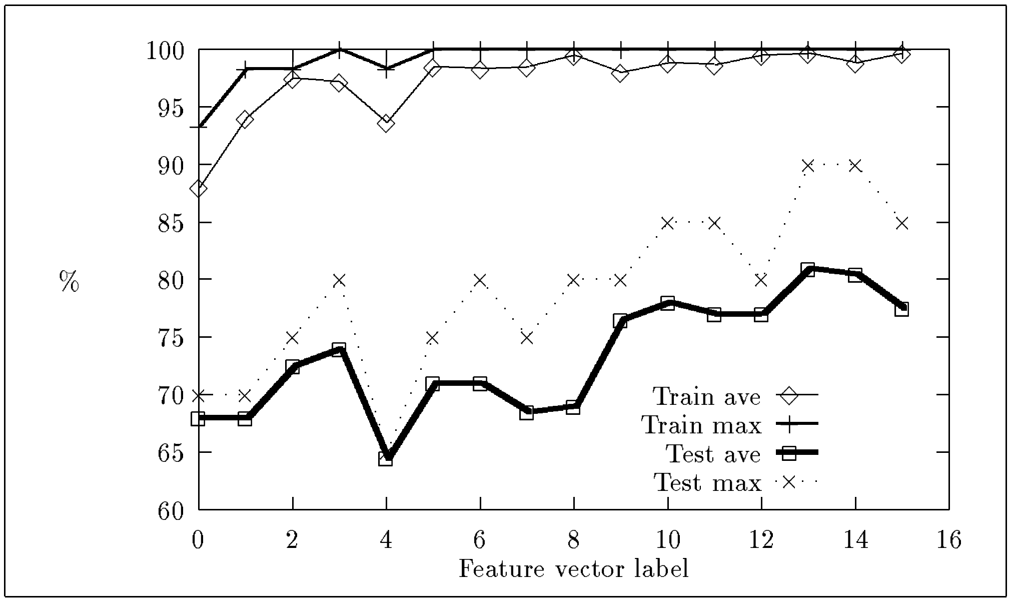


Figure 4.23: Linear classifier performance - greyscale images

Fig. 4.24 shows classification performance for features extracted from the Fourier transform of the unsmoothed binary images. The best classification improved to 95% on types 9,10,11,13,14 and 15. Fig. 4.24 shows that feature vectors constructed with 6 rings generally perform worse than feature vectors with a higher number of rings for a specific number of wedges.

Our next figure (Fig. 4.25) shows the classification performance on features extracted from the Fourier transform of the smoothed binary images. Again 100% classification was achieved for types 9,10,13,14 and 15. Once again classifier performance improves with the higher number of rings for a specific number of wedges. Unfortunately this trend does not continue to a very high number of rings, as classifier performance drops for the highest number of rings, independently of the number of wedges.

The improvement in maximum classification performance by the application of the preprocessing algorithms suggests that similar classification performance can be obtained from the features extracted from the Fourier transform of the thinned binary images. Unfortunately this is not the case. Fig. 4.26 shows us that the general classification performance drops, and that the maximum correct classification

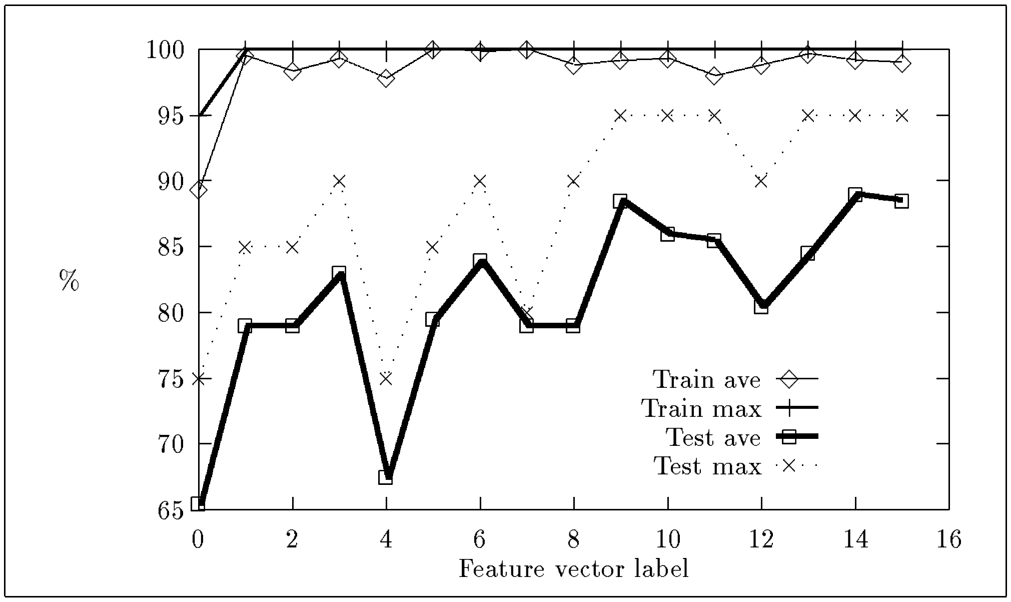


Figure 4.24: Linear classifier performance - unsmoothed binary images

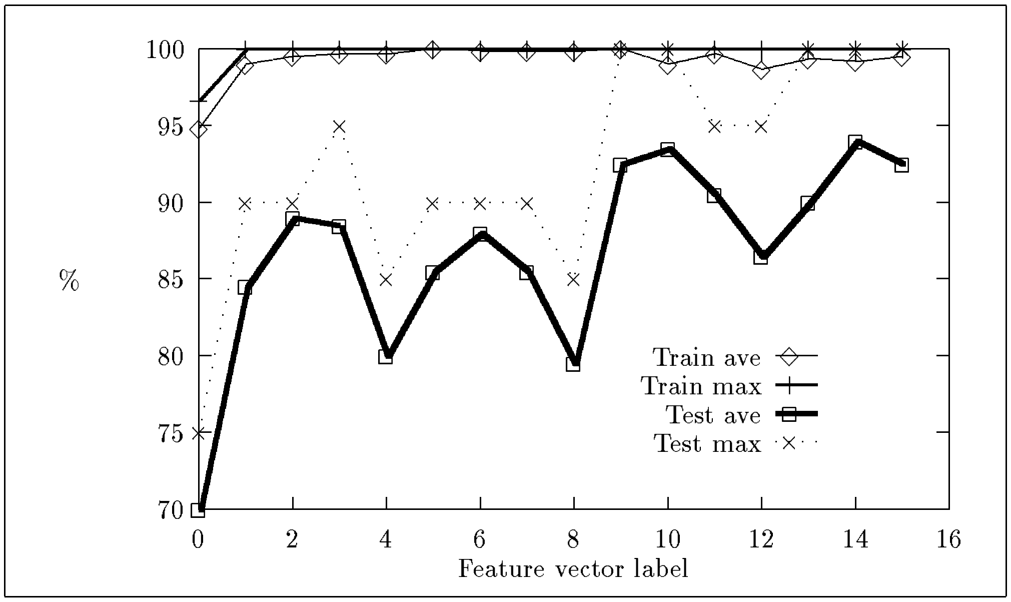


Figure 4.25: Linear classifier performance - smoothed binary images

obtained is only 90% for types 8,10,11 and 13. We assume that the reason for this drop in classifier performance can be contributed to the loss of uniqueness in the frequency domain through the thinning of the image.

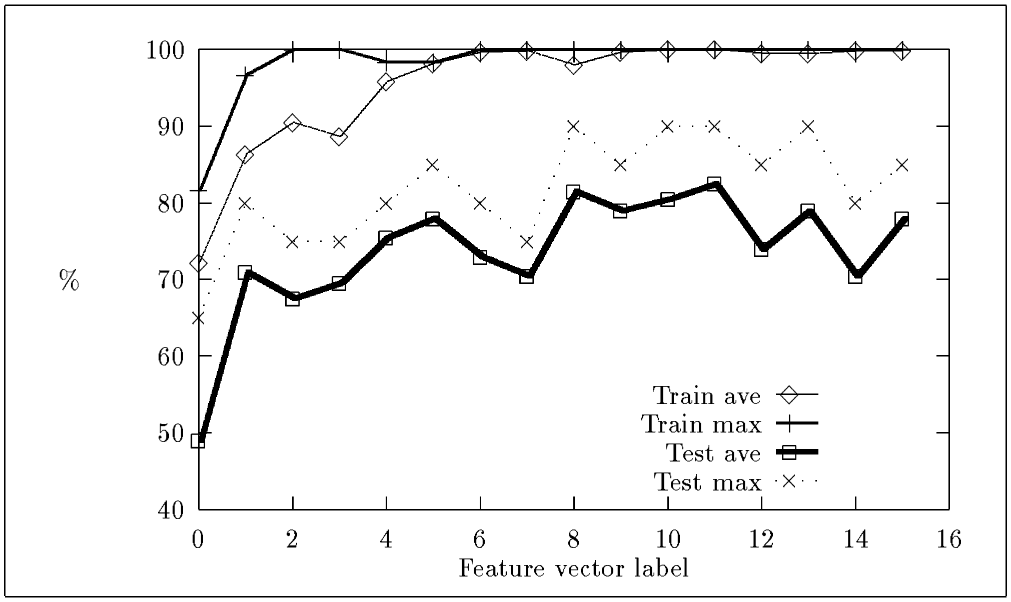


Figure 4.26: Linear classifier performance - thinned binary images

Classification results using the nearest-neighbour classifier

In this experiment we used the same features (Table 4.2) but used the nearest-neighbour classifier.

Fig. 4.27 shows that the best classifier performance of 100% was obtained from features extracted from the smoothed binary image. The types for which this excellent classification performance was obtained are 9,10,13 and 14. The worst classification performance was on the smoothed thinned binary images. The classification performance obtained from the smoothed binary images increase as the number of wedges and rings increase, but an upper limit exists where performance drops if the dimension of the feature vector gets too high.

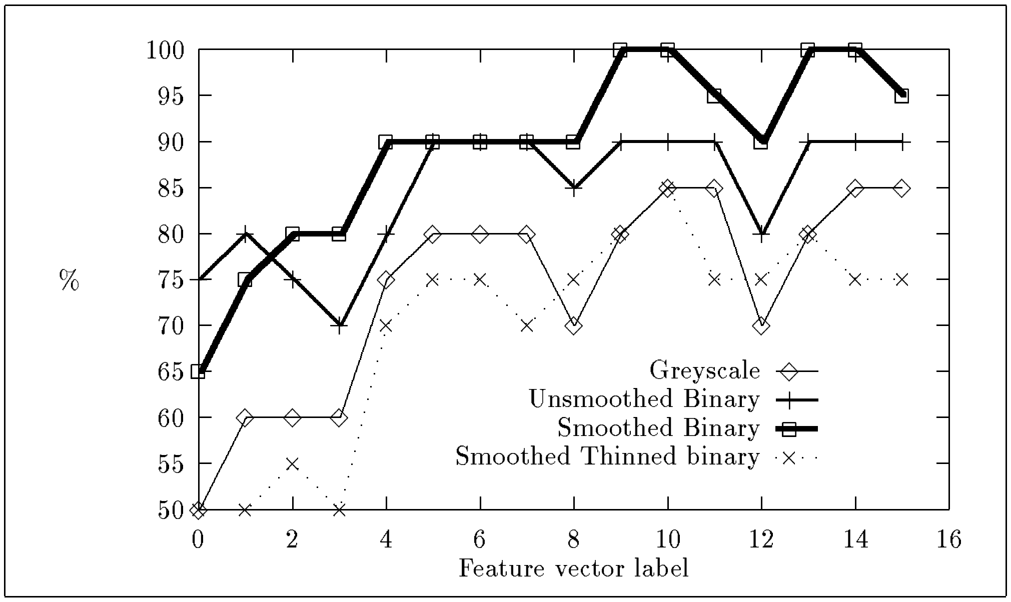


Figure 4.27: Nearest-neighbour classifier performance

In all our experiments the maximum performance on the test set was obtained from features extracted from the frequency domain of the smoothed binary images. All three classifiers (neural-net, linear and nearest-neighbour) were able to classify the test set correctly (100%). The general trend in classifier performance suggests that more than 6 rings perform better for any number of wedges, but that an upper limit exists for the dimension of the feature vector, as classifier performance drops for feature vector types with high dimensions.

4.3 Classification using minutiae

The minutiae are small unique features on a fingerprint. These features are the division of a ridge into two or more ridges, known as forks, ridge endings, islands or loops, crossovers and bridges. Examples of such minutiae are shown in Fig. 4.28.

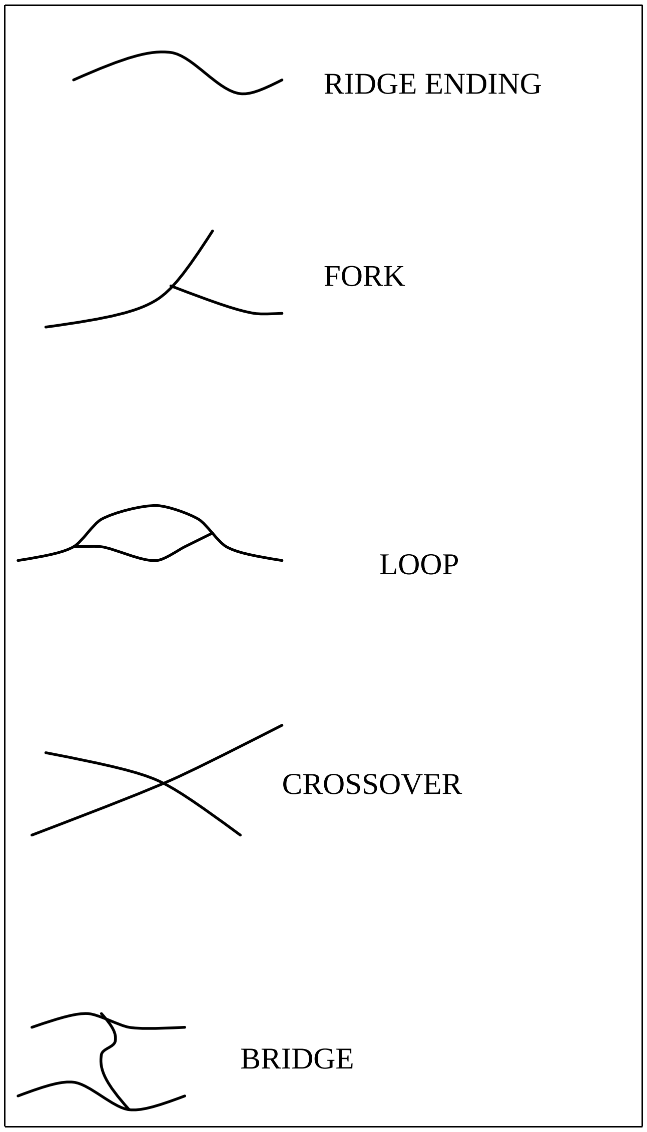


Figure 4.28: Typical minutiae used in fingerprint recognition

By using these minutiae in a structural manner (matching structures such as bridges with one another) [5] classification which is robust with respect to transformations such as rotation and translation as well as distortions can be obtained.

Unfortunately we were unable to use the minutiae for feature-based recognition as it is very difficult to identify the minutiae reliably in a fingerprint image which is not of a good quality. The original greyscale image must be clear and of a good quality otherwise false minutiae will be extracted. The consequence of the extraction of false minutiae is false classification. Prior to the extraction of the minutiae extensive preprocessing is needed because these minutiae can only be extracted from a thinned binary image such as in Fig. 3.14. Unfortunately not all the images are as clear and reliable as this one.

Another problem presents itself in that there is no guarantee that the part of the fingerprint available to the classification system contains any minutiae. Usually the middle portion of the image is available which does not have many minutiae. Therefore no reliable classification can be done on such an image.

Due to the problems presented no meaningful classification results were obtained. This method does have merit but good quality prints of the complete finger are required before any successful experiments can be conducted.

In our next chapter (Chapter 5) we present results obtained in the other main approach to fingerprint recognition we investigated. This approach is the correlation-based branch of Fig. 2.4.

Chapter 5

Correlation-based recognition

For good classification by means of a correlation classifier extensive preprocessing is needed to transform the greyscale image into a usable binary image. A binary image is used during correlation so that the background of an image does not affect the classification performance. The influence of darker or lighter images is also removed by using the binry image during correlation. The preprocessing we implemented consisted of the three stages shown in Chapter 3: binarization of the greyscale image using the recursive greyscale edge follower (Fig. 3.6), the smoothing of the binary image using an adaptive window concept (Fig. 3.7), and the thinning of the smoothed image into its skeleton (Fig. 3.14). The images obtained by the application of these preprocessing algorithms are noiseless binary skeletons of the original images.

Correlation in the spatial domain is very time consuming, especially for large images as was shown in Section 2.3.4. For this reason we implemented the correlation classifier in the frequency domain. Correlation in the spatial domain is equivalent to the inverse Fourier transform of the product of the Fourier transforms of the images being correlated. For two images f_1 and f_2, this can be mathematically expressed as follows:

$$f_1(x, y) \otimes f_2(x, y) \leftrightarrow F_1(u, v) \times F_2^*(u, v)$$

where $\otimes$ denotes the correlation in the spatial domain, * denotes the complex conjugate in the frequency domain and F_1 and F_2 are the Fourier transforms of f_1 and f_2 respectively.

In the correlation classifier an unknown test image is classified in the following manner:
The Fourier transform is computed for the unknown test image. This Fourier transform is multiplied

with each of the known "training" images' Fourier transforms (which are calculated and stored before hand). The inverse Fourier transform is computed for this product, resulting in a two-dimensional correlation function. The correlation image is then searched for its largest value. This value is normalized with the number of binary ones in the unknown test image. This normalized value is known as the correlation value. The process of multiplying, taking the inverse Fourier transform, searching and normalizing is repeated for all the "training" images. Finally the largest correlation value between the test image and and all the training images is found and the test image is labelled as belonging to that class for which the correlation value was the largest. This procedure was first suggested for optical implementation [20].

We normalize the largest value of the inverse Fourier transform with the number of binary ones in the test image so that the maximum correlation between two images cannot be higher than 1.

5.1 Simulation results

We performed experiments with the correlation classifier on binary images at different stages in the preprocessing as described in Chapter 3. The first experiment was performed on unsmoothed binary images. This was followed by an experiment on the smoothed binary images. The last two experiments were conducted on thinned binary images. The first of these two experiments used the thinned binary images extracted from unsmoothed binary images, while the last one used the thinned binary images extracted from the smoothed binary images. We describe the results obtained from the experiments in the following subsections and show the labels as well as correlation values obtained (Tables 5.1 to 5.4).

5.1.1 Unsmoothed binary images

In the first experiment performed, binary images without any smoothing or thinning were used. The correlation classifier was able to correctly classify 75% of the test images. Table 5.1 shows for each class the correct class, the label assigned to it by the correlation classifier, and the correlation value that led to the specific labeling for that class. The classes incorrectly labelled were classes 8 (labelled as 1), 9 (labelled as 5), 12 (labelled as 10), 13 (labelled as 5) and 19 (labelled as 5). It can be seen that the correlation values are low (0.61 and lower), which suggests that the matches found between the test and a training image are not very reliable.

	Unsmoothed binary	
Correct class	Labelled as	Correlation value
1	1	0.39
2	2	0.59
3	3	0.48
4	4	0.53
5	5	0.61
6	6	0.60
7	7	0.60
8	1 $\times$	0.46
9	5 $\times$	0.42
10	10	0.46
11	11	0.54
12	10 $\times$	0.55
13	5 $\times$	0.56
14	14	0.58
15	15	0.37
16	16	0.53
17	17	0.53
18	18	0.48
19	5 $\times$	0.51
20	20	0.49
Performance:	75% correct	

Table 5.1: Correlation classifier performance, unsmoothed binary images

	Smoothed binary	
Correct class	Labelled as	Correlation value
1	1	0.42
2	2	0.62
3	3	0.53
4	4	0.56
5	5	0.63
6	6	0.61
7	7	0.61
8	1 ×	0.49
9	5 ×	0.44
10	10	0.50
11	11	0.55
12	10 ×	0.58
13	5 ×	0.59
14	14	0.62
15	15	0.39
16	16	0.56
17	17	0.57
18	18	0.51
19	5 ×	0.53
20	20	0.53
Performance:	75% correct	

Table 5.2: Correlation classifier performance, smoothed binary images

5.1.2 Smoothed binary images

In the next experiment smoothed binary images were used. The classifier performance did not improve from that obtained from the unsmoothed binary images, i.e. 75% was correctly classified (see Table 5.2). The same classes were incorrectly labelled in this experiment as in the previous experiment. The only difference in results between these two experiments is the correlation values obtained in the experiments. In general the correlation values were slightly higher for the smoothed binary images than that of the unsmoothed binary images. In the previous experiment the average correlation value was 0.51 compared to 0.54 of the current experiment.

	Unsmoothed binary's skeleton	
Correct class	Labelled as	Correlation value
1	1	0.16
2	2	0.25
3	3	0.19
4	4	0.18
5	5	0.24
6	6	0.22
7	7	0.17
8	8	0.16
9	9	0.12
10	10	0.14
11	11	0.15
12	12	0.15
13	5 $\times$	0.15
14	14	0.16
15	15	0.13
16	16	0.15
17	17	0.17
18	18	0.18
19	5 $\times$	0.15
20	20	0.19
Performance:	90% correct	

Table 5.3: Correlation classifier performance, unsmoothed thinned binary images

5.1.3 Thinned images extracted from unsmoothed binary images

In our next experiment the skeletons of the unsmoothed binary images were used. The classifier performance improved from 75% obtained in the previous experiment to 90% in this experiment, as shown in Table 5.3. Only two classes were labelled incorrectly (13 labelled as 5, and 19 labelled as 5). The improvement in classifier performance demonstrates the merit of using preprocessing algorithms in pattern recognition.

	Smoothed binary's skeleton	
Correct class	Labelled as	Correlation value
1	1	0.16
2	2	0.24
3	3	0.18
4	4	0.17
5	5	0.21
6	6	0.22
7	7	0.17
8	8	0.16
9	9	0.12
10	10	0.13
11	11	0.16
12	12	0.13
13	13	0.14
14	14	0.15
15	15	0.12
16	16	0.14
17	17	0.17
18	18	0.18
19	5 $\times$	0.14
20	20	0.17
Performance:	95% correct	

Table 5.4: Correlation classifier performance, skeleton from smoothed binary images

5.1.4 Thinned images extracted from smoothed binary images

Our final experiment using the correlation classifier used skeletons extracted from smoothed binary images. The classifier performance improved once again, from 90% obtained from skeletons extracted from unsmoothed binary images to 95% using the skeletons of the smoothed binary images. Table 5.4 shows that the only class incorrectly labelled is class 19 (labelled as class 5). Visual inspection of class 5 reveals that it is a very general type of fingerprint without any major features.

The results clearly show the improvement obtained in classifier performance by the application of the preprocessing algorithms. However, the computational burden of computing the Fourier transform for each test image and each training image, followed by an inverse Fourier transform calculation of the product of these two Fourier transforms makes the correlation classifier unsuitable for use in a fast fingerprint recognition system.

In our final chapter (Chapter 6) we compare the results obtained in the various experiments as described in Chapters 4 and 5. Finally, a conclusion is made and we propose possible extensions to some of the approaches we have investigated.

Chapter 6

Comparison

In this chapter a comparison between the various results obtained in experiments from Chapters 4 and 5 is made. The "best" classification scheme in terms of computation expenses and classification performance is identified. Possible extensions to some of our schemes are proposed and finally a conclusion is made.

6.1 Classification performance

In the feature-based approach described in Chapter 4 the best classification performance (100%) was obtained for features extracted from the two-dimensional frequency transform of the smoothed binary images. All three the classifiers (neural-net, linear and nearest-neighbour) were able to classify the test set correctly. This means that the classes are well seperated in this feature space.

Features extracted from the Fourier transform of the unsmoothed binary image performed better with all the classifiers than the features extracted from the greyscale and thinned binary images.

Classification performance obtained from the correlation classifier was worse compared to the 100% test classification obtained from the features of the frequency plane. The best classification performance using the correlation classifier was on the skeleton of smoothed binary images (95%). The performance of this classifier on the smoothed binary images and unsmoothed binary images was similar (75%), but not as good as its performance on the skeletons of images (smoothed binary's skeleton 95% and unsmoothed binary's skeleton 90%). The classification obtained from the thinned binary images was good, but the computational burden associated with the correlation classifier severely handicaps this approach compared to the feature-based approach.

The best performance obtained from features extracted from the directional image were those based on the variance-bins type of feature vector. The best classification performance obtained was 90% with the neural-net classifier using a feature vector constructed from a directional image which was quantized into 8 subdirections. The same classification performance was obtained with these features using the linear classifier. The best classification performance obtained with the nearest-neighbour classifier was 85%. To obtain this, a histogram-type of feature vector was used, constructed from a directional image quantized into 24 subdirections.

6.2 Computational burden

6.2.1 Feature extraction and preprocessing

To minimize the computation time with respect to feature extraction, the best approach for classification is the correlation-based approach because no features are extracted from the images. However, extensive preprocessing is needed to transform the greyscale image into a thinned binary image which gave the good classification performance of 95%.

The next best approach is that using the directional image. No preprocessing of any kind is conducted on the image before the features are extracted. However, computation time of the features can increase if a high number of quantized levels are used.

Using features extracted from the frequency plane of the images can be quite expensive. For features to be extracted, a Fourier transform has to be performed on a preprocessed image (preprocessing includes binarization, smoothing and thinning). The calculation of the Fourier transform as well as the preprocessing make this method very expensive to use.

With regard to the preprocessing and the feature extraction the best method is that based on the directional image. No preprocessing is needed and the time spent in extracting the features is minimal if the number of quantized levels is kept low.

6.2.2 Classifiers

To identify the best classifier a distinction between the time spent in training and the time spent to classify a test vector must be made. The general classifier performance does not suffer if the training time is extensive as the training normally occurs off-line. Therefore the only real measurement occurs during the classification of a test vector.

During training the nearest-neighbour classifier is the fastest as no real training occurs. The linear classifier needs training, that is a suitable minimum must be found for the weight matrix **W**. This does take a certain amount of time but not as much as that needed to train the neural-net classifier. During training of the neural net iterations with different initial values for the weight matrix **W** and different numbers of hidden neurons must take place. This can be very time consuming. Fortunately the training is done off-line, therefore this problem does not handicap the neural-net classifier in terms of classifier speed.

During general use the fastest classifier is the linear classifier, followed by the neural-net classifier. The slowest classifier is the nearest-neighbour classifier. The linear and neural-net classifiers are very fast. A feature vector can be labelled as belonging to a class as soon as the vector propagated through the classifier. The linear classifier can achieve this a bit faster than the neural-net classifier as the number of propagating steps is less for the linear classifier than that of the neural-net classifier. To obtain a classification from the nearest-neighbour classifier numerous calculations have to be performed. This severely handicaps the nearest-neighbour classifier.

The correlation classifier cannot compare with any of the other classifiers in terms of classification speed. The calculation of the Fourier- and inverse Fourier transforms makes this classifier too slow when implemented digitally.

6.3 The best

Features extracted from the frequency plane of smoothed binary images using the linear classifier are to our meaning the best suited for practical implementation. The data generation and Fourier transform can be implemented with optics, which can be extremely fast. The binarization, smoothing, feature extraction and classification can be done with dedicated hardware.

Our second choice is the correlation classifier using skeletons of smoothed binary images. The process of correlation is time consuming but once again a dedicated system can successfully overcome this problem.

The third option, the variance-based feature vector extracted from the directional image, with a neural-net classifier is by no means totally unsuited for use in a fingerprint classification system. The ease with which features are extracted without any preprocessing makes this approach ideal for a classification system. However, these features on their own are inadequate and must be combined with some other features for reliable classification.

6.4 Extensions to the system

To improve classification in general, the method of data generation has to be improved. The problems that exist with the current optical system such as the contrast and focus can be overcome with very high quality prisms and lenses. Once the general quality of the fingerprint has been improved, classification performance will improve, because noise which disrupts performance will be suppressed.

Once the data generation method is fast and reliable, classification using the minutiae of a fingerprint can be used with a neural-net classifier for classification purposes. Using the minutiae to construct a feature vector will greatly enhance the classifier performance. All indications are that these minutiae-type of feature vectors should be very robust and well-suited for classification purposes. Another possibility is the combination of different types of feature vectors. A good example could be the combination of features extracted from the directional image combined with minutiae (the directional image's feature vector used as primary classification and the final confirmation done by using the minutiae).

6.5 Summary and Conclusion

In this thesis the problem of fingerprint recognition as well as different possible solutions were described. Methods of data generation, preprocessing, feature extraction and classification were investigated.

Two methods of data generation were described, the well known ink-and-paper method and an optical data generation method using a $90°$ prism. Various preprocessing algorithms were developed. They include a binarization algorithm and a smoothing algorithm. These two algorithms are used in conjunction with a line-thinning algorithm to produce clear and smooth thinned binary images.

Two approaches to fingerprint recognition were investigated, a correlation approach and a feature-based approach. In the feature-based approach, features extracted from the directional image, features extracted from the frequency domain and features extracted from the fingerprint images themselves were considered. These features were tested with various classifiers (neural-net, linear and nearest-neighbour) for performance. Features extracted from smoothed binary images' frequency domain and a linear classifier produced 100% classifier performance.

In the correlation approach, images at different stages of preprocessing were used to test classifier performance. Smoothed thinned binary images performed the best for this approach (95%).

The performance of all the methods investigated does not match those obtained by manual methods using minutiae. The reason for this is the limited intelligence built into the classifier schemes. This does not compare favourably with the intelligence of a human using one of the manual methods, especially if the quality of the fingerprint image is suspect. However, the classification speeds obtained in general are much faster than those that could be obtained using manual methods.

Good classification performance was obtained in general. However, the main problem which hampers the classification of a fingerprint image is still not solved. This is data generation, that creates problems for both the methods described in this thesis. Some of the problems can be solved with preprocessing techniques but much better performance can be obtained if the images are clear and of a good quality to start with. In general, good classification of fingerprint images is possible, but performance will only be perfect once the quality of the original print is of an acceptable standard.

With the quality of fingerprints presently obtainable with optical data generation and the algorithms investigated in this thesis, successful fingerprint classification systems can be developed for several application areas such as access control. However, the success rate of the algorithms does not yet render them adequate for high security applications.

Bibliography

[1] T. Ch and Malleswara Rao. Feature extraction for fingerprint classification. *Pattern Recognition*, 8:181–192, 1976.

[2] Bijan Moayer and King-Sun Fu. A tree systems approach for fingerprint pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(3):376–387, May 1986.

[3] A. K. Majumdar M.R. Verma and B. Chatterjee. Edge detection in fingerprints. *Pattern Recognition*, 20(5):513–523, 1987.

[4] M.R. Verma and B. Chatterjee. Partial fingerprint pattern classification. *J. INSTN. Electronic & Telecom. Engrs.*, 3(1):28–33, 1989.

[5] A.K. Hrechak and J.A. McHugh. Automated fingerprint recognition using structural matching. *Pattern Recognition*, 23(1):893–904, 1990.

[6] Orit Baruch. Line thinning by line following. *Pattern Recognition Letters*, 8:271–276, 1988.

[7] R.E. Gaensslen Peter R. De Forest and Henry C. Lee. *Forensic Science, An Introduction to Criminalistics.* McGraw-Hill, New York, 1983.

[8] Battley H. *Single Fingerprints.* H.M. Stationary Office, London, 1930.

[9] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis.* Wiley, New York, 1973.

[10] E. Barnard and R.A Cole. A neural-net training program based on conjugate-gradient optimization. *Oregon Graduate Center Technical Report No. CSE 89-014*, July 1989.

[11] L.F.A. Wessels and E. Barnard. Local minima and neural nets. *Neural Networks*, Submitted, 1990.

[12] R.C. Gonzales and P. Wintz. *Digital Image Processing.* Addison-Wesley Publishing Company, Massachusetts, 1977.

[13] L. Coetzee and E.C. Botha. Preprocessing of two-dimensional fingerprint images for fingerprint recognition. *Proceedings of the IEEE Comsig91 conference*, pages 69–73, August 1991.

[14] D. Marr and E.C. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London B.*, 207:187–217, 1980.

[15] D.H.Ballard and C.M.Brown. *Computer vision.* Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[16] A.D. Brink. Grey-level thresholding of images using a correlation criterion. *Pattern Recognition Letters*, 9:335–341, 1989.

[17] L. Coetzee and E.C. Botha. Fingerprint recognition using the directional image. Presented at the Second South African workshop on Pattern Recognition,Stellenbosch University, November 1991.

[18] L. Coetzee and E.C. Botha. Fingerprint recognition with a neural-net classifier. *Proceedings of the First South African workshop on Pattern Recognition*, 1:33–40, November 1990.

[19] D.E.Glover. An optical fourier/electronic neurocomputer automated inspection system. In *International Conference on Neural Networks*, pages I–569 – I–576, San Diego, July 1988. IEEE.

[20] J.W. Goodman. *Introduction to Fourier Optics.* McGraw-Hill, New York, 1967.

Fingerprint recognition

by

Louis Coetzee

Submitted as partial fulfilment for the degree

Master of Engineering

in the Faculty Electronics and Computer Engineering

University of Pretoria

Pretoria

February 1992

Abstract

In this thesis methods of automatic fingerprint recognition are investigated. An introduction to fingerprint recognition as well as previous research done in this field are presented. This is followed by descriptions of the approaches we have followed while investigating the subject. These include a feature-based approach and a correlation-based approach. In the feature-based approach features are extracted from fingerprint images (e.g. the directional image) or transformations of the images (e.g. the Fourier transform) to use with various classifiers. These classifiers include the neural-net, linear and nearest-neighbour classifiers.

In the correlation-based approach fingerprint images are classified using a correlation classifier, without any feature extraction.

Simulation results obtained from the two approaches followed are presented. From these results the best method of fingerprint recognition is identified, before we conclude with a discussion of possible extensions to some of the approaches.