Massively Parallel Neural Network Fingerprint Classification System

C. L. Wilson
G. Candela
P. J. Grother
C. I. Watson
R. A. Wilkinson
National Institute of Standards and Technology
Advanced Systems Division
Image Recognition Group

Contents

List of Figures

List of Tables

Abstract

This report describes a massively parallel fingerprint classification system developed at NIST for the FBI that uses image-based ridge-valley features, K-L transforms, and neural networks to perform pattern level classification. The speed of classification is 0.54 seconds per fingerprint on a massively parallel computer. The system is capable of 88% classification accuracy with 10% rejects. As part of this development activity a sample of 4000 fingerprints, 2000 matched pairs, was collected and publicly released as NIST Special Database 4.

Keywords: fingerprint classification, parallel computing, neural networks, patten recognition, image processing.

Preface

This report describes an automated system for fingerprint classification. The report may be used at several different levels. A reader who wants to know the problem, the method of solution, and the results need read only chapters 1, 2, and 7. More detail about the functioning of the various system modules can be obtained by reading the introductions to each of the relevant chapters. More technical details of the system implementation can be obtained by reading the report in full.

Chapter 1

Introduction

1.1 Systems Description

Neural networks have had the potential for massively parallel implementation for some time, but system level image-based applications employing neural networks have only recently been realized, because the requirements for an image system include the image isolation, segmentation, and feature extraction as well as recognition.

The problem, Pattern level Classification Automation (PCA), is one of accurately classifying fingerprints into one of five classes from an input fingerprint image. The five classes are arch, left loop, right loop, tented arch, and whorl. These classes are abbreviated A, L, R, T, and W in this report. A detailed description of these classes is found in [1].

This report discusses machine learning based methods of solving the PCA problem using example images for training. The images used are 512 by 512 8-bit gray with a resolution of 20 pixels/mm. Examples of each of the classes are presented in figures 1.1-1.5. The locations of each of these example fingerprints in NIST Special Database 4 [2], discussed in the appendix, are given in table 1.1. The class TR indicates that the class of the fingerprint is a tented arch, which is cross referenced to a right loop.

Database File	Fingerprint Class
f0004_05	R
f0005_03	A
f0009_08	L
f0026_03	TR
f0046_07	W

Table 1.1: Locations of the example files in NIST Special Database 4.

A functional input-output data flow of the system discussed in this report is shown in table 1.2. The system uses ridge-valley directions, discussed in chapter 3, section 1, to convert the image

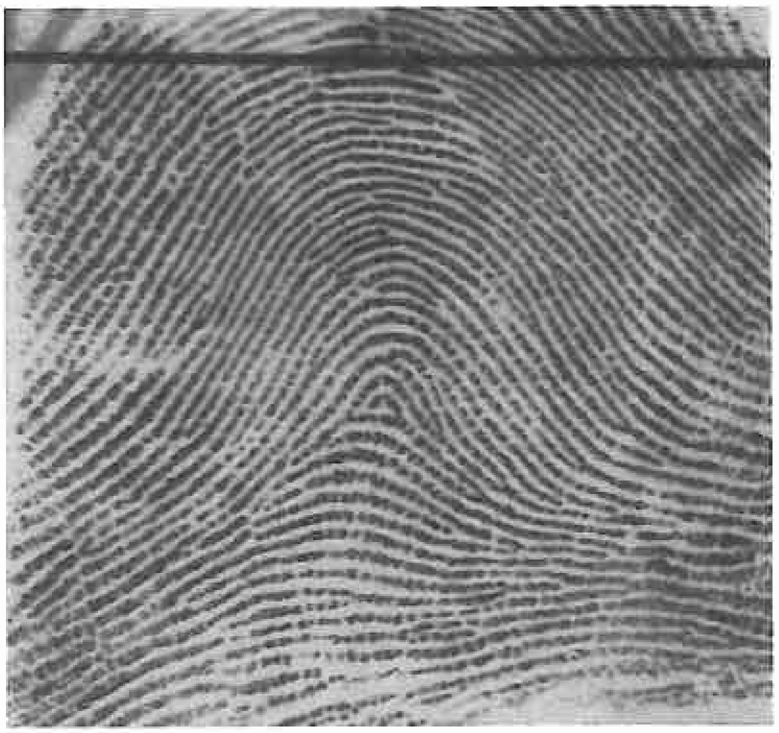

Figure 1.1: Examples of a arch fingerprint image

into a feature set which can be used either for alignment or for classification. The alignment of fingerprint cores using ridge-valley directions as an image alignment method is discussed in chapter 4. The Karhunen Loève, K-L, transforms of ridge-valley directions, which is used as a feature extraction method, is discussed in chapter 5. Multi-Layer Perceptrons, MLP's, are discussed in chapter 6 as a classification method. Finding two ridge-valley direction sets takes 0.4 seconds per image, the alignment 0.1 seconds per image, the K-L transform 20 ms per image, and the classification 1 ms per image. A classification accuracy of 88% is achieved with 10% rejects. The image processing prior to classification takes more than 99% of total processing time; the classification time is 0.03% of the total system's time.

Two neural-network-based methods are used in this system for feature extraction and classification. The K-L method is used [3] for feature extraction. This is a self-organizing method [4] in that it uses no class information to select features. The K-L method maximizes the variance in a feature set by using the principal eigenfunctions of the covariance matrix of the feature set. In the fingerprint system, local ridge directions are extracted from the image and used in subsequent processing. A similar technique has also been used with wavelets for face recognition [5] and for Kanji character recognition [6]. The K-L transform is dimension reducing; for ridge-valley features, the 1680 direction components are converted to (at most) 80 features.

The features generated by the K-L transform are used for training a MLP using Scaled Conjugate Gradient, SCG, optimization [7]. For the problems presented here, this method is from 10 to 100 times faster than backpropagation. Details of the method and procedures for obtaining

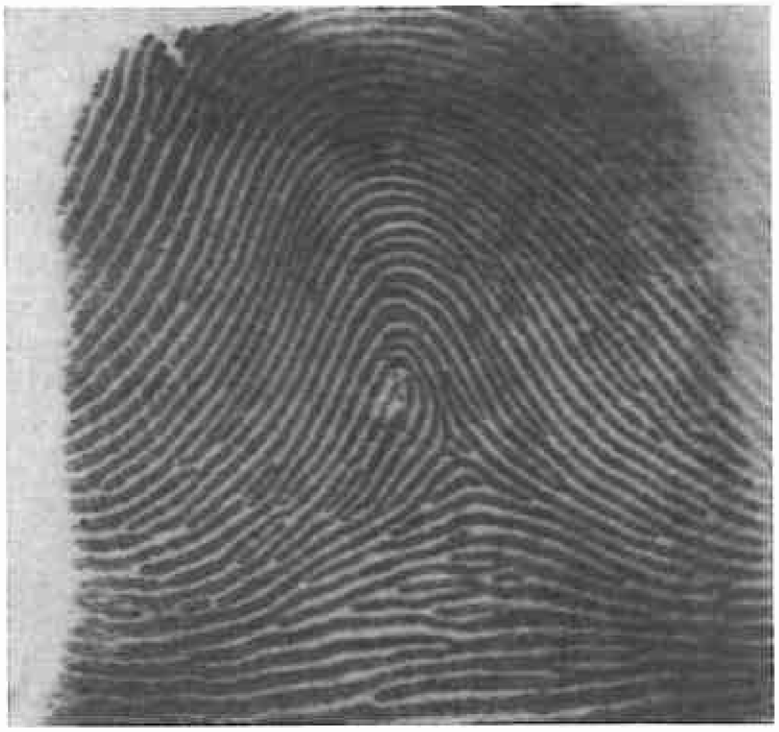

Figure 1.2: Examples of a left loop fingerprint image

the program are available in [7]. Typical classification accuracies on samples with equal numbers of each class are from 75% to 85%. When samples are used with natural distribution of classes accuracies of 94% have been achieved; the more common types of fingerprints are easier to recognize.

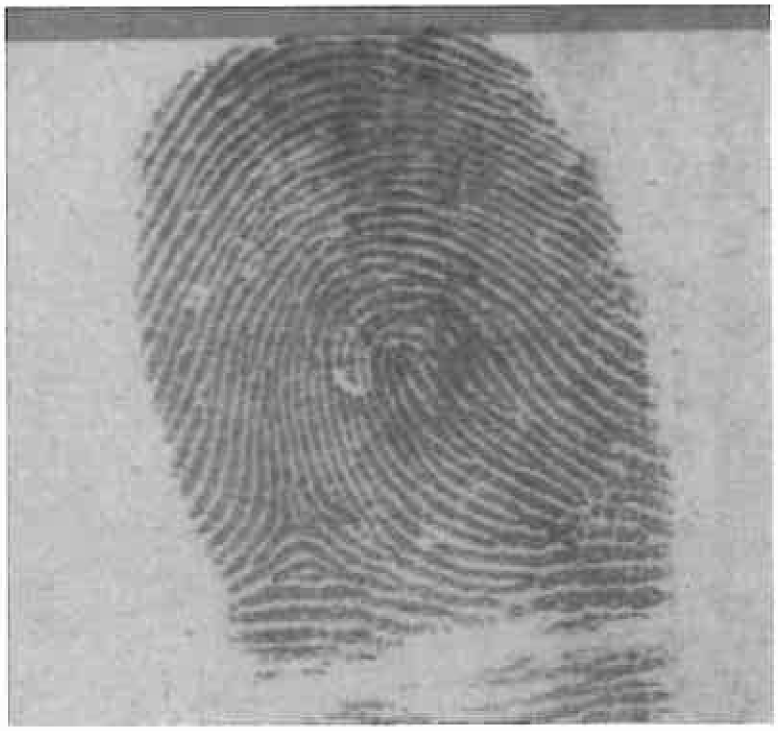

Figure 1.3: Examples of a right loop fingerprint image

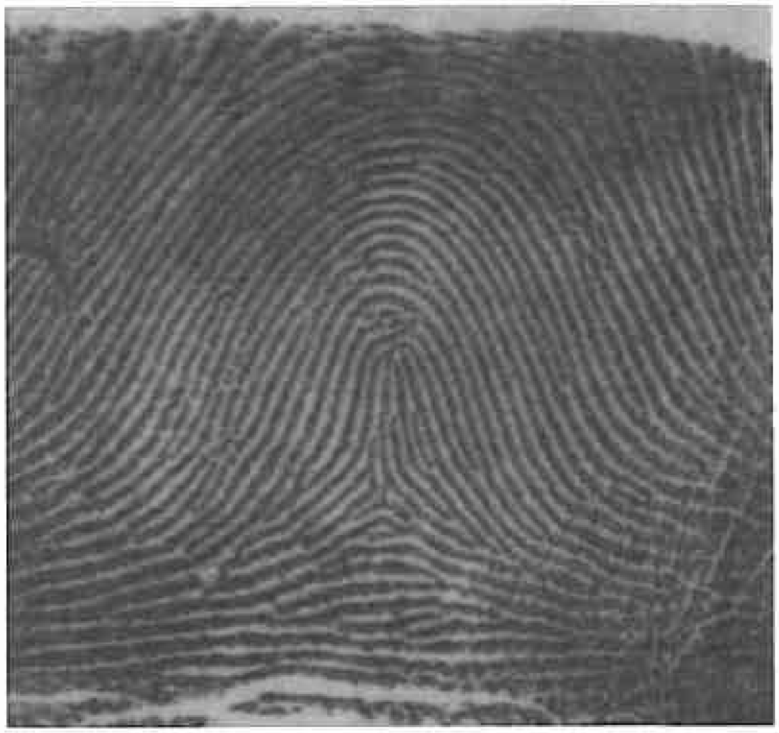

Figure 1.4: Examples of a tented arch fingerprint image which is cross referenced to a right loop

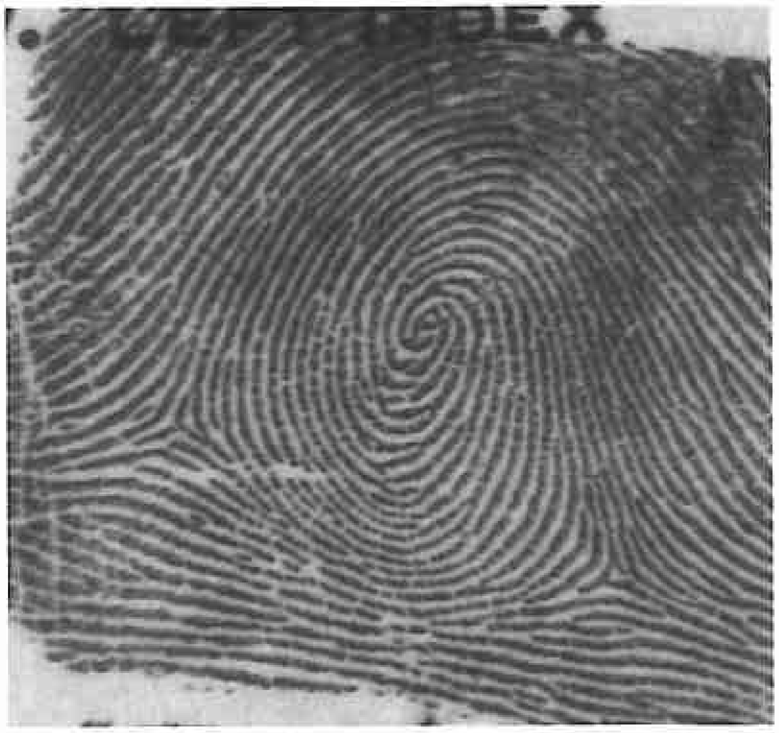

Figure 1.5: Examples of a whorl fingerprint image

System Module	Input	Output	Time (sec.)
Ridge Valley Features	8-bit 512 $\times$ 512 Image	840 directions (unregestered)	0.216
R92 Resistration	840 directions	offset Image	0.1
Registered Ridge Valley Features	x, y translated Image	840 directions (registered)	0.216
K-L Transform	840 directions	64 features	0.02
MLP Classifier	64 features	5 classes	0.001

Table 1.2: Input-Output Data flow for a fingerprint classification system based on supervised learning.

1.2 Parallel SIMD Hardware Architecture

The Single Instruction Multiple Data (SIMD) architecture used for this study was an Active Memory Technology 510c Distributed Array Processor with 8-bit math coprocessors.[1] This machine consists of a 32 by 32 grid of 1-bit processor elements (PE) and a 32 by 32 grid of 8-bit processors. Operation of the PE array is controlled by a four million-instruction-per-second RISC master control unit (MCU). All program instructions are stored in a separate program memory and are passed to the PE array through the MCU. A block diagram of this architecture is shown in figure 1.6.

Processor Elements Array and the Array Memory

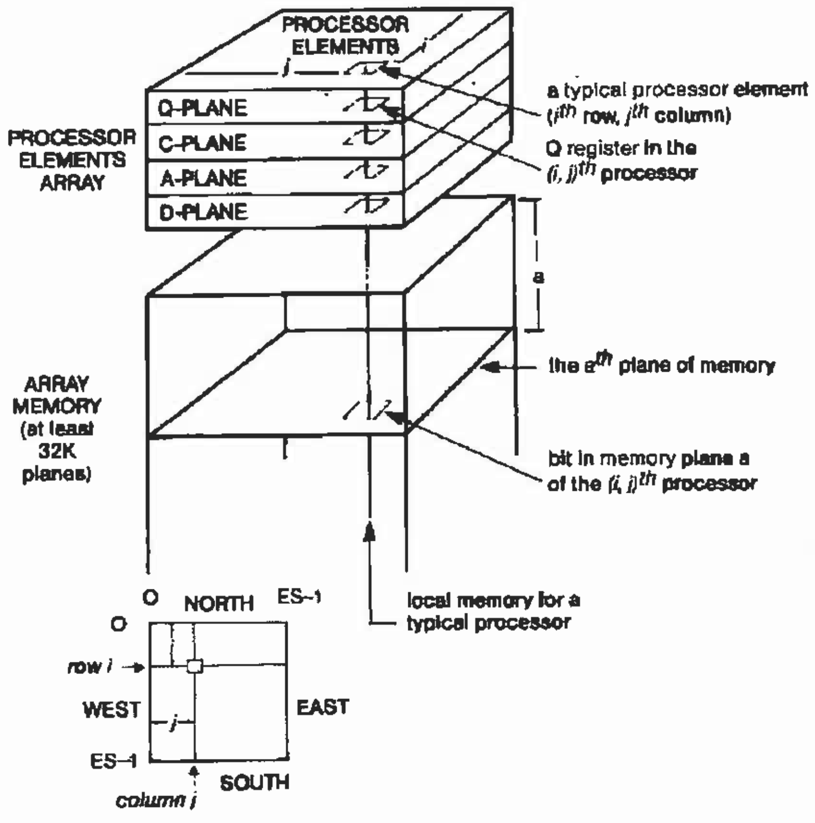
PROCESSOR ELEMENTS
i
j
Q-PLANE
C-PLANE
A-PLANE
D-PLANE
PROCESSOR ELEMENTS ARRAY
a typical processor element (i^{th} row, j^{th} column)
Q register in the $(i, j)^{th}$ processor
a
the a^{th} plane of memory
ARRAY MEMORY (at least 32K planes)
bit in memory plane a of the $(i, j)^{th}$ processor
local memory for a typical processor
O
NORTH
ES-1
O
row i
WEST
EAST
ES-1
SOUTH
column j

Figure 1.6: Array processor architecture for a massively parallel computer.

All data are stored in a separate array memory. The array memory is organized in 32 by 32 1-bit planes with corresponding bits in each plane connected to one PE. Data can also be passed between

[1]DAP510c or equivalent commercial equipment may be identified in order to adequately specify or describe the subject matter of this work. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment identified is necessarily the best available for the purpose.

PEs along the grid. The cycle time of all PEs is 100 ns. This processor configuration is capable of performing ten billion binary operations per second; processing time increases proportionally with the precision of data items used. Two data mappings are particularly well suited to the DAP structure: a vector mode in which successive bits of a single word are mapped into a row of the array. and a matrix mode in which successive bits of a word are mapped into layers of the array memory vertically. Operations in both of these modes of operation are used in the system implementation presented in this paper.

The use of a massively parallel computer allows the application of techniques that would not be implemented on a serial computer due to their computational expense. Image processing and analysis can be done with fewer system cycles since many pixels of the image can be manipulated or analyzed at once. In the case of the DAP510c. up to 1024 pixels can be manipulated at the same time.

1.3 Outline of Report

Chaper 2 of this report discusses the PCA problem from the general point of view of self-organizing and supervised pattern recognition. Chapter 3 discusses three image-processing methods which have been used for image feature extraction and image reconstruction. The three methods are ridge-valley filtering, Fourier transform filtering, and Gabor filtering. Ridge-valley filtering is used in the rest of the report because of greater computational efficiency on the parallel computer used for this work. Chapter 4 discusses the alignment method used based on the R92 [8] algorithm. Chapter 5 discusses the K-L transform method of feature extraction. Chapter 6 discusses the MLP classification from K-L features. Chapter 7 provides a summary and some initial conclusions on the solution of the PCA problem. The appendix discusses the test data used and the construction of the test database.

Chapter 2

Basic Models for Classification

2.1 Introduction

In the past few years neural networks have been discussed as a possible method for constructing computer programs that can solve problems, like speech recognition and character recognition, where "human-like" response or artificial intelligence is needed. The most novel characteristics of neural networks are their ability to learn from examples, their ability to operate in parallel, and their ability to perform well using data that are noisy or incomplete. In this chapter, these characteristics of neural networks are illustrated using real examples from the field of fingerprint classification, not "toy" problems. The goal of this chapter is to define the PCA problem in a way that will allow neural networks to make a significant advance in solving this problem and to show one way computers can be programmed to learn to solve pattern classification problems.

2.1.1 Other Work

The neural approach to machine learning was originally devised by Rosenblat [9] by connecting together a layer of artificial neurons [10] on a perceptron network. The weaknesses which were present in this approach were analyzed by Minski and Papert [11], and work on neurally based methods was abandoned by all but a small group of researchers for the next ten years. The advent of new methods for network construction and training in this ten year period led to rapid expansions in neural net research in the late 1980s.

Two types of learning, supervised learning and self-organization, are common in neural networks. The material presented in this paper does not cover the mathematical detail of these methods. A good source of general information on neural networks is Lippmann's review [12]. The primary research sources for neural networks are available in Anderson and Rosenfeld [13]. More detailed information on the supervised learning methods discussed here is given in [14]; self-organizing methods are discussed by Kohonen [15] and Grossberg [16].

Neural Network Approach

The principal difference between neural network methods and rule-based methods is that the former attempts to simulate intelligent behavior by using machine learning and the latter uses logical symbol manipulation. Once the learning phase has been completed the network response is automatic and similar in character to reflex responses in living organisms. The processes where these methods have been most successful are in areas where human responses are automatic, such as touching ones nose or recognizing characters. Neural networks have been successful in engineering applications such as character recognition, speech recognition, and control systems for manufacturing, where information is incomplete and context dependent. In other areas, such as learning arithmetic [17], where automatic responses by humans are not expected, and precise answers are required, neural networks have had substantially less success.

For example, to teach a neural network to recognize and classify fingerprints that are right and left loops, a neural network is shown features derived from examples of these fingerprint classes.. The program learns the critical factors for separation of right and left loops from the feature set and sorts future images of these fingerprints into the appropriate classes. As the class set expands, more examples, a more complex feature set, and a more complex classification network are needed.

Artificial Intelligence Rule Bases Approach

The alternate approach to artificial intelligence is rule-based. An introductory text on this approach is Winston's book on artificial intelligence programming using Lisp [18]. Rather than teaching the program the difference between right and left loops, a rule-based program is constructed to distinguish between the two digits by writing rules explicitly into the program. For the L-R loop problem the rules might be as simple as "it's a loop if only a single core and delta are present; it's a left loop if the core is to the left of the delta and a right loop otherwise." This rule gives ambiguous results if the core is above the delta since the correct class is unknown to the system. As the class set expands, more complex rules are needed.

2.1.2 Pattern Classification and Feature Extraction

The implementation of PCA discussed in this report combines several neural network methods to carry out the filtering and feature extraction. The filtering methods used are based on the ridge-valley method described in chapter 3 and on a method based on K-L transforms [3] described in chapter 5. In the supervised method, the recognition is done using features extracted from the K-L transform.

The basic classification process starts with an image of an fingerprint. In the self-organizing method, the ridge directions of the image are applied directly to the neural network and any filtering is learned as features are extracted. In the supervised method, the features are extracted using the K-L transform, which is self-organizing, but classification is carried out using an MLP network.

2.2 Self-Organized vs. Supervised Learning

Artificial neural network systems are constructed as interacting subsystems that have parallel data flow between the layers and parallel processing of data in each subsystem. This makes them ideal for implementation on parallel computers. Ideally, the subsystems are composed of layers of processors that carry out the processing functions within each layer using many parallel input and output paths. In figure 2.1 for example, all pixels of the image are simultaneously applied to the left of the figure so that all parts of the input are filtered in parallel.

Figure 2.1 illustrates the processing needed to construct a simple classification system. In PCA the input is an image containing a single fingerprint. If the input is filtered, an image of the fingerprint with ridges enhanced is produced. The input to the system is initially converted to a more compact representation in terms of ridge direction data; this conversion is called ridge-valley feature extraction. After the ridge-valley feature extraction is performed, then a set of numbers which represents the input data in a more compact form, ridge direction data, is produced. In the next calculation the K-L transform is used to filter the ridge-valley data by expanding it in terms of a set of characteristic image components, the eigenfunctions of the image covariance. This representation of the data is then used for classifying the input in each of the learned classes, providing an estimate of the probability of the input being in each of the known classes. In the final calculation, the input is assigned to one or more of the known classes.

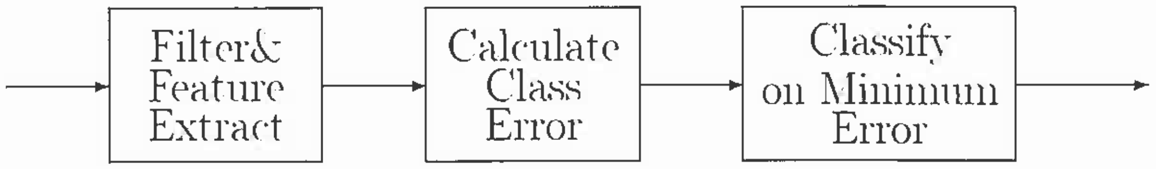

Figure 2.1: Data flow in a simple network designed for pattern classification.

The process described above is all that is necessary for classification but needs a modification to allow learning. This modification is shown in figure 2.2. The filter and feature extraction process remain unchanged, as does the idea of calculating class errors, but a switch is introduced into this calculation which decides if the error is low enough to allow classification. This corrective path is used to provide the network with a learning capability. The differences between a self-organizing system and a supervised system are in the processes which deal with changing the network when the error correction switch is triggered.

2.2.1 Self-Organized Learning is a Clustering Method

Figure 2.3 illustrates a method used to perform self-organized learning. The data input path contains any required filters or feature extraction calculations. The switch used to activate learning compares the pattern to classes which have been learned and makes a decision on the basis of pattern similarity; it then either modifies known patterns using the new pattern or creates a new pattern class to accommodate the pattern. In the initial phases of the process of self-organization, the weights of the neural network, which act as a memory, are loaded with data that matches all classes equally, usually random data. As learning proceeds, the weights for strong information associated with exemplars of the class types present in the learning data are reinforces. Self-organizing methods based on decision trees have existed for some time. The unique feature of the neural network methods is the parallel nature of the algorithms.

2.2.2 Supervised Learning is a Bounding Method

Figure 2.4 illustrates the method used for supervised learning. As in figure 2.3, data filtering and feature extraction are done at the input. The learning switch operates on a different principle. In the supervised system, the learning switch is driven by the change in the error of the global system which requires a set of data for which the correct answers are known. The correct responses are used to calculate an error; a search over this error surface is supervised by the error detection and switching process. This process is an error minimization process, and in this work the conjugate gradient method is used for error minimization.

2.2.3 Example: A Two-Feature Problem

An example of a two-class, two-feature problem using self-organization is shown in figure 2.5. The problem is to distinguish the arch class "A" from the whorl class "W." Two hundred samples of each fingerprint class are used in this example. A three-class, two-feature problem is shown in figure 2.6. In this example, the tented arch class "T" is added to the set of classes, and two hundred examples of it are added to the previous sample. Feature extraction in both examples is performed by correlating the input fingerprint image with the first two principal components, or eigenfunctions, using the K-L transform. One principal component is plotted on each axis of these figures. The self-organization method used calculates the distance from the mean of each class. If the distributions of the classes are assumed to be normal, ellipses with major and minor axis lengths 2.17 times the standard deviation are expected to enclose 99% of each class. This method is a variation of the Probabilistic Neural Network [19]. Ellipses of these sizes are used in figures 2.5 and 2.6.

This is an extremely simple clustering method, but it illustrates the essential concepts of many self-organizing neural systems. In the A-W problem, the two classes are fairly well separated; only a small number of characters fall outside the 99% cluster boundaries. In the A-T-W problem, the "T"s overlap both the "A"s and part of the"W"s, but the overlap with the the "A"s is total. This is an illustration of the observation that "T"s look more like "A"s than "W"s. Both problems illustrate an important property of clustering methods: the ability to identify cases where the decision uncertainty is very high. Two types of uncertainty are shown. Uncertainty based on insufficient information is demonstrated by points outside any of the ellipses, the "W"s at the bottom of the 99% boundary. Uncertainty caused by confusion is demonstrated by points in two ellipses at once. In most self-organizing systems a point in this overlap region will be assigned to classes based on some correlation measure between the point and examples which have been

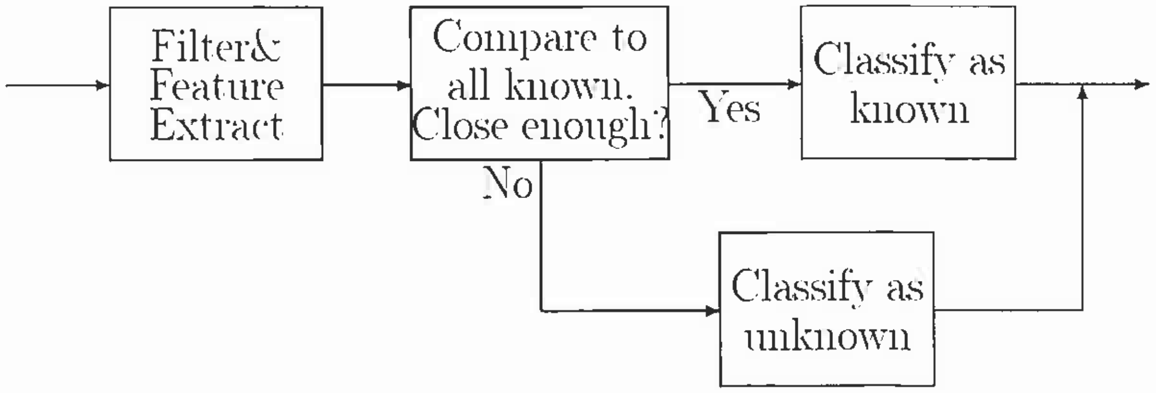

Figure 2.2: Data flow in a simple classification system.

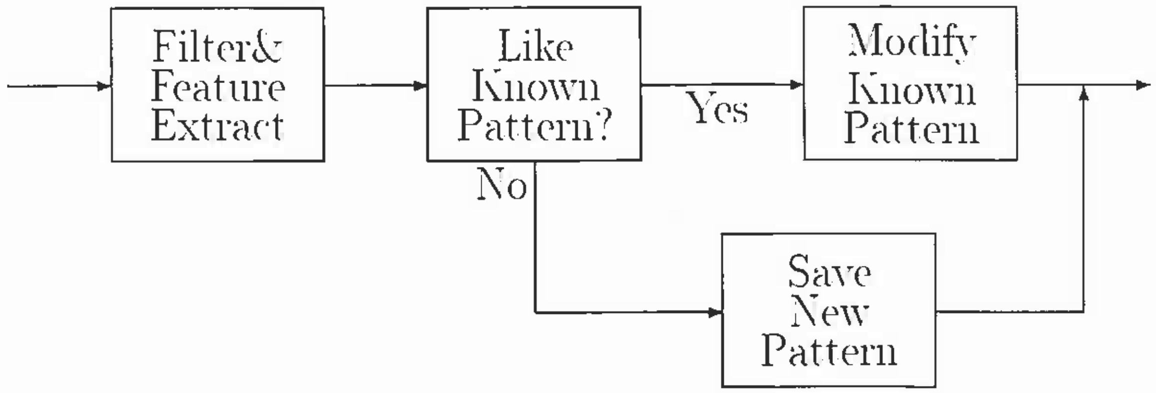

Figure 2.3: Data flow in a simple self-organizing system.

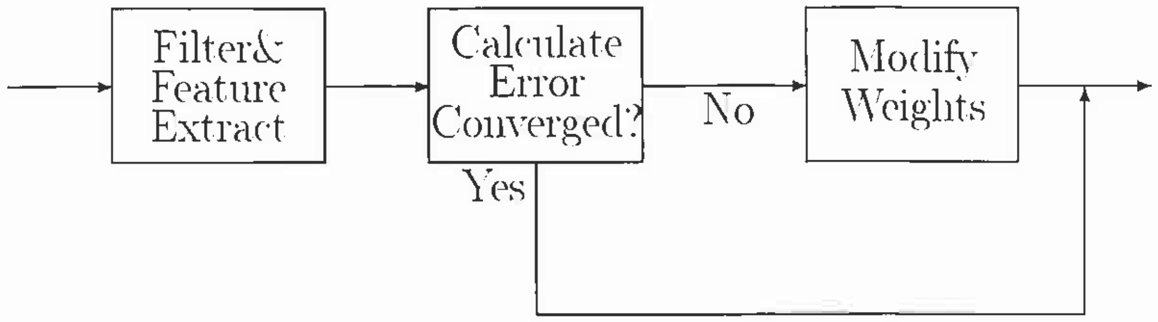

Figure 2.4: Data flow in a simple supervised-learning algorithm.

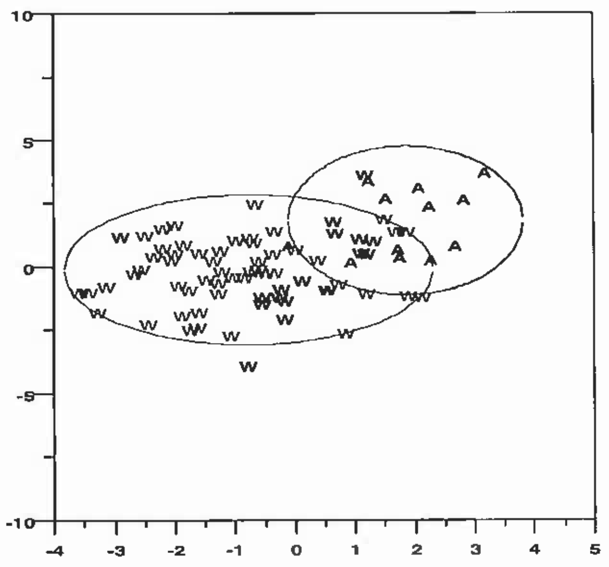

Figure 2.5: Self-organized clusters for the A-W classification problem.

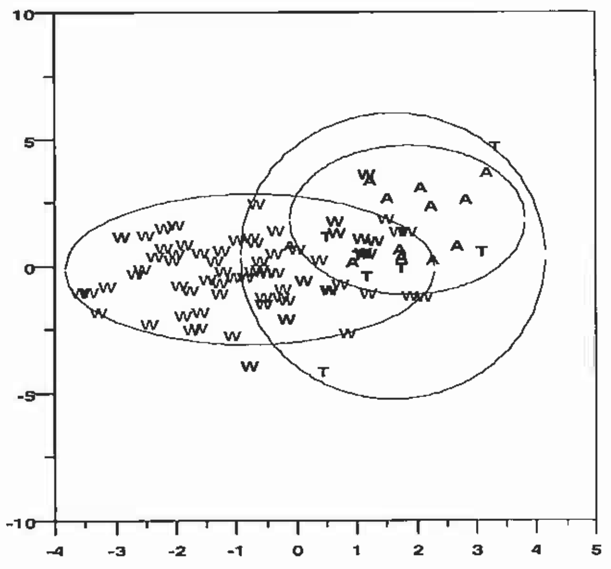

Figure 2.6: Self-organized clusters for the A-T-W classification problem.

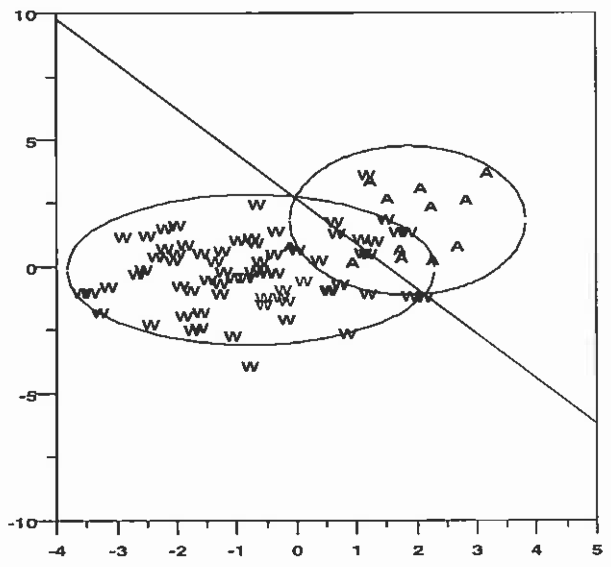

Figure 2.7: Supervised linear model for the A-W classification problem superimposed on the previous self-organizing clusters.

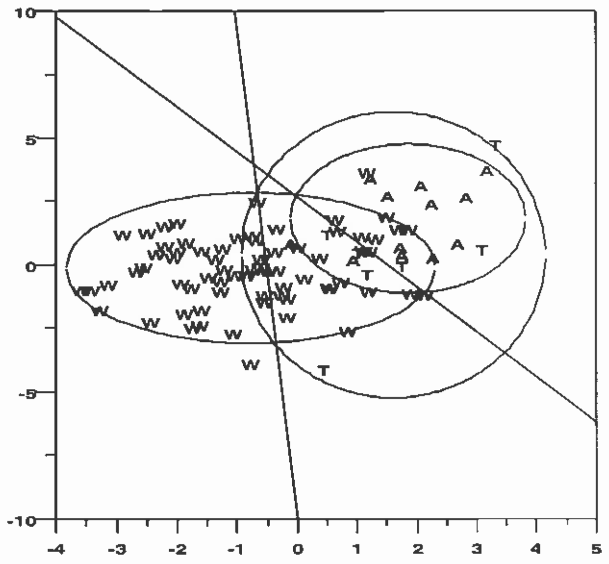

Figure 2.8: Supervised linear model for the A-T-W classification problem superimposed on the previous self-organizing clusters.

Chapter 3

Image-Based Feature Extraction and Filtering

In this chapter three methods of feature extraction and image filtering are discussed. These three methods are the ridge-valley filter, a Fourier-transform-based adaptive bandpass filter, and Gabor filtering. The method used throughout the rest of this report is the ridge-valley filter. This choice was based on the filter cost for an acceptable level of feature and image quality. The ridge-valley filter, implemented on a parallel computer, can process a 16 by 16 image tile in 256 μs. The Fourier transform filter, programed in assembly code on the same computer, can process a 32 by 32 image in 1 ms. The 32-term Gabor filter, using prefactored coefficients, can process a 32 by 32 image in 9 ms. The quality of image reconstruction available with these three methods is shown in figures 3.1 through 3.3. The original gray image is shown in figure 1.1.

The figures demonstrate that all of the methods remove some data and introduce some artifacts. The ridge-valley filtered image shown in figure 3.1 produces the image with the fewest artifacts. These artifacts are primarily white spaces in lines that change shape as a result of processing. The Fourier-transform-based filtered image shown in figure 3.2 introduces more artifacts. Most of these are associated with 32 by 32 tile edges. Note that in the upper right, the ridges are joined across the horizontal line at the top of the fingerprint. The Gabor transform filtered image shown in figure 3.3 has serious artifact problems caused by using only 32 Gabor functions. From these images we conclude that the best speed-quality combination is achieved using the ridge-valley filter. All three filters are discussed in detail in the following sections.

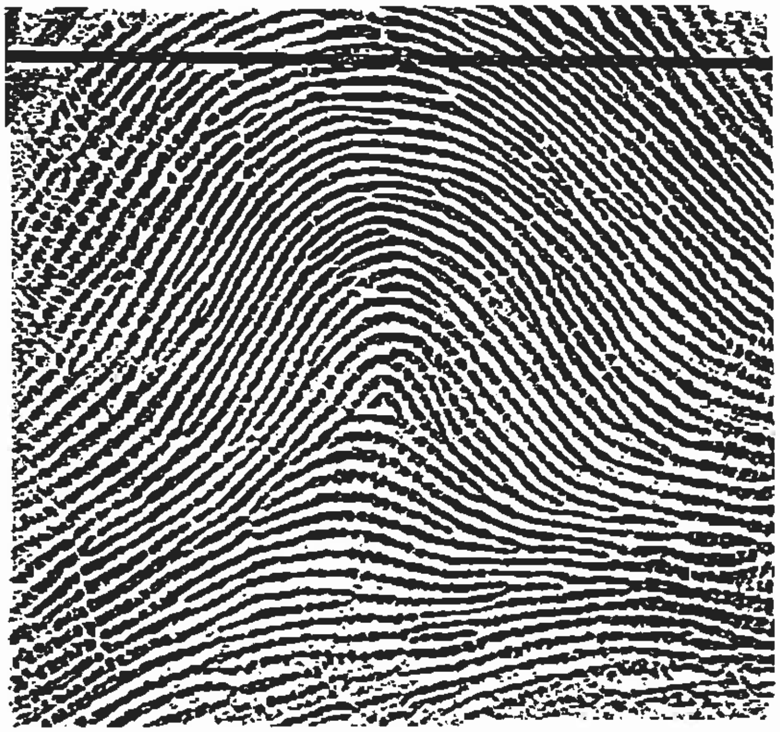

Figure 3.1: Example of ridge-valley filtering of a fingerprint image.

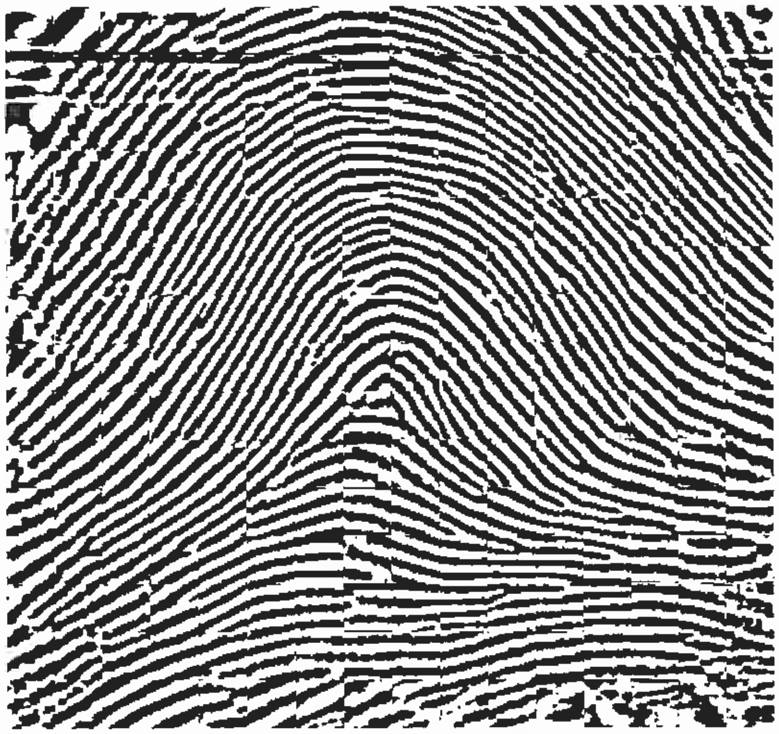

Figure 3.2: Example of Fourier transform based filtering of a fingerprint image.

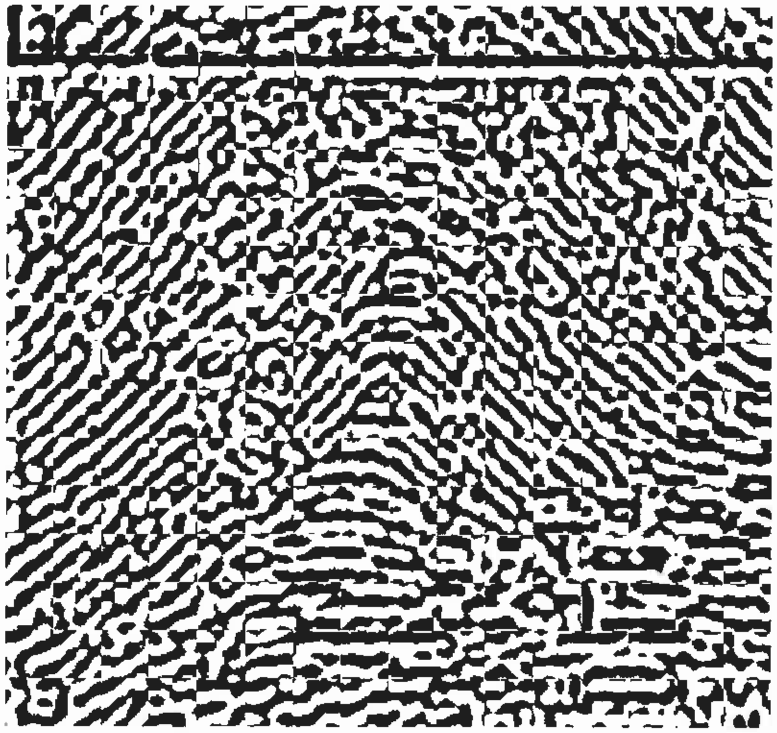

Figure 3.3: Example of Gabor filtering of a fingerprint image.

3.1 Ridge-Valley Features

This program is based on a ridge-valley fingerprint binarizer described in reference [20]. The binarizer works as follows. For each pixel (C) of the image, slit sums $s_i, i = 1 \dots 8$, are produced, where each s_i is the sum of the values of all the pixels labled i in figure 3.4. The binarizer is intended to reduce a gray-level fingerprint image to an acceptable black and white image. It is a compromise between "local thresholding" and "slit comparison" formulas. The local thresholding formula sets the output pixel to white if the input pixel sum, S, exceeds the average of the pixels in the slits, that is, if

$$4S > \frac{1}{8}\sum_{i=1}^{8} s_i. \tag{3.1}$$

The slit comparison formula sets the output pixel to white if the average of the maximum and minimum slit sums is greater than the average of all the slit sums:

$$s_{max} + s_{min} > \frac{1}{4}\sum_{i=1}^{8} s_i. \tag{3.2}$$

The motivation 3.2 is as follows. If a pixel is in a light "valley" area of the print, then one of its eight slits will lie along the valley and have a high sum, whereas the other seven slits will cross several ridges and valleys each and have roughly equal, lower sums; a similar reasoning applies to a pixel in a dark "ridge" area. Stock and Swonger [20] used an equally-weighted compromise between the above two formulas, so that the output pixel is set to white if

$$4S + s_{max} + s_{min} > \frac{3}{8}\sum_{i=1}^{8} s_i. \tag{3.3}$$

The ridge-valley direction finder (which is apparently very similar to the one currently implemented for the FBI on special parallel hardware) is an extension of the ridge-valley binarizer. Upon binarization, each "ridge" (black) pixel is considered to have the direction of its minimum-sum slit, and each "valley" (white) pixel, the direction of its maximum-sum slit. Each pixel, then, has a direction, coarsely quantized to 8 bins. To produce a much smaller grid of directions, spaced every 16 pixels, the pixel directions are averaged over 16 by 16-pixel squares. Averaging has a smoothing effect and produces a finer quantization of directions.

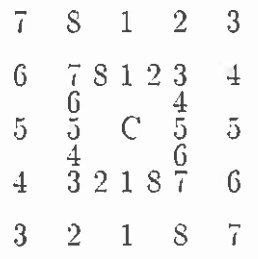

Figure 3.4: Ridge-valley filter pixel arrangement.

The ridge angle θ at a location is here defined to be 0° if the ridges are horizontal, increasing towards 180° as the ridges rotate counterclockwise, and snapping back to 0° when the ridges become horizontal again: $0° \le \theta < 180°$. When pixel directions are averaged, the quantities averaged are actually not the pixel ridge angles θ, but rather the pixel "direction vectors" $(\cos 2\theta, \sin 2\theta)$. If the angles themselves are averaged, results can be absurd: for example, the average of a 1° direction and a 179° direction, each nearly horizontal, produces a vertical 90° direction. The problem is that as a rotating ridge gets nearly around to being horizontal again, the angle does not continuously approach the starting angle of 0°, but instead becomes large and then jumps to 0°. Moving the angle origin or multiplying angles by a factor cannot eliminate this discontinuity problem. Averaging the cosines and sines of the angles also fails: for the same 1° and 179° directions, the (cosine, sine) vectors are (0.999848, 0.017452) and (-0.999848, 0.017452), whose average is the vertical (0, 0.017452). This fails because 1° ridges and 179° ridges, although almost opposite as vectors, are almost parallel as ridges. This scheme arbitrarily assigns to ridges a nonexistent flow direction, forcing the 1° and 179° ridges to flow in almost opposite directions. Fortunately, both doubling the angles and taking cosines and sines solve the problem: for example, 1° and 179° ridges become (0.999391, 0.034900) and (0.999391, -0.034900), which average to the horizontal (0.999391, 0).

The direction finder's output is a grid of these averages of pixel direction vectors. Since the pixel direction vectors have length 1, each average vector has a length of at most 1. If a region's ridge direction is poorly defined, for example because of blurring, the directions of its several pixels will vary and cancel each other out, producing a short average vector. The length of an average vector is therefore a direction confidence measure. Examples of the ridge directions generated by this method are shown in figures 3.5-3.9 for the five fingerprints shown in chapter 1.

As implemented on the DAP parallel SIMD computer, this direction finder takes about 0.22 s to assign 840 (28 by 30) ridge directions to a 480 by 512 pixel fingerprint.

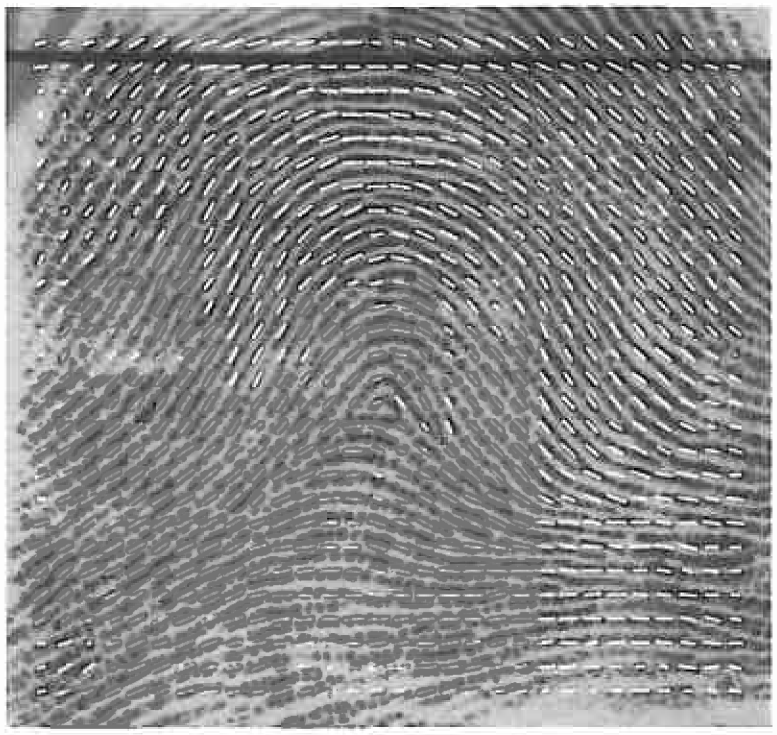

Figure 3.5: Examples of a arch fingerprint image with ridge direction from the ridge-valley algorithm

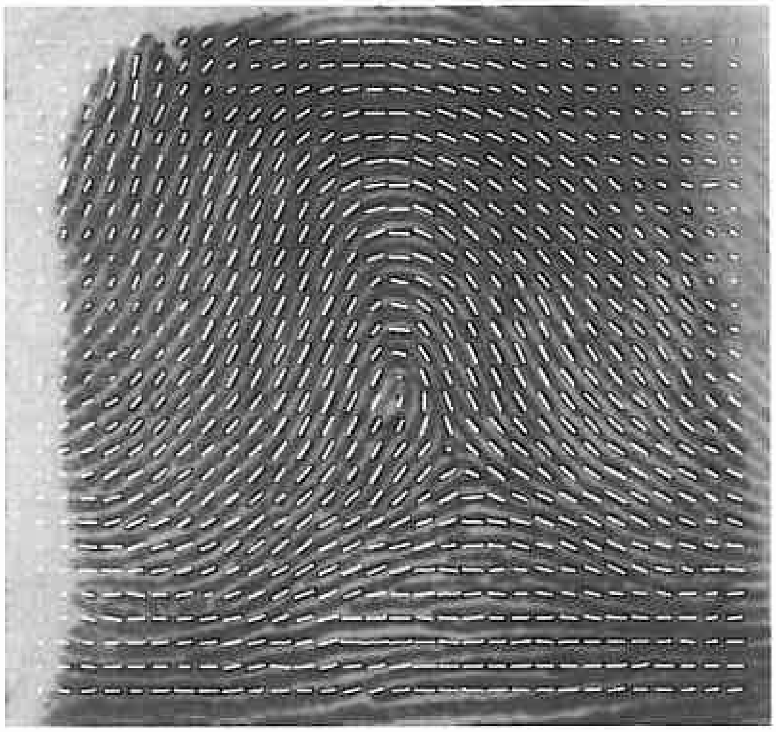

Figure 3.6: Examples of a left loop fingerprint image with ridge direction from the ridge-valley algorithm

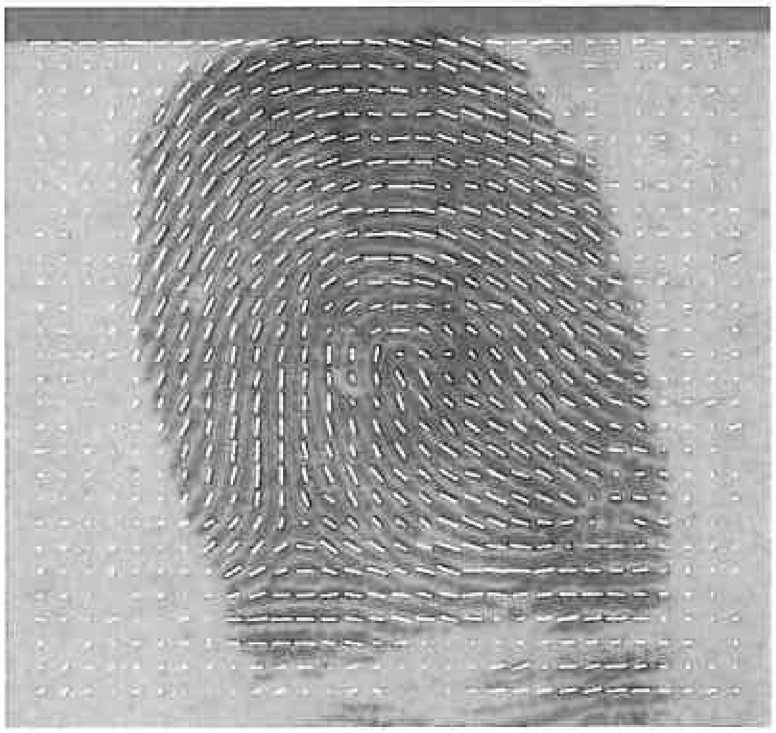

Figure 3.7: Examples of a right loop fingerprint imagewith ridge direction from the ridge-valley algorithm

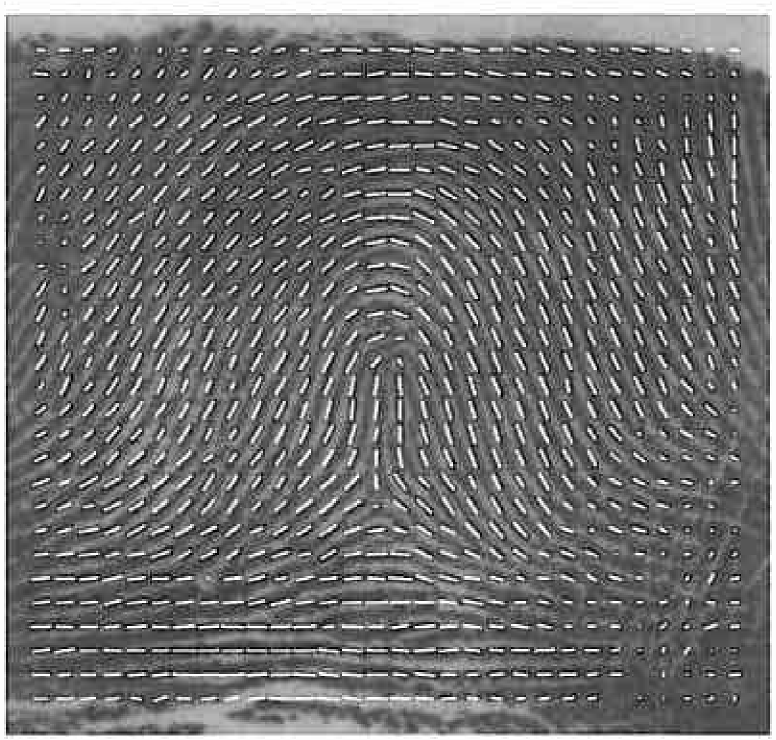

Figure 3.8: Examples of a tented arch fingerprint image which is cross referenced to a right loopwith ridge direction from the ridge-valley algorithm

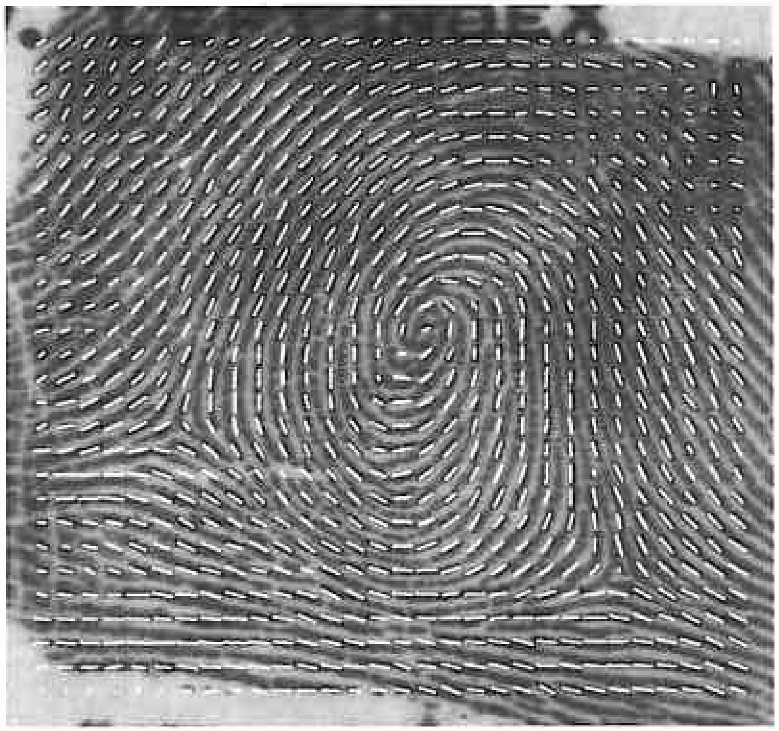

Figure 3.9: Examples of a whorl fingerprint imagewith ridge direction from the ridge-valley algorithm

3.2 Fourier Transform Features

Figure 3.10: Examples image of a space walk.

This method is based on the observation that in a conventional image such as figure 3.10 the power spectrum peaked at the lowest frequencies and falls off as $1/f$ as shown in figure 3.11. In a fingerprint, this is not true. The local power spectrum has two distinct peaks on either side of the center DC value which represent the ridge frequencies of that region of the fingerprint as shown in figure 3.12.

In this FFT-based method, each 32 by 32 pixel region has its two-dimensional fast Fourier transform taken, and the power spectrum of the result is used to derive a ridge direction. It was observed that a regular ridge pattern produces a Fourier power spectrum consisting of two bright spots symmetrically arranged about the center and that the spots rotated about the center as the ridge pattern was rotated. Each pixel of the power spectrum, then, corresponds to a particular ridge direction. One can assign an average direction for the region by using the power spectrum elements, normalized by dividing by their sum, as weights on the "basic" direction vectors corresponding to the power spectrum element positions. This is similar to the weighted averaging of the eight directions used in the ridge-valley method above.

Two things were done to improve the speed of this method. First, since the FFT of pure-real input has a certain symmetry (the $(-x, -y)$ element is the conjugate of the (x, y) element), it is possible to take the FFTs of two regions with one call of the FFT routine, by using one region as the real input and the other as the imaginary input and then doing a little algebra after the FFT

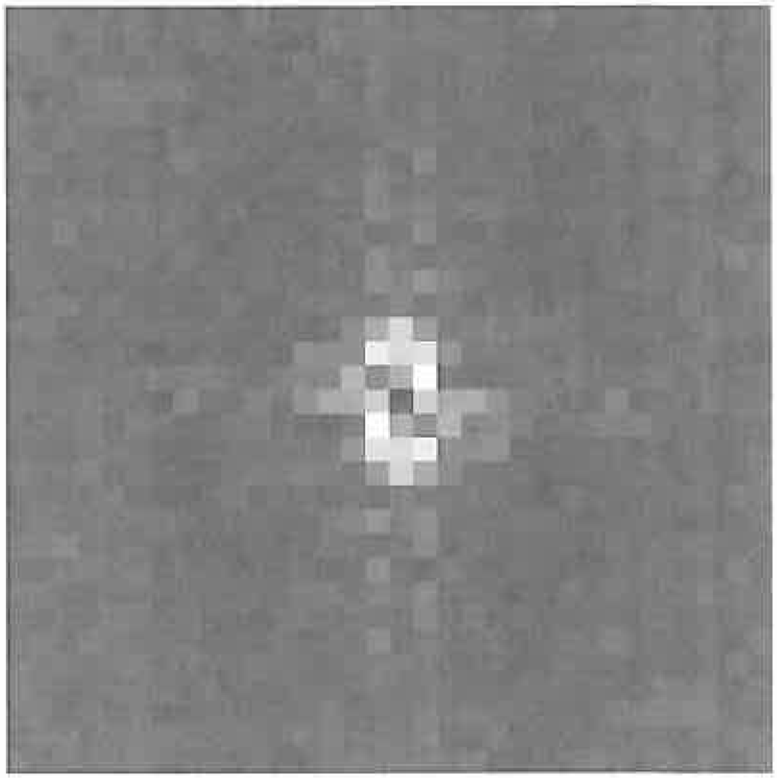

Figure 3.11: Examples of Fourier transform power spectrum of a 32×32 part of the space walk image. DC is at the center of the image and high frequencies are at the corners.

to separate out the desired outputs. Secondly, since the fingerprint regions are almost always very weak in high frequencies, it is possible to apply the checkerboard pattern to one region and add it to another (non-checked) region and to use this as one input to the FFT; the outputs of the two regions can later be separated, even though they are added together, because one (the checked one) has its low frequency, and hence far-from-zero, elements near the center of the result, and the other has its low frequencies near the corners. Combining these two shortcuts, the program was made to run considerably faster with no noticeable diminution in quality.

As implemented on the DAP, this method took about 1.575 s to assign 256 ridge directions to a 512 by 512 pixel print, compared to 0.216 second for the ridge-valley method. It also produced noticeably inferior results compared to ridge-valley as shown in figure 3.2.

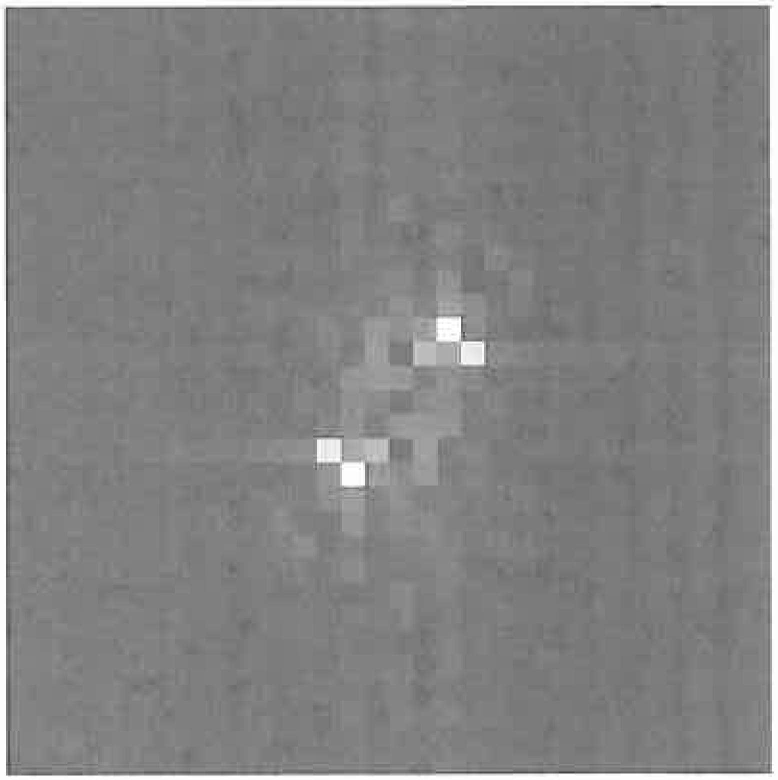

Figure 3.12: Examples of Fourier transform power spectrum of a 32×32 part of a fingerprint image. DC is at the center of the image and high frequencies are at the corners.

3.3 Gabor Features

3.3.1 Gabor Functions

The Gabor filtering section is accomplished using a least-squares fit of each 32 by 32 image tile. The kernel functions used are Gabor functions. The least-squares fitting of the filter coefficients is necessitated by the nonorthogonal nature of the Gabor functions. Adopting the convention that bold upper case variables represent array processor matrix data types and bold lower case variables represent array processor vector data types, the Gabor functions are defined as:

$$\mathbf{G}_j(\mathbf{X}, \mathbf{Y}) = \exp(-\mathbf{R}^2) \left\{ \begin{array}{l} \sin(\omega_j \mathbf{X}') \\ \cos(\omega_j \mathbf{X}') \end{array} \right. \tag{3.4}$$

where the matrix variables $\mathbf{R}$, $\mathbf{X}'$, and $\mathbf{Y}'$ are given by:

$$\mathbf{R}^2 = (\mathbf{X}'^2 + \mathbf{Y}'^2)/\sigma_j^2 \tag{3.5}$$

$$\begin{pmatrix} \mathbf{X}' \\ \mathbf{Y}' \end{pmatrix} = T \begin{pmatrix} \mathbf{X} - x_{0j} \\ \mathbf{Y} - y_{0j} \end{pmatrix}. \tag{3.6}$$

The $\mathbf{X}$ and $\mathbf{Y}$ matrices in the array processor are row and column expanded in the form:

$$\mathbf{X} = \begin{pmatrix} x_1 & x_2 & \ldots & x_{32} \\ x_1 & x_2 & \ldots & x_{32} \\ & & \ldots & \\ x_1 & x_2 & \ldots & x_{32} \end{pmatrix} \tag{3.7}$$

$$\mathbf{Y} = \begin{pmatrix} y_1 & y_1 & \ldots & y_1 \\ y_2 & y_2 & \ldots & y_2 \\ & & \ldots & \\ y_{32} & y_{32} & \ldots & y_{32} \end{pmatrix} \tag{3.8}$$

A typical scalar transformation to be applied to each element of the matrix variables is a rotation of the form:

$$T = \begin{pmatrix} \cos\theta_j & \sin\theta_j \\ -\sin\theta_j & \cos\theta_j \end{pmatrix}. \tag{3.9}$$

The matrix function $\mathbf{G}_j$ is then expressed as a function of the scalar variables: ω_j, which is the spatial frequency of the function; σ_j, the spatial extent of the function; (x_{0j}, y_{0j}), the origin of the function; and θ_j, the orientation of the function.

3.3.2 Tiling

Since the Gabor basis functions are an infinite set, it is necessary to select a specific subset of them to be used as the filter elements which cover the image tiles. This selection process is referred to as tiling the image. For the class of filter discussed here each set of image origins has twice the sample density of the previous level and the number of directions selected, n_θ, is fixed. This results in a filter with directional sensitivity and positional sensitivity determined by the choice of the level parameter, i. The subimages of the fingerprint images used in this study are 32×32 so that using large values of i would result in massive over-sampling of the image. The Gabor filter for the lowest value of i, on the other hand, is approximately a directional bar detector and adds little to the filter's spatial resolution. At each level the frequency and spatial resolutions, ω_i and σ_i, are adjusted to allow small overlaps in extent and provide octave spatial frequency response.

After extensive experimentation, it was found that a reasonably good approximation to the fingerprint subimages could be obtained by using only level 2 Gabor functions. Reasonable directional selectivity was obtained with four fold symmetry, $n_\theta = 4$.

The results of the experiments are easily explained. For $i = 1$, the Gabor functions form a directional filter and provide only field-centered spatial location and is too low frequency to sample the ridge data. As i increases the resolution of the filter increases. At level 3 the spatial and frequency resolution exceed the ridge pitch of the fingerprint and provide limited improvement in resolution. All experiments also suggest that, given the complex structure of equations 3.4-3.9, sampling an image containing less than 16 pixels for each d_i interval is not an efficient use of Gabor functions.

i	x_{0i}	y_{0i}	ω_i	σ_i	θ_i
1	$\frac{d_0}{2}$	$\frac{d_0}{2}$	$\frac{2\pi}{d_0}$	d_0	$\frac{2\pi}{n_\theta}, \frac{4\pi}{n_\theta}, \ldots, 2\pi$
2	$\frac{d_1}{2}, \frac{3d_1}{2}$	$\frac{d_1}{2}, \frac{3d_1}{2}$	$\frac{2\pi}{d_1}$	d_1	$\frac{2\pi}{n_\theta}, \frac{4\pi}{n_\theta}, \ldots, 2\pi$
...					
n	$\frac{d_n}{2}, \frac{3d_n}{2}, \ldots, \frac{(2^n-1)d_n}{2}$	$\frac{d_n}{2}, \frac{3d_n}{2}, \ldots, \frac{(2^n-1)d_n}{2}$	$\frac{2\pi}{d_n}$	d_n	$\frac{2\pi}{n_\theta}, \frac{4\pi}{n_\theta}, \ldots, 2\pi$

Table 3.1: Table of the possible Gabor functions used to tile the image. Each level, i, contains $2i^2 n_\theta$ possible Gabor functions.

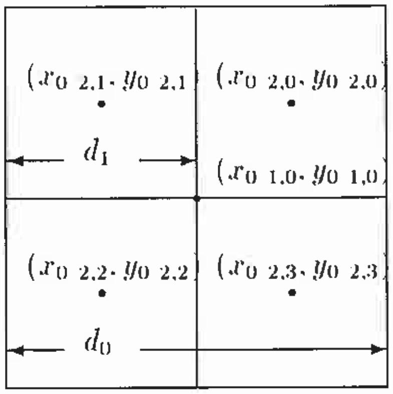
$(x_{0\ 2,1}, y_{0\ 2,1})$
$(x_{0\ 2,0}, y_{0\ 2,0})$
d_1
$(x_{0\ 1,0}, y_{0\ 1,0})$
$(x_{0\ 2,2}, y_{0\ 2,2})$
$(x_{0\ 2,3}, y_{0\ 2,3})$
d_0

Figure 3.13: Location of the first two levels of tiling points for the Gabor functions. The full set of locations is given in table 2.2; in general $d_{i+1} = d_i/2$.

3.3.3 Filtering

Since the set of Gabor functions is nonorthogonal, the filtering must be performed by least squares optimization. On the small image tiles discussed here, direct methods are far more efficient for this operation than the neural net method proposed for data compression [21]. Given n different $\mathbf{G}_j$'s, the filtering operation is based on obtaining a least-squares fit to the image $\mathbf{q}$ by forming the matrix $\mathbf{A}$, each component of which is the inner product of the form:

$$a_{ij} = \mathbf{G}_i \cdot \mathbf{G}_j \tag{3.10}$$

and the vector,

$$b_i = \mathbf{q} \cdot \mathbf{G}_i \tag{3.11}$$

and solving

$$\mathbf{b} = \mathbf{A}\mathbf{c} \tag{3.12}$$

for the filter coefficients, $\mathbf{c}$. Since the matrix $\mathbf{A}$ is the same for any given set of n Gabor functions, the matrix is factored once, and only generation of $\mathbf{b}$ and back substitution of the factored $\mathbf{A}$ matrix are required to obtain each $\mathbf{c}$. The image is converted to its filtered form:

$$\mathbf{q}' = \sum_{j=1}^{n} c_j \mathbf{G}_j. \tag{3.13}$$

The effect of the filter can be seen figure 3.3. Each input image tile contains 1024 8-bit elements. Using equations 3.10- 3.13, the reconstructed image $\mathbf{q}'$, shown in figure 3.3 is produced. This image is constructed using 32 8-bit values of c_j. The image is then thresholded at zero to yield the filtered image shown in figure 3.3.

Chapter 4

Registration

Fingerprint registration was done using the algorithm of [8]. The six flow charts of Joseph Wegstein's R92 algorithm (pp. 28-33) were implemented in parallel Fortran; the parallel implementation is designated R92-P. Results obtained from the original Fortran code of Wegstein's work were used for comparison.

Registration of fingerprints is done by consistently finding a point on a fingerprint that is well defined even after changing the viewing orientation. Once this point is found, it can be used to quickly align two fingerprints in a manner which will reduce the amount of translation needed to compare the two fingerprints. The R92 implementation that Wegstein [8] developed has hard coded or fixed variables that made it unusable with current NIST images. The equations from the R92 algorithm were not documented. During porting to a parallel machine, the code was changed to be usable with current NIST images. These changes included new input parameters and modified equations inside the code.

There are 35 parameters to this program which were analyzed to adapt the code to current NIST fingerprint images. The R92-P version used the same values as in Wegstein's R92 implementation for all the parameters except two. These two parameters reflect the change made in tiling of the fingerprint. Wegstein's R92 tiled a fingerprint into 29 columns and 29 rows. Our parallel implementation tiles NIST fingerprints into 30 columns and 32 rows with the last two columns containing no real data. R92 scans columns 3 to 27 and rows 1 to 25 while R92-P scans columns 3 to 29 and rows 1 to 30.

The tile sizes in R92-P are smaller than those in R92. This improves the accuracy and is also faster on the parallel machine since the tile sizes were selected to fully use the machine's processing array. Because of this change, all the registration point calculations had to be changed to reflect the new tile sizes.

The size of the fingerprint images also appear to be different. The exact size of the image being analyzed by R92 is unclear. R92-P uses a 512 pixel by 512 pixel image divided into tiles of 16 by 16 pixels. Figure 4.1 shows a fingerprint image used for discussion in this chapter. When determining the ridge direction in a tile, the tile is expanded into a 32 by 32 pixel tile for a more accurate ridge direction calculation and full use of the parallel architecture. When this process is carried out, a 32 by 32 matrix of ridge directions is produced.

The matrix of ridge directions obtained from figure 4.1, as illustrated in figure 4.2, is analyzed to build a K-table as shown in table 4.1. This table lists the first location in each row of the matrix where the ridge direction changes from positive slope to negative slope to produce a well-formed arch. Associated with each K-table entry are many facts about the entry that are used later to calculate the registration point of the fingerprint. The ROW and COL values are the row and column of the tile in the K-table. The SCORE is how well the arch is formed at this tile location. The BC SUM is the sum of this tile's angle with its neighbor to the east, while the AD SUM is BC SUM plus the one angle to the west and east of the BC SUM. SUM HIGH and SUM LOW are summations of groups of angles below the tile being analyzed. For these two values, five sets of four angles are individually summed, and the lowest and highest are saved in the K-table.

With the K-table filled in, each entry is then scored. The score indicates how well the arch is formed at this point. The point closest to the core of the fingerprint is intended to get the largest score. If scores are equal, the entry closest to the bottom of the image is considered the winner. Calculating a score for a K-table entry uses six angles and one parameter, RK3. RK3 is the minimum value of the difference of two angles. For this work, the parameter was set to 0.0 degrees which is a horizontal line. The six angles are the entry in the K-table, the two angles to its left and the three angles to its right. So if the entry in the K-table is (i,j), then the angles are at positions (i,j-2), (i,j-1), (i,j), (i,j+1), (i,j+2), and (i,j+3). These are labeled M, A, B, C, D, and N, respectively. For each of the differences, M-B, A-B, C-N, and C-D, which are greater than RK3, the score is increased by one point. If A has a positive slope, meaning the angle of A is greater than RK3, then the score is increased by one point and another if M-A is greater than RK3. If D has a negative slope, meaning the angle of D is less than RK3, then the score is increased by one point and another if D-N is greater than RK3. If N has a negative slope, then the score is increased by one point. All these comparisons form the score for the entry.

Using the information gathered about the winning entry a registration point is generated usually at the core of the fingerprint. Figure 4.2 shows the registration point, x = 262 and y = 215, which is marked by the large crosshair which is close to the center of the image. This point is selected by first determining if the fingerprint is possibly an arch.

If the image is an arch then the x position is determined by equation 4.1.

$$x = (\alpha * A(R,C))/(A(R,C) - A(R,C+1)) + (\alpha * (C-1)) + \beta. \tag{4.1}$$

The y position is determined by equation 4.2.

$$y = \frac{((\alpha * R + \beta) * ts(k+1) + (\alpha * (R-1) + \beta) * ts(k) + (\alpha * (R-2) + \beta) * ts(k-1))}{(ts(k+1) + ts(k) + ts(k-1))} \tag{4.2}$$

where A is the angle at an entry position, R is the row of the entry, C is the column of the entry, k is the entry number, ts is a sum of angles, α is 16.0 which is the width of a tile in pixels, and β is 8.5. The ts value is calculated by summing up to six angles. These angles are the current tiles angle and the five tile angles to the east of that tile. While the angle is not greater than 99 degrees its absolute value is added to ts. For the angles 89, 85, 81, 75, 100, and 60, the sum would be 330 (89 + 85 + 81 + 75). Since 100 is greater than 99, the summation stops at 75.

For an image that is possibly something other than an arch, the computation of the point is slightly more complex.

$$x = (dh * (xx1 - xx2))/ds + xx2 \tag{4.3}$$

$$y = (\alpha * R) - dh \tag{4.4}$$

$$dh = (dh1 + dh2)/2.0 \tag{4.5}$$

$$dh1 = \begin{cases} \alpha & \text{if } dsp1 \geq 90.0 \\ (dsp1 * \alpha)/90.0 & \text{otherwise} \end{cases} \tag{4.6}$$

$$dh2 = \begin{cases} ((dsp2 - 90.0) * \alpha)/90.0 & \text{if } dsp2 > 90.0 \\ 0.0 & \text{otherwise} \end{cases} \tag{4.7}$$

$$xx1 = (\alpha * A(R,C)/(A(R,C) - A(R,C+1)) + (\alpha * (C-1)) + \beta \tag{4.8}$$

$$xx2 = (\alpha * A(R+1, JL))/dsp2 + (\alpha * (JL - 1)) + \beta \tag{4.9}$$

$$dsp1 = A(R,C) - A(R,C+1) \tag{4.10}$$

$$dsp2 = A(R+1, JL) - A(R+1, JL+1) \tag{4.11}$$

where A is the angle at an entry position, R is the row of the entry, C is the column of the entry, JL is the cross-reference point column, α is 16.0, and β is 8.5.

The registration point is used in later work as a common point for the translation of all images. Each fingerprint has the registration point generated. Then the mean of all the registration points is used as the common point. Each image is then translated so that its registration point is now overlying the common point.

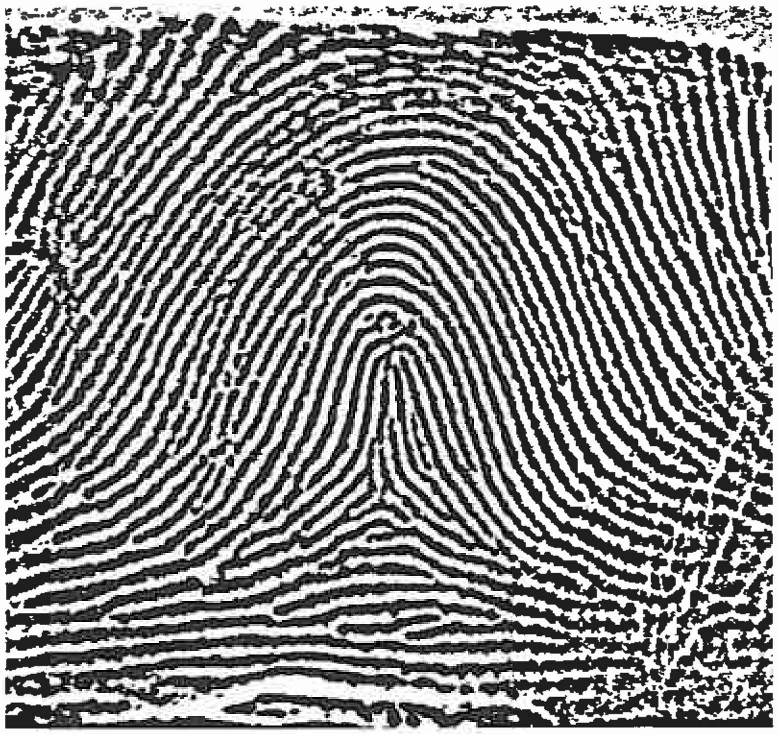

Figure 4.1: A fingerprint image, a binarized version of figure 1.4.

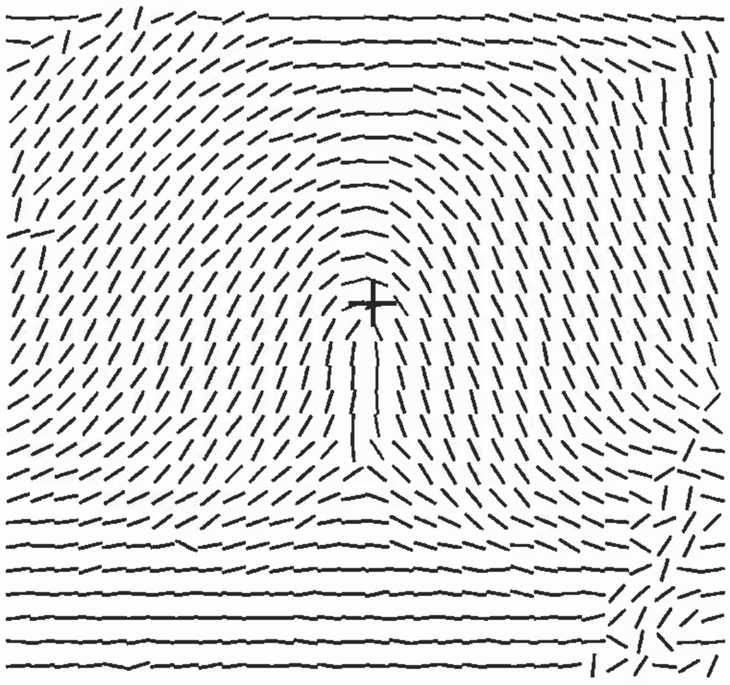

Figure 4.2: Ridge direction matrix of the fingerprint in figure 4.1.

ROW	COL	SCORE	BC SUM	AD SUM	SUM HIGH	SUM LOW
8	17	10	20.21	58.85	161.0733	59.868
9	16	10	10.69	59.87	137.8646	86.097
10	16	10	25.17	86.10	166.6743	101.759
11	16	10	23.84	101.76	179.6504	98.188
12	16	10	29.07	98.19	187.3843	117.861
13	16	10	30.54	117.86	207.0527	142.808
14	17	10	65.91	65.91	263.8445	244.043

Table 4.1: K-table for Figure 4.2.

Chapter 5

K-L Transform

5.1 Introduction

Each raw fingerprint image contains 256,000 8-bit pixels of data. The ridge-valley feature extraction procedure can reduce this to 1620 ridge direction components. This is still far too large to be used in supervised training on a MLP network. A fully connected network with 1620 inputs, 200 hidden nodes, and 5 class outputs would require over 337,000 weights. Since generalization capacity falls and training set size increases with network size [22], it is essential to decrease the number of inputs to the MLP. The K-L transform [3] provides a method for this.

The discrete Karhunen Loève transform (KLT) [23] assumes no model of the human perception mechanism, but directly references statistically salient information in the fingerprint set. The eigenvectors of the covariance matrix of the fingerprint ensemble are taken as a minimal orthogonal basis set, of which any fingerprint is a linear superposition. The eigenvectors are the principal statistical components of the variance in the original image space. Their respective eigenvalues indicate the significance of the eigenvector in describing the fingerprints' construction: those with the smallest eigenvalues represent irrelevancies. The motivation for doing this lies not only in the well-documented optimality of the KLT, but in recent studies [24] showing that evolution of synaptic structures in linear Hebbian neural networks [4] is dynamically governed by the same statistical basis as that of the KLT.

Although eigenvectors appear in some unsupervised network training [25], they are most readily obtained using one of the traditional numerical iterative methods [26]. The eigenvectors can therefore be regarded as a trained weight layer. The use of eigenvectors is rather a *prescription* of this feature extraction layer derived as a leastmean-square fit to the data. This is potentially detrimental to the perceptron as a classifier but it yields pragmatic gains. Perceptron networks are known to exhibit better generalization if the training sets are large. Rather than use raw images as input it is preferable to use greater numbers of precomputed low dimensional K-L transforms for training.

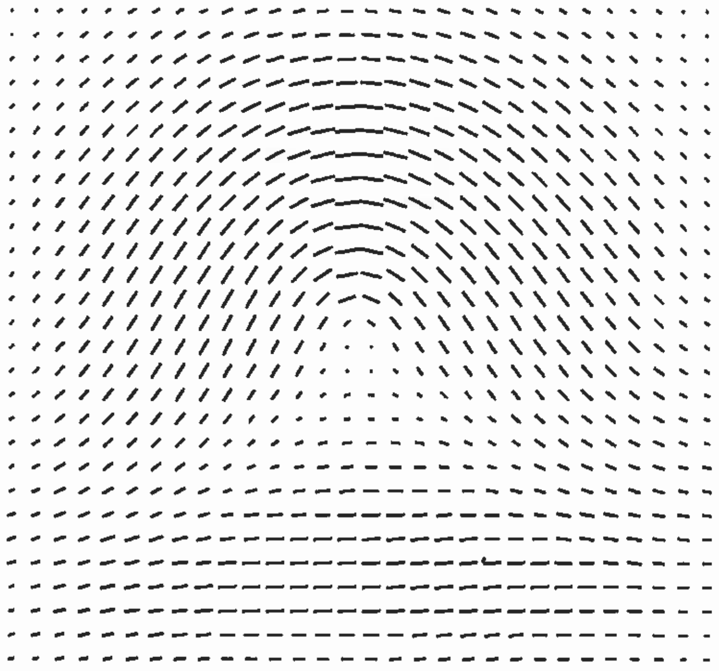

Figure 5.1: The mean ridge flow for a set of fingerprint images.

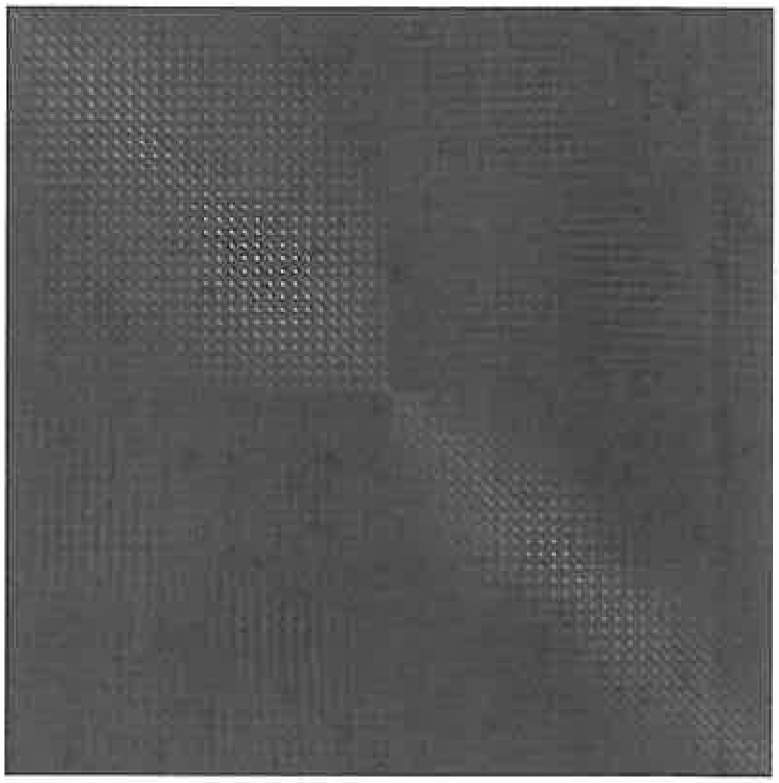

Figure 5.2: The covarience matrix for a set of fingerprint images.

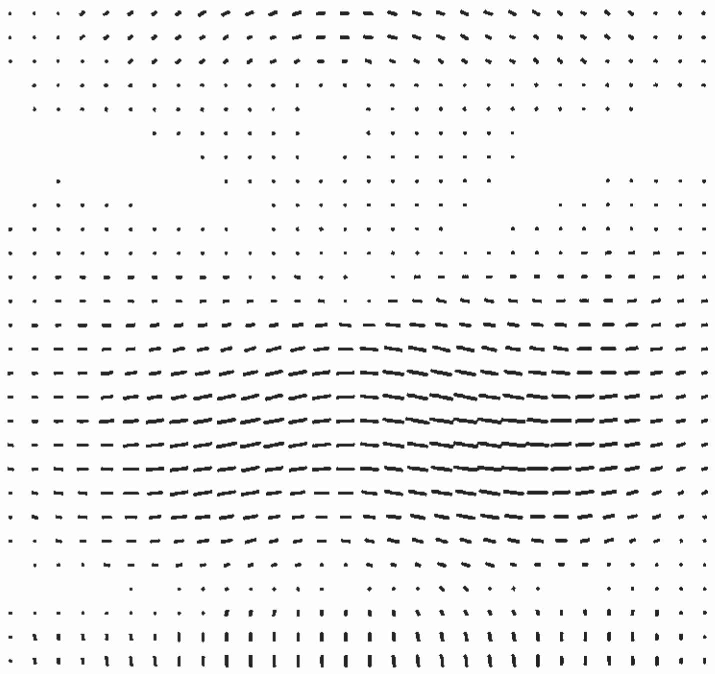

Figure 5.3: The principal eigenvector for a set of fingerprint images.

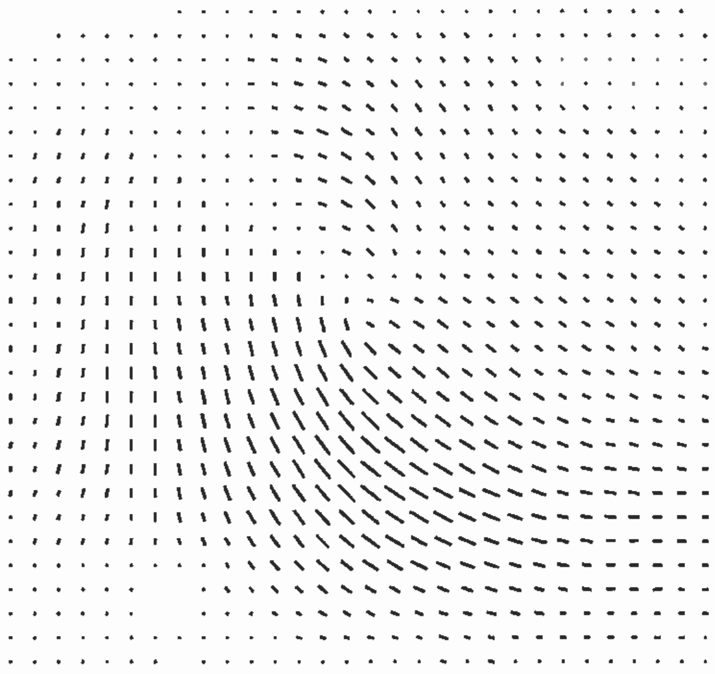

Figure 5.4: The second eigenvector for a set of fingerprint images.

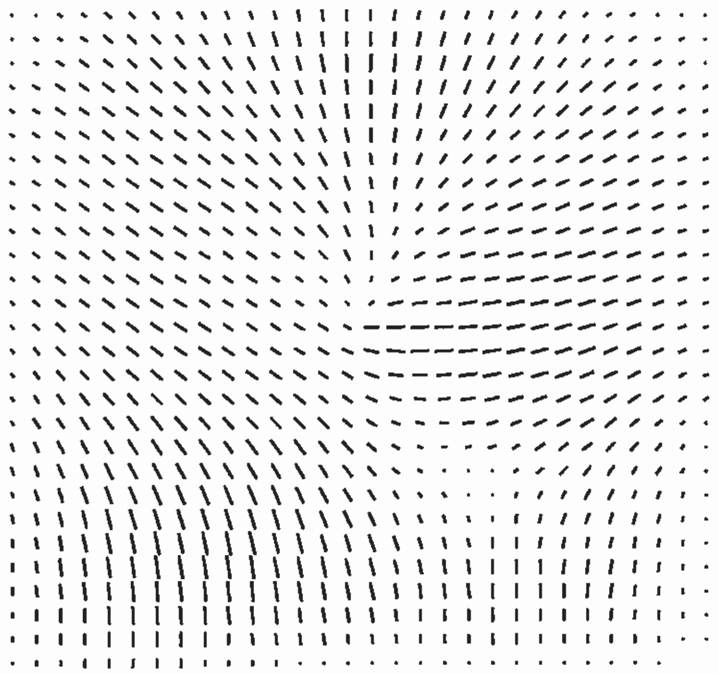

Figure 5.5: The third eigenvector for a set of fingerprint images.

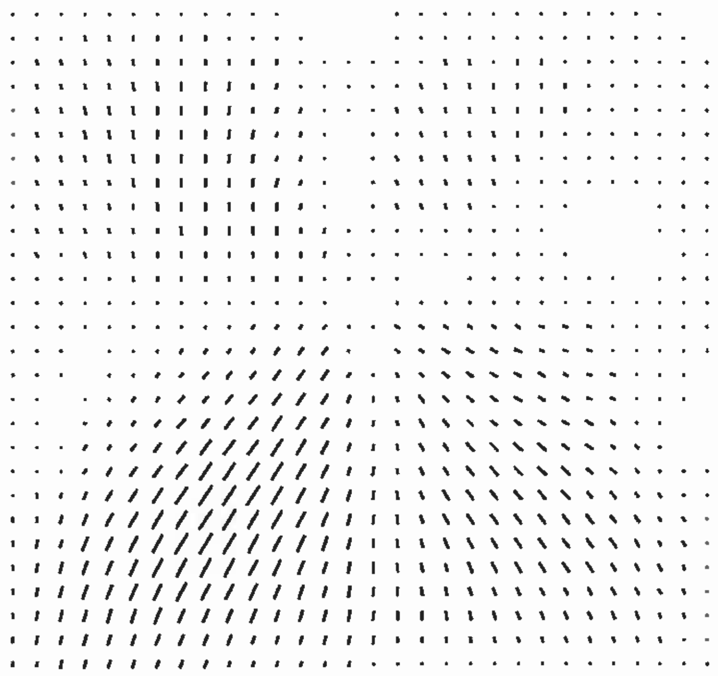

Figure 5.6: The fourth eigenvector for a set of fingerprint images.

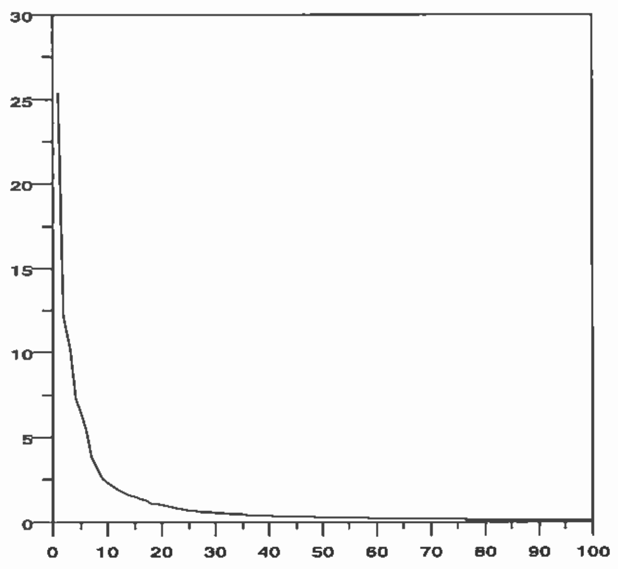

Figure 5.7: The eigenspectrum for a set of fingerprint images. The eigenvalue has dropped from 25.42 for the first, principal, eigenvalue to 7.25 for the fourth eigenvalue.

5.2 Statistical Representation

Consider that a sample of fingerprints which has direction available as computed using the method discussed in chapter 3. Each of the P images $u^{(p)}$ has $N/2$ x and $N/2$ y directions. The p^{th} finger vector is regarded as a real matrix $\mathbf{u}^{(p)}$ such that its elements are given thus

The covariance matrix gives the mean, overall images in the ensemble, of all the N by N direction correlations in each image and, as such, statistically describes how fingerprint images vary. The real symmetric covariance matrix $\mathbf{R}$ is formed as the sample expectation of the outer product of P such vectors.

$$\mathbf{R} = \frac{1}{P}\sum_{p=1}^{P}(\mathbf{u}^{(p)} - \mathbf{u})(\mathbf{u}^{(p)} - \mathbf{u})^T \tag{5.1}$$

where the vector $\mathbf{u}$ is taken as the vector mean of each element over the P image vectors

$$\bar{\mathbf{u}} = \frac{1}{P}\sum_{p=1}^{P}\mathbf{u}^{(p)}. \tag{5.2}$$

Considering the N^2 vectors $\mathbf{u}^{(p)} - \mathbf{u}$ to be the P columns of a matrix $\mathbf{U}$, the above equations can be compactly represented as

$$\mathbf{R} = \mathbf{U}\mathbf{U}^T. \tag{5.3}$$

The computation of $\mathbf{R}$ is reduced by extracting $\mathbf{u}$ from the sum in equation (5.1) to obtain

$$\mathbf{R} = \mathbf{R_o} - \mathbf{u}\mathbf{u}^T \tag{5.4}$$

where $\mathbf{R_o}$ is the correlation matrix of the vectors $\mathbf{u}$. The covariance and correlation matrices are only identical if the mean vector is itself zero. The mean image of equation (5.2) is shown in figure 5.1. The covariance matrix itself is shown in figure 5.2.

5.3 K-L Transform Representation

The matrix Ψ has the N eigenvectors, ψ_i, as its columns and is defined in the equation

$$\mathbf{R}\Psi = \Psi\Lambda \tag{5.5}$$

where $\mathbf{R}$ is the covariance matrix, and the only nonzero elements of Λ are the eigenvalues λ_i on its diagonal. The eigenvector matrix is orthogonal since, by definition, it diagonalizes the matrix $\mathbf{R}$

$$\Psi^T\mathbf{U}\mathbf{U}^T\Psi = \Lambda. \tag{5.6}$$

Given that the directions are vectors in an N^2 space, the eigenvectors of the covariance matrix are the directions of maximum variance in that space. They are mutually orthonormal and thereby define the principal axes [1] of a hyperellipse in that space. The eigenvalues $diag(\Lambda)$ define the statistical length of these axes as defined by the direction set; thus the first column of Ψ corresponding to the largest eigenvalue is the major axis. Any N^2 vector $\mathbf{u}$ in this space can be expressed as a linear combination of the basis vectors thus:

$$\mathbf{u}^{(p)} = \Psi \mathbf{v}^{(p)} \tag{5.7}$$

where the inversion of this formula, $\mathbf{v}^{(p)}$, defines the Karhunen Loève transform of $\mathbf{u}^{(p)}$, the elements of which are the projection of the image vector $\mathbf{u}^{(p)}$ onto the principal axes Ψ.

$$\mathbf{v}^{(p)} = \Psi^T \mathbf{u}^{(p)}. \tag{5.8}$$

This equation applies to all finger vectors $\mathbf{u}^{(p)}$ and is therefore compactly represented as

$$\mathbf{V} = \Psi^T \mathbf{U} \tag{5.9}$$

where again the columns of $\mathbf{U}$ and $\mathbf{V}$ are the image and transform vectors respectively. The first four eigenvectors of a covariance matrix are shown in figures 5.3 to 5.6.

[1]The Karhunen Loève transform is also known as the method of principal components or the Hotelling transform.

5.4 Transform Coefficient Decorrelation

The KL transform vectors in the columns of **V** are to be used as input to some classifier. The variance of the KL coefficients themselves is of interest.

$$\mathbf{R_v} = \mathbf{V}\mathbf{V}^T = \Psi^T \mathbf{U}\mathbf{U}^T \Psi = \Lambda. \tag{5.10}$$

The last term of this equation is the diagonal eigenvalue matrix. This implies that the K-L correlation matrix is diagonal. Thus, by design, the Karhunen Loève transform coefficients are perfectly decorrelated. Further, the variances of these coefficients, σ_i^2, are the respective eigenvalues of the original covariance matrix; that is

$$\sigma_i = \sqrt{\lambda_i}. \tag{5.11}$$

Chapter 6

Classification of Fingerprints

6.1 The Layered Perceptron Networks

The two-weight layer[1] perceptron nonlinearly classifies K-L feature vectors. With the evolved K-L feature extraction, the network may be regarded as the three layer fingerprint classifier of figure 6.1. The first set of weights Ψ_0 is the ***pre-trained*** incomplete eigenvector basis set of equation 5.9. The latter perceptron weight layers, also fully interconnected, are trained using the conjugate gradient algorithm outlined in section [7].

All the Karhunen Loève transform vectors are propagated through the network together and the weights are updated. This is *batch mode* training. The use of different subsets of the training patterns to calculate each weight update is known as *on line* training. It is not used in this investigation. Formally, the forward propagation is represented as:

$$\mathbf{V_1} = \mathbf{\Psi_0^T V_0} \Longrightarrow \mathbf{V_2} = \mathbf{f}\left(\mathbf{\Psi_1^T V_1}\right) \Longrightarrow \mathbf{V_3} = \mathbf{f}\left(\mathbf{\Psi_2^T V_2}\right) \tag{6.1}$$

where the network nonlinearity is introduced by squashing all activations with the usual sigmoid function $f(x) = (1 + e^{-x})^{-1}$.

[1]The authors have elected to resolve the ambiguity in counting either layers of weights or layers of neurons, pervasive throughout the literature, by adopting the standard of counting weight layers.

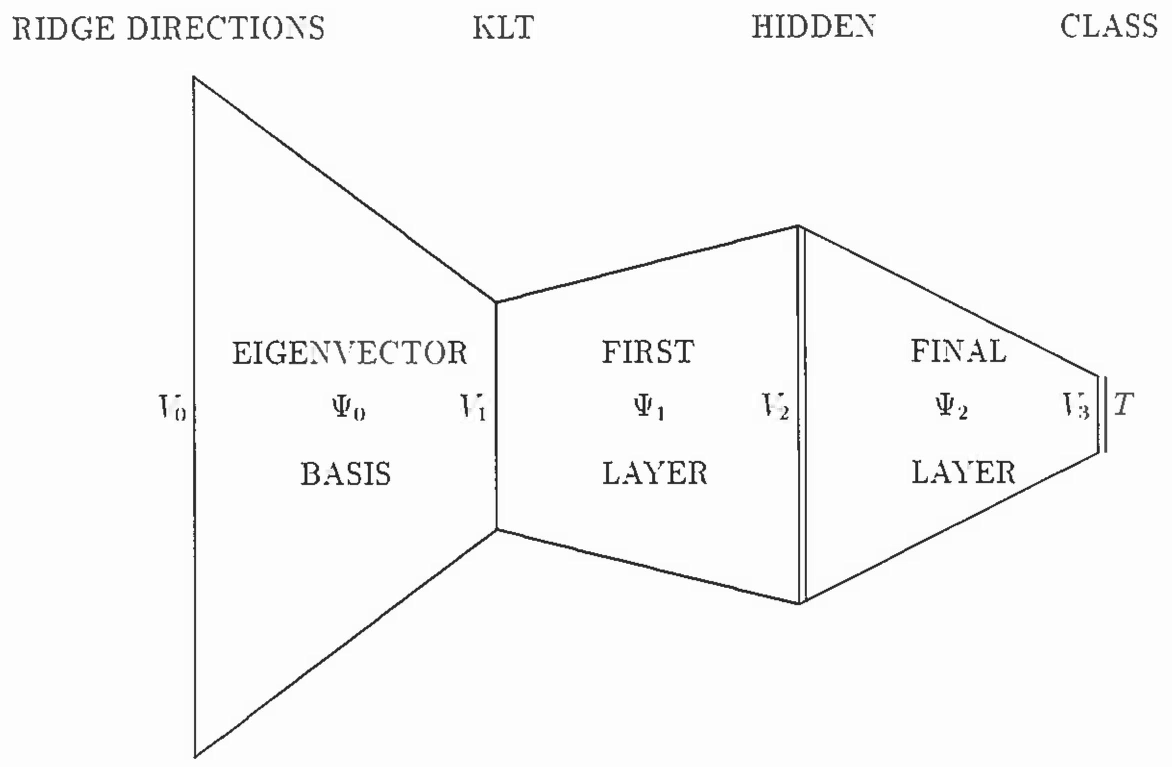

Figure 6.1: Classification Architecture. All weight layers are fully connected. The eigenvectors are obtained a priori to the training of the subsequent layers.

6.2 Classification of Fingerprints

The linear superposition of a complete set of orthogonal basis functions will exactly reproducean arbitrary eigen fingerprint. However, the whole motivation for using the KLT is to reduce the dimensionality of the feature space by adopting an incomplete basis, i.e., the leading principal components. Only images that resemble the original training ridge-valley directions are adequately representable by the reduced basis. It is important that the eigenvectors are obtained from a statistically large sample since adequate statistical sampling of the feature set is required for this reduction to be useful.

6.3 Conjugate Training Algorithm for Feed Forward Neural Networks

Backpropagation [14] is the common method for training multilayer perceptron networks. Essentially, it implements a first order minimization of some error objective. The algorithm has the disadvantages that convergence is slow [27] and that there are, in the usual implementation [14], two adjustable parameters, η and α, that have to be manually optimized for the particular problem.

Conjugate gradient methods have been used for many years [28] for minimizing functions, and have recently [29] been discovered by the neural network community. The usual methods require an expensive line search or its equivalent. Møller [30] has introduced a scaled conjugate gradient method; instead of a line search, an estimate of the second derivative along the search direction is used to find an approximation to the minimum error along the search direction. In both backpropagation and scaled conjugate gradient, the most time-consuming part of the calculation is done by the forward error and gradient calculation. In backpropagation this is done once per iteration. Although the scaled conjugate gradient method does this calculation twice per iteration, the factor of two overhead is algorithmically negligible since convergence is an order of magnitude faster [30] [7].

6.4 Training and Testing

The number of training patterns was fixed at 2000 except in those experiments where shifted versions of the set of 2000 were used. This set was comprised of equal numbers of each class. Different starting weights yield alternative minima corresponding to a distribution of network performance. Training was performed using uniformly distributed (on the range [-0.5,+0.5]) initial random weights. The target activations were 0.0 for all nodes except for a 1.0 on the node representing the given class. The objective function included a regularization [31] term, the square weight vector length.

Testing used the K-L transformation of 2000 fingerprints which were different rollings obtained from the same fingers. This set was disjoint from the training set. The characters from which they were obtained were not used in the calculation of the covariance matrix or its eigenvector basis set. Classification involves a single forward pass through a set of weights. The true classes are known *a priori* so that the generalization properties of the classifier are obtained.

6.5 Single Network Classification

The classification features are passed through a K-L transform to extract maximum variance features in much the way described above. This reduces the feature set from 1620 to 96 features. These reduced feature set is used to train a MLP using SCG optimization. Hidden layer values of 32 to 96 nodes were tested. The output layer contains five nodes, one for each class. Typical classification accuracy is 81% with no rejects, to 88% with 10% rejects.

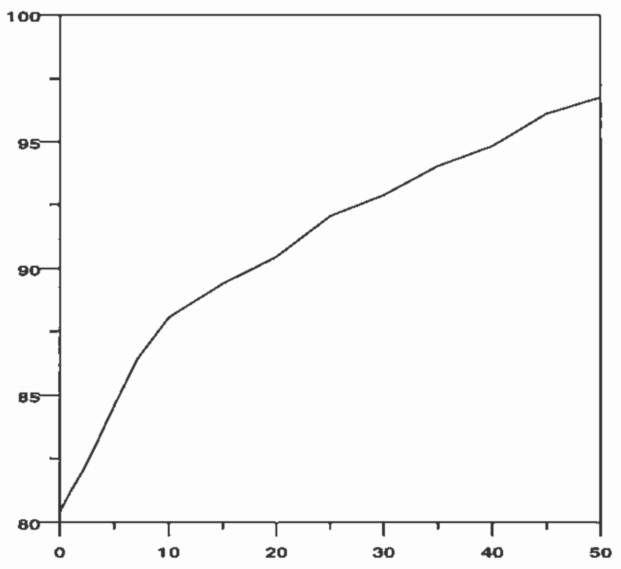

Figure 6.2: Reject versus accuracy curve for 48-96-5 netwotk.

Table 6.1 shows recognition rates for 20 different network configurations. The reject versus accuracy curve for the best 48-96-5 network is shown in figure 6.2. As the network complexity increases the accuracy on 2000 prints goes from 76% to 80% with typically 5% improvement from registration. The best network has 5200 weights. This is over 2.5 degrees of freedom per training sample. This suggests that the net is memorizing rather than generalizing to achieve the classification.

The shape of the reject-versus-accuracy curve is also indicative of the nature of the accuracy loss. The reject accuracy curve suggests that even at 5,000 weights, the problem may be generating too simple a classification surface to be accurate or that the sharpness of the surface must be refined by further sampling. An alternative approach is to use a more complex set of networks as discussed in the next section.

The sample size is too small to provide good generalization. Further regularization and network pruning may be helpful, and additional features will be required to improve accuracy. If this is primarily a statistical inference problem, then reducing the error from 10% to 1%, a factor of 9, would require a sample size increase of 81.

no. inputs	no. hiddens	pct. correct
32	32	76.37
32	48	75.99
32	64	78.85
32	80	78.42
32	96	78.85
48	32	76.85
48	48	78.04
48	64	79.28
48	80	79.43
48	96	80.05
64	32	78.33
64	48	79.62
64	64	79.38
64	80	79.14
64	96	80.38
80	32	79.33
80	48	79.81
80	64	79.71
80	80	79.09
80	96	78.62

Table 6.1: Percentages correct for various network architectures using a single network.

6.6 Single Network Using Shifted Data

In an attempt to increase the effectiveness of the conjugate gradient algorithm, the ridge-valley features of the fingerprint database set were shifted to all adjacent grid points, king shifted, increasing the training set to nine times its original size (see table 6.2). The first classification experiments were done with unregistered fingerprints, and the results were similar to those with no king shifting, but with registered the fingerprints. Four of the experiments were run involving training on all 18000 patterns together and then training on all 18000 patterns in different size groups. Each group of 2000 patterns was tested such that the different king-shifted groups were kept together for training (i.e. north, south, northwest, ..). The groups of 4500 patterns shuffled the different king shifted groups together before testing. In all of these tests, attempts were also made to combine registering and king shifting the patterns, but the results have not shown a significant improvement over king shifting or registering alone. The reject-accuracy curve for the best of these tests is shown in figure 6.3.

This sequence of tests shows that some advantage is obtained by using registration and that king shifting removes some of this advantage. The advantage is not in initial unrejected accuracy but in a much more rapid rise in the reject-accuracy curve. At 10% rejects, the method from section 7.1 yields 87% accuracy, while at 10% rejects king shifting yields 82% accuracy.

classes	king	noking	reg	unreg	train pats	test pats	correct	kl inps	hids
5	x			x	18000	2000	76.05%	64	64
5	x		x		18000	2000	78.15%	64	64
5	x			x	18000	2000	77.40%	80	80
5	x		x		18000	2000	78.80%	80	80
5	x			x	9 by 2000	2000	76.05%	64	64
5	x		x		9 by 2000	2000	77.65%	64	64
5	x		x		4 by 4500	2000	72.20%	64	64
5	x		x		4 by 4500	2000	69.90%	80	80

Table 6.2: Test results using king shifting and registration combinations.

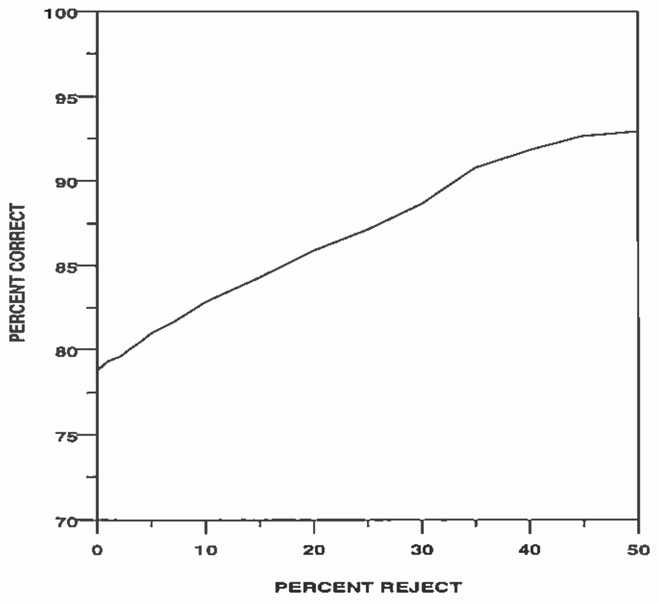

Figure 6.3: Reject versus accuracy curve for the best king-shifted network.

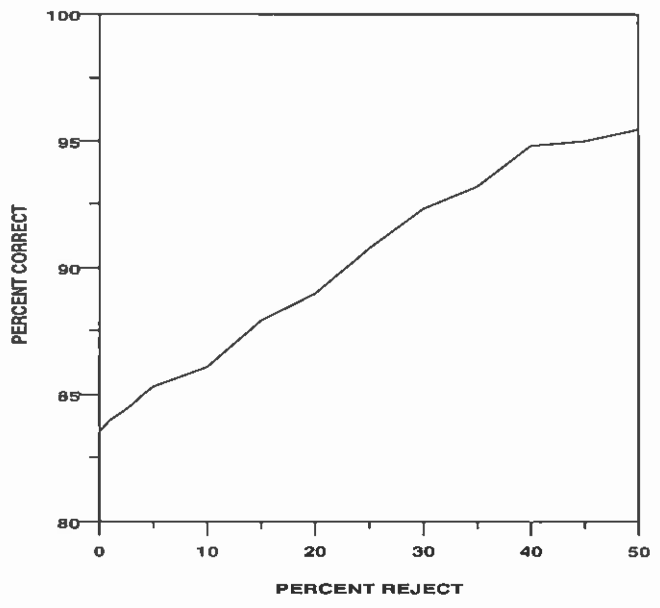

Figure 6.4: Reject-versus-accuracy curve for 80-80-5 network for class AT.

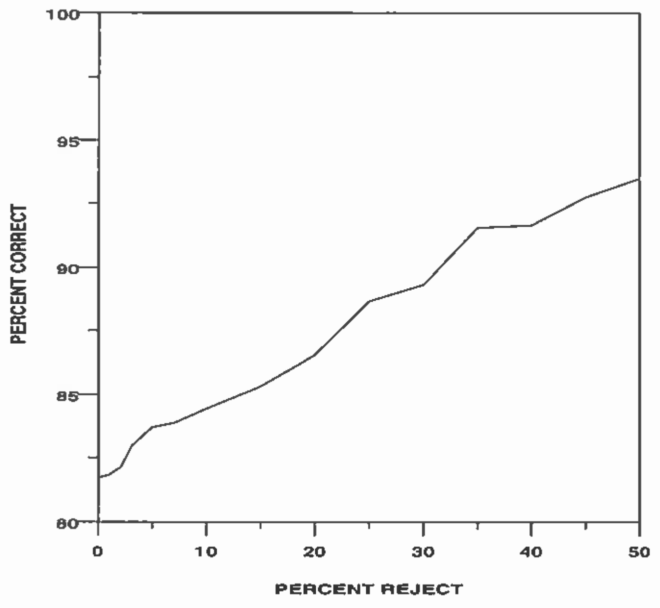

Figure 6.5: Reject-versus-accuracy curve for 48-96-5 network for class A.

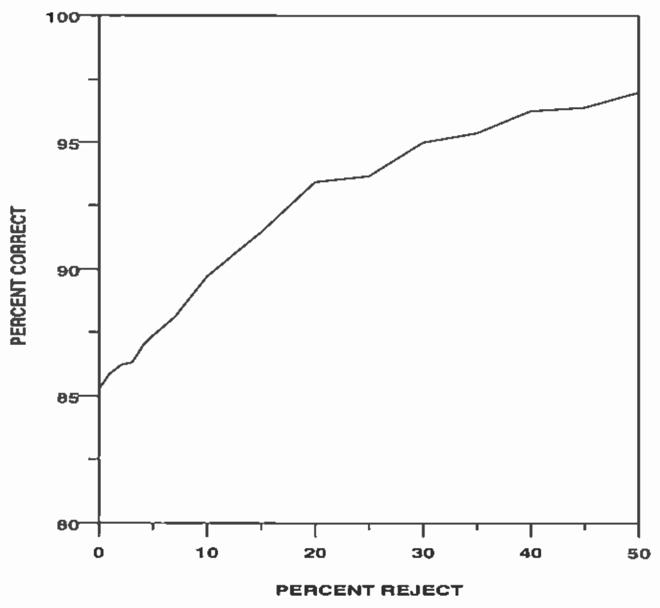

Figure 6.6: Reject-versus-accuracy curve for 80-80-5 network for class L.

6.7 Two Network Methods

A network was tested which was created by combining the Arch and Tented arch classes and then attempting to reclassify these prints in a second run (see table 6.3). The numbers shown in table 6.3 were tests run using a set of known Arches and Tented Archs not the output of a four-class system. These tests were run to see how well the two classes would separate before working with the four-class system.

classes	king	noking	reg	unreg	train pats	test pats	correct	kl inps	hids
2		x	x		838	838	82.94%	64	64
2		x	x		838	838	84.25%	64	80
2		x	x		838	838	85.20%	80	80
2		x	x		838	838	83.77%	48	48
2		x	x		838	838	84.61%	48	96
2	x		x		7524	838	81.74%	64	64

Table 6.3: Test results for separating arches and tented arches.

A system using only four classes was able to get about 86.06% correct (see table 6.4). This four-class net had 200 known arches and 200 tented arches combined to produce the arch class.

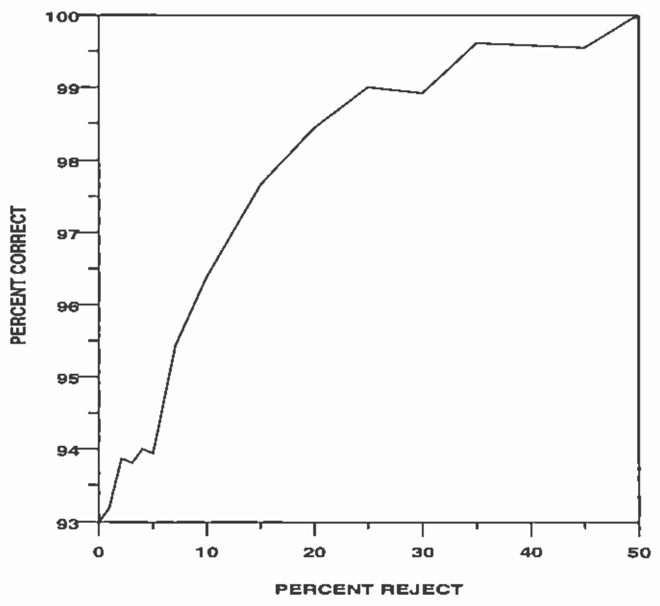

Figure 6.7: Reject-versus-accuracy curve for 80-80-5 network for class W.

When the output of what the four-class net called arches was put into the two-class net, the two-class net was able to get 99.67% correct, which made the overall percent correct 86%.

When the test was run using an even number of prints from each of the five classes nets the four-class net was still able to get 85.70% correct, but the two-class net was not as effective in separating arches from tented arches, only getting 85.05% correct. This made the overall percent correct 80.7%.

The difficult of classification of each of the classes used in this test can be seen by plotting reject-accuracy curves for each class. These curves are shown in figures 6.4 to 6.8. The arch class in figure 6.5 is the most difficult to classify. The right and left loops in figures 6.8 and figure 6.6 are next most difficult, and the whorl in figure 6.7 class is simplest to classify.

classes	king	noking	reg	unreg	train pats	test pats	correct	kl inps	hids
4	x		x		14400	1600	84.12%	64	64
4	x		x		14400	1600	86.06%	80	128
4		x	x		1600	2000	85.70%	80	128
2		x	x		838	305	99.96%	80	80
2		x	x		838	669	85.05%	80	80

Table 6.4: Test results for separating arches and tented arches from a four class net.

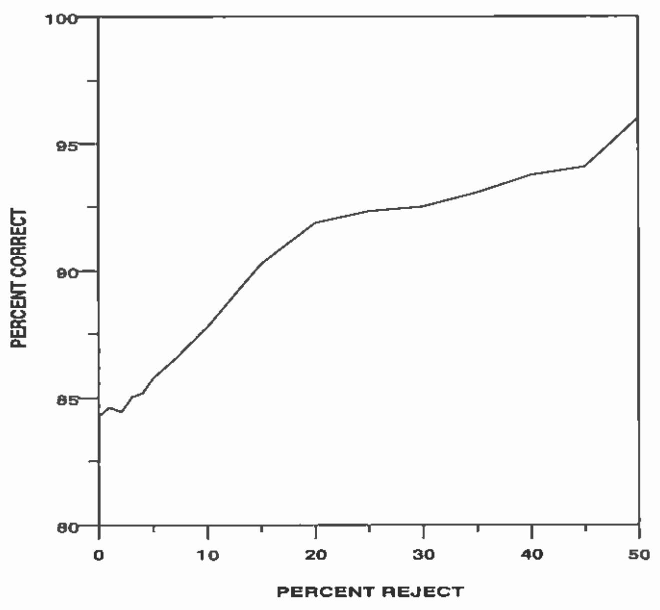

Figure 6.8: Reject-versus-accuracy curve for 80-80-5 network for class R.

Chapter 7

Conclusions

7.1 Accuracy

Accuracy is the most difficulty part of the PCA problem. The best accuracy achieved to date on a balanced sample, with equal numbers of A-T-L-R-W patterns, is 89.3% with 10% rejects. This is achieved by combining the A and T classes into a single class. This class is then separated with 99.6% accuracy, without rejects, using a second network. To put this result into perspective, 89% is approximately the level of accuracy achieved with handprinted digits using Gabor features [32] in early 1991. Results achieved on the handprint digit problem, by expanding the training and testing sets and by using better segmentation and feature extraction, have allowed accuracy on character recognition to improve to 98.96% with 10% rejects. This suggests that a PCA system can be built which will achieve 99% accuracy.

7.2 Speed

The speed achieved in this study, 0.5 seconds per fingerprint, demonstrates that existing parallel computers are fast enough to process the FBI's current workload with a small number of systems. This speed is also essential for large scale testing of potential recognition methods. If feature extraction using ridge directions or some other method takes 1000 seconds per fingerprint instead of 1 second per fingerprint, the training time for the existing database goes from 20 minutes to over 550 hours, or 23 days. This demonstrates that the extensive training and testing performed on 2000 fingerprint samples required for this study is practical only if speeds near the FBI current operative requirement are achieved.

7.3 Testing Requirements

All work performed in this study has used NIST Special Database 4 as described in the appendix of this report. The sample size available from this database should be sufficient to test PCA accuracy to about the 2-3% error level. We estimate that further testing to reduce error to the 0.3% level will require sample sizes from 25 to 81 times as large as the one presently available.

7.4 System Design

The present version of the system has been designed in a modular way so that different versions of each of the modules can be tested independently. This has allowed three methods of ridge direction feature extraction to be tested and has allowed K-L transforms and MLPs of several different sizes to be used. This modular structure will allow the addition of a reject-testing method after the classification section: it has allowed networks to be developed that lump all arches into a single class at first and that separate A-T combinations with a second network. The modular structure separating image processing from classification has demonstrated its usefulness and should be retained. Substantial cost savings can result from this type of program design. The present system at NIST was constructed with about two staff-years of programing effort. This was possible by building on approximately 12 staff-years of effort spent on several character recognition system configurations [33]. This transfer of software expertise was possible because of the modular design of both systems. This modular approach should be adopted in PCA to allow transfer of technology from areas of pattern recognition other than character recognition.

Acknowledgement

The authors would like to acknowledge Jim Blue, Jane Walters, Jon Geist, Mike Gilcrest, and Fred Preston for assistance in improving the presentation and clarity of this report.

Appendix A

Database Specifications

A.1 Introduction

This chapter describes the NIST fingerprint database, NIST Special Database 4, which contains 8-bit gray scale images of randomly selected fingerprints. The database has been distributed for use in the development and testing of automated fingerprint classification systems on a common set of images. The CD-ROM contains 4000 (2000 pairs) fingerprints stored in NIST's IHead raster data format [34] and compressed using a modified JPEG lossless compression algorithm [35]. Each print is 512 by 512 pixels with 32 rows of white space at the bottom of the fingerprint. Approximately 636 Megabytes of storage are needed when the prints are compressed whereas 1.1 Gigabytes are needed when uncompressed (1.6 : 1 average compression ratio).

The fingerprints are classified into one of five categories (L = left loop, W = whirl, R = right loop, T = tented arch, and A = arch) with an equal number of prints from each class (400). Each filename contains a reference to the hand and digit number so the classes can be converted to other classification techniques (i.e., radial and ulnar). All classes are stored in the NIST IHead id field of each file, allowing for comparison with hypothesized classes.

A.2 Compression

The compression used was developed from techniques outlined in the WG10 "JPEG" (draft) standard [35] for 8-bit gray scale images with modifications to the compressed data format. The NIST IHead format already contained most of the information needed in the decompression algorithm, so the JPEG compressed data format was modified to contain only the information needed when reconstructing the Huffman code tables and identifying the type of predictor used in the coding process. Codes used to compress and decompress the images are still developed per the draft standard, but only applied to 8-bit gray scale images.

The standard uses a differential coding scheme and allows for seven possible ways of predicting a pixel value. Tests showed that predictor number 4 provided the best compression on up to 99.9% of the fingerprint images; therefore, this predictor was used to compress all of the images.

A.3 Reflectance Calibration

The reflectance values for the fingerprint data set in NIST Special Database 4 was calibrated using a reflection step table [36]. Shown below is an equation used to predict the reflectance of a given data point. This predicted reflectance closely follows the actual reflectance obtained using the reflection step table.

$$\text{predicted\%reflectance} = 10 + (.32 * \text{grayscale} - \text{pixel} - \text{value}).$$

A.4 Database Organization

NIST Special Database 4 contains 4000 8-bit gray scale fingerprint images stored in the data directory (see figure A.1). The images, which use approximately 636 Megabytes of storage compressed and 1.1 Gigabytes of storage uncompressed, are distributed on a ISO-9660 formatted CD-ROM and compressed using a modified JPEG lossless compression algorithm. Included with the fingerprint data are software and documentation.

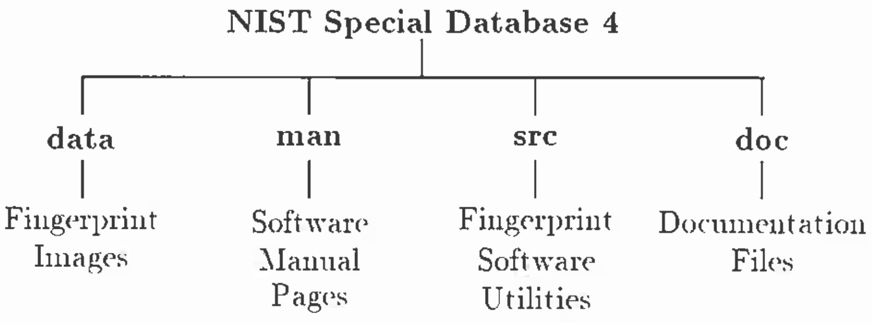

Figure A.1: Top level directory tree for NIST Special Database 4.

A.4.1 Database File Hierarchy

The top level of the file structure contains four directories doc, src, man, and data. The code needed to decompress and use the image data is contained in the src directory, with man pages for the source code stored in the man directories. Documentation for the CD-ROM is in the doc directory. The data directory contains the fingerprint images stored in eight subdirectories for easier access (see figure A.2). Each subdirectory contains 250 fingerprint pairs (two different rollings of the same fingerprint). Fingerprints are stored with filenames containing one letter, four digits, an underscore then two more digits, and a ".pct" extension. The first character in the file name is always an "f" or "s" distinguishing the first and second rollings of each fingerprint. The next four characters indicate the print number (i.e. 0001 to 2000), and the final two digits represent the finger number in the same order as on a fingerprint card (see figure A.3).

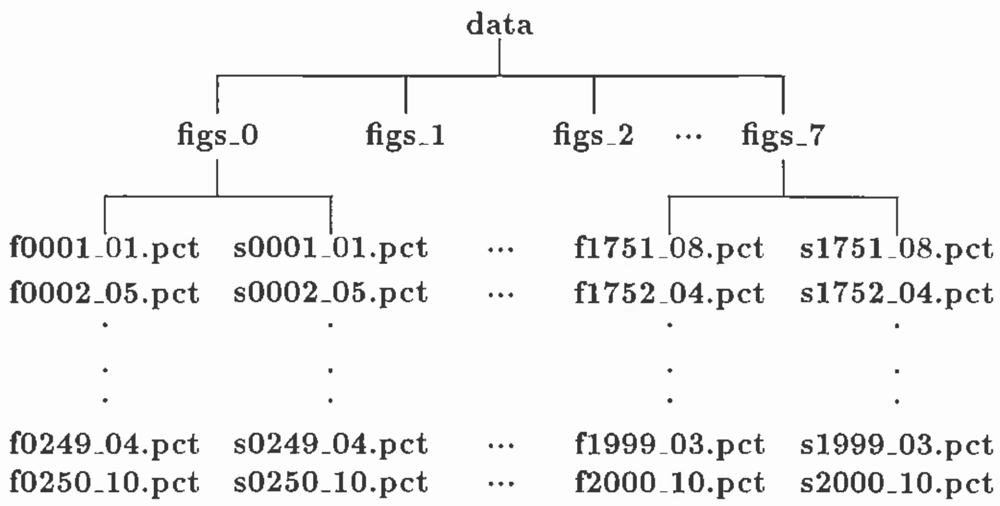

Figure A.2: Arrangement of fingerprint files for NIST Special Database 4.

1. R. Thumb	2. R. Index	3. R. Middle	4. R. Ring	5. R. Little
6. L. Thumb	7. L. Index	8. L. Middle	9. L. Ring	10. L. Little

Figure A.3: Layout of fingerprint card numbers.

A.4.2 Cross-Referenced Fingerprints

The cross-referencing of a fingerprint is caused by a variety of ambiguities such as a scar occurring in the fingerprint, the quality of the print rolling, or the print having ridge structures characteristic of two different classes. The cross-referenced prints could easily cause a wrong classification when used in testing an automatic classification system, but could provide a challenge in the later stages of development. NIST Special Database 4 contains 350 fingerprint pairs (17.5%) whose classifications are cross-referenced to a second classification. The id fields of these prints contain the fingerprint classifications immediately followed by the cross-referenced class.

Bibliography

[1] *The Science of Fingerprints*. U. S. Department of Justice, Washington, D. C., 1984.

[2] C. I. Watson and C. L. Wilson. Fingerprint database. *National Institute of Standards and Technology*, Special Database 4, **FPDB**, April 18, 1992.

[3] P. J. Grother. Karhunen Loève feature extraction for neural handwritten character recognition. In *Proceedings: Applications of Artificial Neural Networks III*. Orlando, SPIE, April 1992.

[4] R. Linsker. Self-organization in a perceptual network. *Computer*, 21:105–117, 1988.

[5] M. V. Wickerhauser. Fast approximate factor analysis. In *Proceedings October 1991*. SPIE, Washington University in St. Louis, Department of Mathematics, 1991.

[6] T. P. Vogl andm K. L. Blackwell, S. D. Hyman, G. S. Barbour, and D. L. Alkon. Classification of Japanese Kanji using principal component analysis as a preprocessor to an artificial neural network. In *International Joint Conference on Neural Networks*, volume 1, pages 233–238. IEEE and International Neural Network Society, 7 1991.

[7] J. L. Blue and P. J. Grother. Training feed forward networks using conjugate gradients. In *Proceedings February 1992*. SPIE, National Institute of Standards and Technology, Gaithersburg Md., 1992.

[8] J. H. Wegstein. An automated fingerprint identification system. *National Institute of Standards and Technology*, NBS Special Publication 500-89, February 1982.

[9] F. Rosenblat. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[10] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophysics*, 9:115–133, 1943.

[11] M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, Mass., 1969.

[12] R. P. Lippman. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, April 1987.

[13] J. A. Anderson and E. Rosenfeld. *Neurocomputing*. MIT Press, Cambridge, Mass., 1989.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, et al., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, chapter 8, pages 318-362. MIT Press, Cambridge, 1988.

[15] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, second edition, 1988.

[16] S. Grossberg. On learning and energy-entropy dependence in recurrent and nonrecurrent signed networks. *J. Statistical Physics*, 1:319-350, 1969.

[17] J. A. Anderson, M. L. Rossen, S. R. Viscuso, and M. E. Sereno. Experiments with the representation in neural networks: Object, motion, speech, and arithmetic. In H. Fanken and M. Stadler, editors, *Synergetics of Cognition*, pages 54-69. Springer-Verlag, Berlin, 1990.

[18] P. H. Winston. *Artificial Intelligence.* Addison-Wesley, Reading, Mass., 1984.

[19] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109-118, 1990.

[20] R. M. Stock and C. W. Swonger. Development and evaluation of a reader of fingerprint minutiae. *Cornell Aeronautical Laboratory*, Technical Report CAL No. XM-2478-X-1:13-17, 1969.

[21] J. G. Daugman. Complete discrete 2-d Gabor transform by neural networks for image analysis and compression. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 36:1169-1179, 1988.

[22] I. Guyon, V. N. Vipnick, B. E. Boser, L. Y. Botton, and S. A. Solla. Structural risk minimization for character recognition. In R. Lippmann, editor, *Advances in Neural Information Processing System*, volume 4. Morgan Kauffman, 1992.

[23] Anil K. Jain. *Fundamentals of Digital Image Processing*, chapter 5.11, pages 163-174. Prentice Hall Inc., 1989.

[24] D. J. C. MacKay and K. D. Miller. Analysis of Linsker's simulations of Hebbian rules. *Neural Computation*, 2:169-182, 1990.

[25] J. Rubner and K. Schulten. Development of feature detectors by self-organization. *Biological Cybernetics*, 62:193-199, 1990.

[26] S. A. Teukolsky W. H. Press, B. P. Flannery and W. T. Vetterling. *Numerical Recipes*, chapter 11. Cambridge University Press, 1989.

[27] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[28] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149-154, 1964.

[29] E. M. Johansson, F. U. Dowla, and D. M. Goodman. Backpropagation learning for multi-layer feed-forward neural networks using the conjugate gradient method. *IEEE Trans. on Neural Networks*, 1991.

[30] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, to be published.

[31] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[32] M. D. Garris, R. A. Wilkinson, and C .L. Wilson. Methods for enhancing neural network handwritten character recognition. In *International Joint Conference on Neural Networks*, volume 1, pages 695-700. NIST Gaithersburg, Md., 7 1991. IEEE and International Neural Network Society.

[33] M. D. Garris, C. L. Wilson, J. L. Blue, G. T. Candela, P. Groither, S. Janet, and R. A. Wilkinson. Massivelly parallel implementation of character recognition systems. In *Proceedings February 1992*. SPIE, National Institute of Standards and Technology Gaithersburg, Md., 1992.

[34] C. L. Wilson and M. D. Garris. Handprinted character database. *National Institute of Standards and Technology*, Special Database 1, **HWDB**, April 18, 1990.

[35] *Digital Compression and Coding of Continuous-Tone Still Images*. WG10 JPEG, committee draft ISO/IEC CD 10198-1., Washington, D. C., March 3, 1991.

[36] Standard Reference Material. *Reflection Step Table 2601*. National Institute of Standards and Technology, 1971.