ARTIFICIAL NEURAL NETWORKS FOR IMAGE RECOGNITION: A STUDY OF FEATURE EXTRACTION METHODS AND AN IMPLEMENTATION FOR HANDWRITTEN CHARACTER RECOGNITION

by

Deshendran Moodley

Submitted in fulfilment of the academic
requirements for the degree of

Master of Science

in the

Department of Computer Science and Information Systems
University of Natal

Pietermaritzburg

1996

ABSTRACT

The use of computers for digital image recognition has become quite widespread. Applications include face recognition, handwriting interpretation and fingerprint analysis. A feature vector whose dimension is much lower than the original image data is used to represent the image. This removes redundancy from the data and drastically cuts the computational cost of the classification stage. The most important criterion for the extracted features is that they must retain as much of the discriminatory information present in the original data. Feature extraction methods which have been used with neural networks are moment invariants, Zernike moments, Fourier descriptors, Gabor filters and wavelets. These together with the Neocognitron which incorporates feature extraction within a neural network architecture are described and two methods, Zernike moments and the Neocognitron are chosen to illustrate the role of feature extraction in image recognition.

PREFACE

The experimental work described in this dissertation was carried out in the Department of Computer Science and Information Systems, University of Natal, Pietermaritzburg, from January 1995 to January 1996, under the supervision of Professor Vevek Ram and Professor Linda M. Haines.

These studies represent original work by the author and have not otherwise been submitted in any form for any degree or diploma to any university. Where use has been made of the work of others it is duly acknowledged in the text.

ACKNOWLEDGEMENTS

Many people have been supportive with their contributions to this work. I would like to thank my supervisor Professor Ram and co-supervisor Professor Linda Haines for their patience and guidance in the preparation and presentation of this thesis.

I would also like to express my gratitude to my family and friends without whom this thesis would not have been timeously completed.

I acknowledge the Foundation for Research and Development (FRD) for their financial support.

TABLE OF CONTENTS

Page

1 INTRODUCTION 1
1.1 DIGITAL IMAGE RECOGNITION 1
1.2 ARTIFICIAL NEURAL NETWORKS 3
1.3 THE FEATURE EXTRACTION STAGE 4
1.4 THE RESEARCH PROBLEM AND OBJECTIVES 5
1.5 SCOPE OF THE RESEARCH AND DELIMITATIONS 6
1.6 IMPORTANCE OF THE RESEARCH 6
1.7 RESEARCH METHODOLOGY 6

2 ARTIFICIAL NEURAL NETWORKS FOR IMAGE RECOGNITION 7
2.1 ORIGIN 7
2.2 EARLY MODELS 8
2.2.1 The McCulloch and Pitts model 8
2.2.2 The perceptron 9
2.3 THE MULTILAYERED PERCEPTRON 12
2.3.1 The processing element 12
2.3.2 Learning by backpropagation 14
2.3.3 Improvements 17
2.3.4 Classification of the MLP 19
2.4 ALTERNATIVE NEURAL NETWORK MODELS 19
2.4.1 The Neocognitron 19
2.4.2 Hopfield networks and associative memory 19
2.4.3 Adaptive resonance theory (ART) 20
2.5 SUMMARY 20

3 FEATURE EXTRACTION 21
3.1 INTRODUCTION 21
3.1.1 Need for feature extraction 21
3.1.2 Properties of feature extraction methods 23
3.2 MOMENT INVARIANTS 24
3.3 ZERNIKE MOMENTS 27
3.4 FOURIER DESCRIPTORS 33
3.4.1 The Fourier transform 33
3.4.2 Invariant feature extraction 34
3.5 GABOR FILTERS 36
3.5.1 Orientation and spatial frequency of texture 36
3.5.2 Biological motivation 39
3.5.3 Definition 39
3.5.4 Implementation of Gabor filters 43
3.5.5 Examples 44
3.5.6 Characteristics 46
3.6 THE MULTIRESOLUTION WAVELET REPRESENTATION 47

3.7 THE NEOCOGNITRON 50
3.7.1 Structure 50
3.7.2 Operation 51
3.7.3 Implementation 56
3.7.4 Training 57
3.7.5 Biological motivation 59
3.8 SUMMARY 60

4 IMPLEMENTATION 62
4.1 GENERAL REQUIREMENTS DEFINITION FOR A CHARACTER RECOGNITION SYSTEM 65
4.1.1 Functional requirements 65
4.1.2 Database requirements 65
4.1.3 Non-functional requirements 65
4.2 DATABASE 66
4.3 EXPERIMENT 1: THE MULTILAYERED PERCEPTRON 67
4.3.1 Software 68
4.3.2 Normalisation 69
4.3.3 Partitioning of the database 71
4.3.4 Training and testing the MLP 73
4.4 EXPERIMENT II: ZERNIKE MOMENTS 78
4.4.1 System design and implementation 79
4.4.2 Training and testing 81
4.5 EXPERIMENT III: THE NEOCOGNITRON 86
4.5.1 Design and implementation 86
4.5.2 Training and testing 88
4.6 SUMMARY OF RESULTS 89

5 EVALUATION AND CONCLUSION 91
5.1 THE MULTILAYERED PERCEPTION 93
5.1.1 Strengths 93
5.1.2 Weaknesses and limitations 94
5.2 ZERNIKE MOMENTS 94
5.2.1 Strengths 94
5.2.2 Weaknesses 95
5.3 THE NEOCOGNITRON 95
5.3.1 Strengths 95
5.3.2 Weaknesses 96
5.4 FUTURE RESEARCH 97
5.5 CONCLUSION 98

REFERENCES 101

APPENDIX I 107

APPENDIX II 110

LIST OF FIGURES

Figure 1.1.1 - Producing a digital image (bottom row) from a photograph (top row - left). 2

Figure 1.3.1 - Broad structure of an image recognition system. 5

Figure 2.1.1 A biological neuron. 8

Figure 2.2.1 - Inclusive OR operation performed by a 'McCulloch-Pitts' neuron.............. 9

Figure 2.2.2 - A human neuron and a perceptron. 9

Figure 2.2.3 - The XOR problem. 11

Figure 2.3.1 - The multilayered perceptron and a processing element. 13

Figure 2.3.2 - The sigmoid function. 14

Figure 3.3.1 - θ and ρ for a pixel (x,y). 28

Figure 3.3.2 - Binary image represented as a matrix. 30

Figure 3.3.3 - Binary image mapped to the unit circle. 30

Figure 3.3.4 - Five instances of the character b (a) before and (b) after standardization. ...32

Figure 3.3.5 - The letter b rotated through angles of 90°, 180°, 270°............................. 32

Figure 3.4.1 - Radial and angular bins in the frequency domain (Kulkarni, 1994)............... 35

Figure 3.5.1 - Image illustrating importance of local orientation in images......................... 36

Figure 3.5.2 - Two textures with the same orientation, but different frequencies. 37

Figure 3.5.3 - Two textures with the same orientation but different frequencies, which can not be distinguished easily in the spatial domain. 38

Figure 3.5.4 - Ccomparison of the receptive field profile of simple cells in the cat visual cortex to best fitting two dimensional Gabor functions. The bottom row shows the residual errors which are considered to be random (from Daugman & Kammen, 1987). 40

Figure 3.5.5 - An example of a two dimensional Gabor function showing orientation selectivity in the spatial domain (left) and frequency selectivity in the frequency domain ,the Fourier transform (right) from Daugman & Kammen,1987. 40

Figure 3.5.6 - Log polar distribution of two dimensional Gabor filters (adapted from Lisboa 1992). 42

Figure 3.5.7 - Segmentation of different orientations using two Gabor filter, (a) original image; (b) amplitude of Gabor filter (F, B,θ) =(50, 0.7, 42^0); (c) amplitude of Gabor filter (F, B,θ) =(50, 0.7, -43^0) adapted from Bovik et al. (1990). 45

Figure 3.5.8 - (a) original image; (b) amplitude of Gabor filter (F, B,θ) =(40,1, 45^0); (c) postfiltered image (adapted from Bovik et al. 1990). 45

Figure 3.5.9. - (a) original image; (b) amplitude of Gabor filter (F, B,θ) =(40,1, 89^0); (c) amplitude of Gabor filter (F, B,θ) =(40,1.3, 89^0); (d) segmentation (adapted from Bovik et al. 1990). 46

Figure 3.6.1 - An image (a) and its multiresolution representation (b) (adapted from Mallat, 1989). 48

Figure 3.7.1 - Three dimensional structure of the Neocognitron. 50

Figure 3.7.2 - A layer in the Neocognitron containing sub layers of simple cell planes and complex cell planes. 51

Figure 3.7.3 - Structure of a Neocognitron for classification of handwritten numerals (Fukushima, 1983). 52

Figure 3.7.4 - Example of a 3x3 feature or mask to which cells in a simple plane respond. 52

Figure 3.7.5 - Input neighbourhoods of complex cells. 53

Figure 3.7.6 - Illustration of the operation of the Neocognitron. 55

Figure 3.7.7 - The inhibitory and excitatory inputs of a simple cell. 56

Figure 4.1 - The components of a handwriting recognition system 62

Figure 4.2.1 - An example of an handwritten digit from the fl3 database in 32 x 32 text format. 67

Figure 4.3.1 - Experiment 1 - handwritten digit recognition with no feature extraction. ... 68

Figure 4.3.2 - 19 x 19 normalized image of the digit '0' in figure 4.2.1. 70

Figure 4.3.3 - The normalization process used to scale down the 32x32 pixel images from the *fl3* database to 19x19 pixels. 70

Figure 4.3.4 - 8 normalized 19 x 19 images from the fl3 database. Notice the segmentation error in the first image and the high degree of distortion evident in the other images. 72

Figure 4.3.5 - Learning curve for the MLP on fl3 data set: number of correctly learnt facts (y axis) versus number of runs (x axis) from table 4.1. 75

Figure 4.3.6 - Characters from the fl3 database which vary very little in size, shape and orientation. 76

Figure 4.3.7 - Handwritten digits from two different writers. 76

Figure 4.3.8 - Data set II containing normal as well as rotated digits 77

Figure 4.3.9 - Learning curve for the MLP on data set II, which contains the rotated digits: number of correctly learnt facts (y axis) versus number of runs (x axis) from table 4.2. 78

Figure 4.4.1 - Structure of an image recognition system that contains a feature extraction stage. 78

Figure 4.4.2 - General breakdown of feature extraction method of Zernike moments. 79

Figure 4.4.3 - Structure of program used to calculate the magnitude of the Zernike moments, $|A_{nm}|$. 80

Figure 4.4.4 - Coefficients of 47 Zernike moments of digits '0' - '9' from figure 4.3.7 on page 75. 83

Figure 4.4.5 - Learning curve for the MLP on the Zernike moments of data set II which contained the rotated digits: number of correctly learnt facts (y axis) versus number of runs (x axis) from table 4.4 85

Figure 4.5.1 - Structure of an image recognition system which uses the Neocognitron. ... 86

Figure 4.5.2 - 10 images used to train the Neocognitron for data set I. 88

Figure 4.5.3 - 9 images used to train the Neocognitron for data set II. 88

LIST OF TABLES

Table 2.1 - Similarities between the perceptron and the biological neuron........................ 10

Table 3.1 - Analogy of moments with rigid bodies (adapted from Schalkoff, 1989). 24

Table 3.2. - Seven moment invariants derived from central moments (Jain, 1989; Schalkoff, 1989). 25

Table 3.3. - The magnitudes of 4 Zernike moments for the rotated images in figure 3.3.5. 32

Table 3.4 - Summary of feature extraction methods. 61

Table 4.1 - Number of facts learnt at each iteration during training of the MLP on the fl3 data set. 74

Table 4.2 - Number of images learnt at each iteration during training of the MLP on data set containing rotated digits. 77

Table 4.3 - The 47 Zernike moments, $A_{nm,}$ used to represent an image. 80

Table 4.4 - Number of images learnt at each iteration using the Zernike moments of data set II, which contained the rotated digits. 84

Table 4.5. - Structure of the Neocognitron used in the experiment. 87

Table 4.6 - Performance of the three experiments on data set I. 89

Table 4.7 - Performance of the three experiments on data set II. 89

1. INTRODUCTION

Vision is the process whereby energy from the three-dimensional world is converted to two-dimensional entities called images. These images are then processed to either form some conclusions or to invoke some actions. For a long time visual scenes could only be captured through photography. However recent technological advances allow visual scenes to be captured, manipulated and evaluated electronically with computers and this has resulted in the development of the area of computer vision. Computer vision has in turn led to the automation of a variety of tasks which include industrial inspection, analysis of aerial maps, vehicle guidance, image compression, which allows for the transmission of high resolution television pictures, and image enhancement, which can be used for the enhancement of X-rays or photographs. One important application in computer vision is digital image recognition. A digital image is a two dimensional entity which is acquired by the use of a television camera or digital scanner. The use of computers for digital image recognition has become quite widespread. Applications include human face recognition (Bouattour et al. 1992) , handwritten character recognition (Khotanzad & Lu , 1991) and meteor trail classification (Fraser et al. , 1992), amongst others.

1.1 DIGITAL IMAGE RECOGNITION

A digital image is a two-dimensional computer representation of a visual scene. The digital image consists of a two dimensional array of numbers with the rows and columns of the array corresponding to the vertical and horizontal co-ordinates of the image. To produce a digital representation of a black and white photograph, samples of brightness are taken at regular intervals from the photograph with the size of the intervals depending on the size of the array. These brightness values will range from 0 to 255 with 0 representing white, 255 representing black and the integer values in between representing different shades of grey. For example the value 20 will represent a lighter shade of grey than 130. This system is known as the grey scale system. Other systems incorporate colour information as well as brightness information.

As an example, suppose that a 10 x 10 array was used to represent the 10 cm x 10 cm black and white photograph depicting an eagle shown in the top row of figure 1.1.1. Then each element in the array will represent the average brightness of a 1 cm by 1 cm block in the corresponding position on the photograph. This implies that the element in row 1 and column 1 of the array will be the average brightness of a 1cm by 1cm block on the top left hand corner of the photograph. The 10 x 10 array will then form a digital representation of the photograph as shown in the bottom row of figure 1.1.1. Each element in the array represents a picture element, normally abbreviated to "pixel", in the digital image. Since the pixels can only take on different shades of grey, the digital image is known as a grey scale digital image.

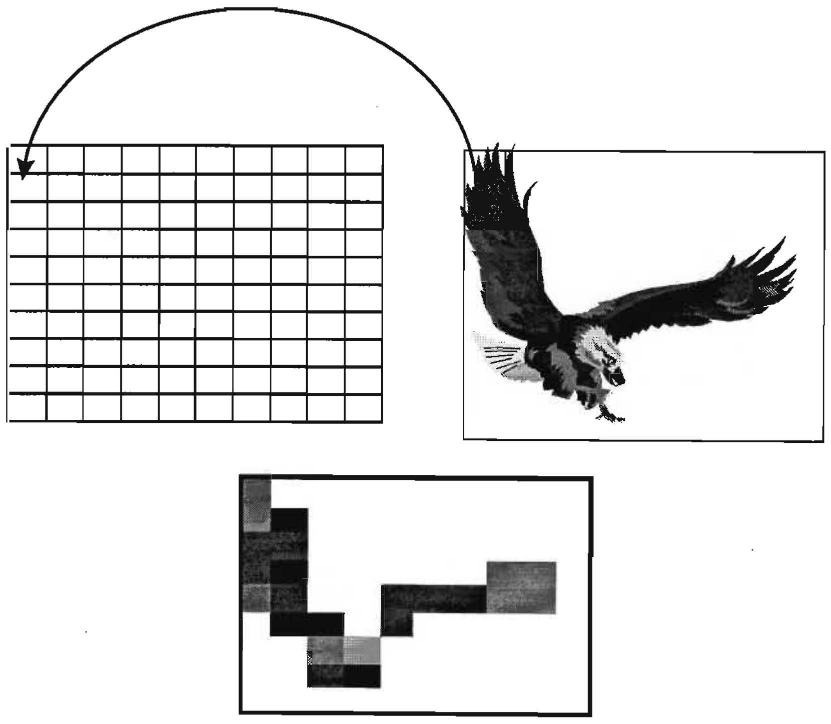


Figure 1.1.1 - Producing a digital image (bottom row) from a photograph (top row - left).

Digital image recognition involves the mapping of digital images onto specific predefined classes. If these digital images depict birds, then the output classes could be 'eagle', 'swallow' and 'falcon'. Digital image recognition falls within the general area of pattern recognition which involves the mapping of a feature space to output classes. For digital image recognition, the feature space Z^{mxn} consists of the brightness values of an mxn two dimensional array representing a grey scale digital image. The output space Z^s represents the *s* unique output classes for the feature vector, Z^{mxn}, and comprises vectors of the form (c_1, c_2, .. , c_s) with c_i taking on values of 0 or 1. In particular only one element in such a vector takes on the value 1 with the remaining elements being 0, that is, if the feature vector is classified into class c_i, then only $c_i = 1$ and the others will be 0.

1.2 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks and more specifically the multilayered perceptron (MLP) have been used extensively for image recognition applications. These include human face recognition (Bouattour et al. 1992) , human face segmentation (Viennet & Soulie, 1992), radar data classification (Pernot & Vallet, 1991), handwritten character recognition (Khotanzad and Lu , 1991) and meteor trail classification (Fraser et al., 1992). In these applications artificial neural networks have been shown to be superior, in terms of accuracy and sensitivity to noise, to the traditional classifiers such as nearest neighbour (NN) and minimum mean distance (MMD) for image recognition tasks.

The multilayered perceptron consists of three layers, the input layer, the hidden layer and the output layer with each layer containing processing elements, called neurons, which are connected to the neurons in the previous layer. The classification process involves the presentation of a feature vector to the input layer and the producing of a response from a specific neuron in the output layer, corresponding to an output class. If the feature vector consists of a digital image, then each neuron in the input layer will receive input from a single pixel in the digital image. In this instance, the number of neurons in the input layer will equal the number of pixels in the digital image and the number of neurons in the output layer will correspond to the number of output classes.

The use of the multilayered perceptron in digital image recognition has the following limitations:

- Unnecessary processing of redundant information. The inputs to the multilayered perceptron are the pixel values of the digital images which may contain information which is not needed for discrimination between output classes. For example, if the task was to classify fruit into apples, oranges and bananas the information about the size, position or how the fruit is rotated in the image is of no significance and is thus redundant in this application.

- For digital images a large number of neurons are needed in the input layer. However large networks are computationally infeasible, as the time needed for training is related to the size of the network.

1.3 THE FEATURE EXTRACTION STAGE

The above limitations of artificial neural networks can be overcome by the introduction of a feature extraction stage in which the digital image is processed so that only relevant features needed for discrimination between the output classes are extracted. This new feature set will then form the input to the neural network. In this way redundant information will be removed and the size of the feature set reduced. A number of feature extraction methods such as moment invariants (Jain, 1989), Zernike moments (Khotanzad and Hong, 1990), Fourier descriptors (Kulkarni, 1994), Gabor filters (Daugman, 1988) and wavelets (Mallat, 1989) are available in image recognition. Another interesting model, the Neocognitron (Fukushima et al., 1983), combines the two areas of features extraction and neural networks.

Generally feature extraction methods must have the following properties:

- Extracted features must retain much of the discriminant information present in the original data.
- Features should have small intra-class variance, that is, slightly different shapes with similar general characteristics should have similar numerical values.
- Features must also have large inter-class differences, that is, features from different classes should be quite different numerically.

An additional stage, called the pre-processing stage, in which the raw data is manipulated before feature extraction, may also be introduced. This stage is dependent on the nature of the images to be classified. Typical pre-processing techniques include noise removal and contrast enhancement (Schalkoff, 1989). For example, if the system can only accept 19 x 19 pixel images and the digital images used for classification are not of this size then the images are scaled to a 19 x 19 pixel image in the pre-processing stage.

Figure 1.3.1 shows the broad structure of the image recognition system described above.

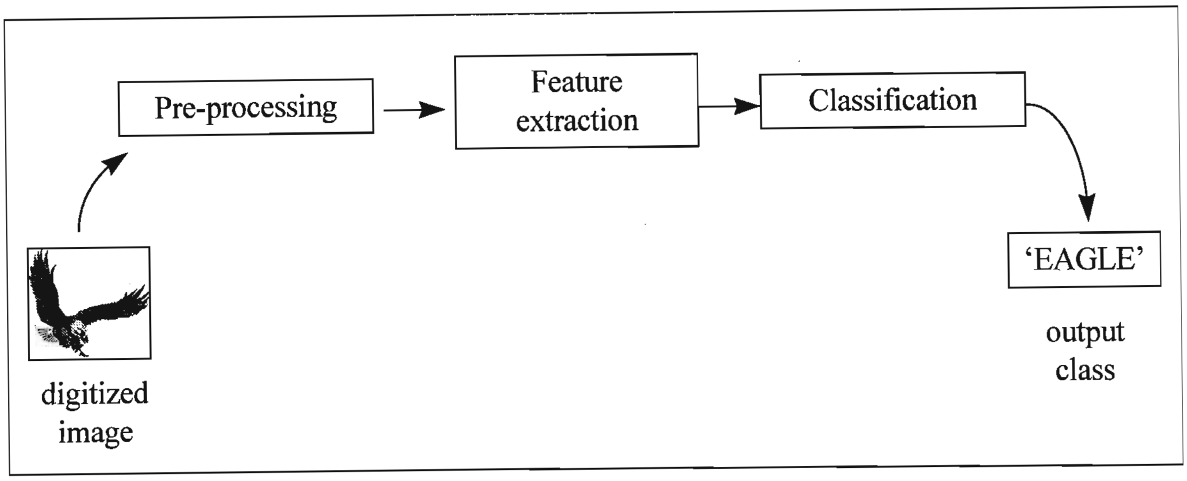


Figure 1.3.1 - Broad structure of an image recognition system.

1.4 THE RESEARCH PROBLEM AND OBJECTIVES:

The broad aim of the research is to investigate the role of feature extraction in a neural network approach to image recognition.

The objectives of the research are:

1. To investigate existing feature extraction methods, namely, moment invariants, Zernike moments, Fourier descriptors, Gabor filters, wavelets and the Neocognitron for image recognition.

2. To identify suitable feature extraction methods from those in (1) for two dimensional shape recognition applications.

3. To implement experimentally the two most suitable methods identified in (2) and to evaluate their relative effectiveness on handwritten digit recognition.

1.5 SCOPE OF THE RESEARCH AND DELIMITATIONS

The main focus of the research is on feature extraction methods which have been used together with neural networks for applications in digital image recognition. The discussion on neural networks is confined mostly to the multilayered perceptron and its application to digital image recognition, more specifically the recognition of two dimensional grey scale and binary (black and white) images.

1.6 IMPORTANCE OF THE RESEARCH

Feature extraction, used appropriately, can improve the overall efficiency of a digital image recognition system. However, the choice of feature extraction method to use is difficult and depends usually on the nature of the application. The intention of the research is to simplify this decision.

1.7 RESEARCH METHODOLOGY

The area of feed forward neural networks with application to image recognition will be described. Thereafter various feature extraction methods which have been used together with feedforward networks will be discussed and evaluated.

A practical digital image recognition application will be chosen to illustrate the role of feature extraction in digital image recognition. Appropriate feature extraction methods will be identified and implemented experimentally for the application. A data set will be constructed and used to test the experimental implementations. These results will be discussed and used to evaluate the feature extraction methods.

This chapter has provided an overview of the content of the thesis. In chapter two an overview of neural networks and their application to digital image recognition is given. In chapter three various feature extraction methods are described and evaluated. In chapter four, three experimental implementations for handwritten character recognition are described and tested. Chapter five contains the conclusions of the research.

2. ARTIFICIAL NEURAL NETWORKS FOR IMAGE RECOGNITION

Artificial neural networks have been used extensively for digital image recognition applications. These include handwritten character recognition (Rajavelu et al., 1989, Hinton et al., 1992, English et al., 1993, Hussain & Kabuka, 1994) and face recognition (Wurtz, 1994). In this chapter, artificial neural networks are traced back to their origins and a widely used network, the multilayered perceptron (MLP), is described.

2.1 ORIGIN

Biological systems, such as the mammalian brain, are able to execute perceptive tasks faster and more effectively than the most powerful computers. These tasks include object recognition, room navigation, reading, hearing and walking. The human nervous system consists of hundreds of billions of interconnected cells called neurons. Each neuron is able to receive, process and transmit electrochemical signals through the neural pathways in the brain. The neuron (figure 2.1.1) consists of the cell body, synapse, axon and dendrites (Stephen, 1984). The dendrites receive signals from other neurons at connection points called synapses. The synapses attach "weights" to incoming signals causing them to have a positive or excitatory effect, or a negative or inhibitory effect on the neuron. The cell body sums the signals provided by the dendrites and if this sum exceeds a specified threshold value, the cell fires, sending a signal down the axon to other neurons. This signal represents the output of the neuron. A highly excited neuron tends to send out a signal and an inhibited one does not.

The improved understanding of the functioning of the neuron and the pattern of its interconnections has allowed researchers to abstract the main characteristics of biological neural networks to develop artificial neural networks which simulate their biological counterparts.

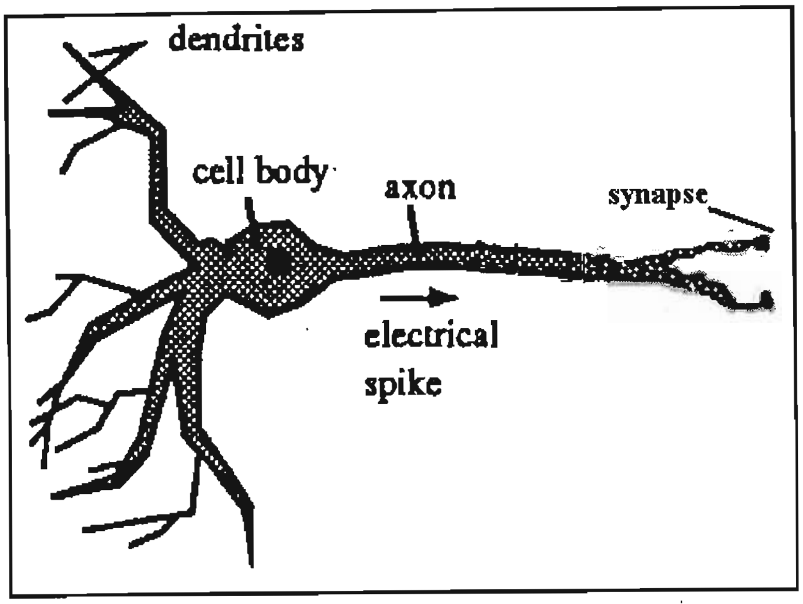


Figure 2.1.1 A biological neuron.

2.2 EARLY MODELS

Most modern models of the neuron are based on the early models of McCulloch and Pitts (1943), Donald Hebb (1949) and Rosenblatt (1958).

2.2.1 The McCulloch and Pitts model

The results of McCulloch and Pitts (1943) formed the foundation for research into artificial neural networks. They showed that networks composed of binary valued neurons, that is, neurons with two states and a fixed threshold value, are capable of performing arbitrary logical operations. The neuron receives inputs from excitatory and inhibitory synapses. The excitatory synapses have the same weights, while the inhibitory synapses have an overriding action, in the sense that the neuron cannot fire if the inhibitory neuron is active. The operation is as follows: if no inhibitory neuron is active then the neuron sums its excitatory synapses, and if this sum meets or exceeds the threshold value then the neuron fires.

The inclusive OR operation is an example of a logical operation that can be performed by the 'McCulloch-Pitts' neuron and figure 2.2.1 shows such a neuron.

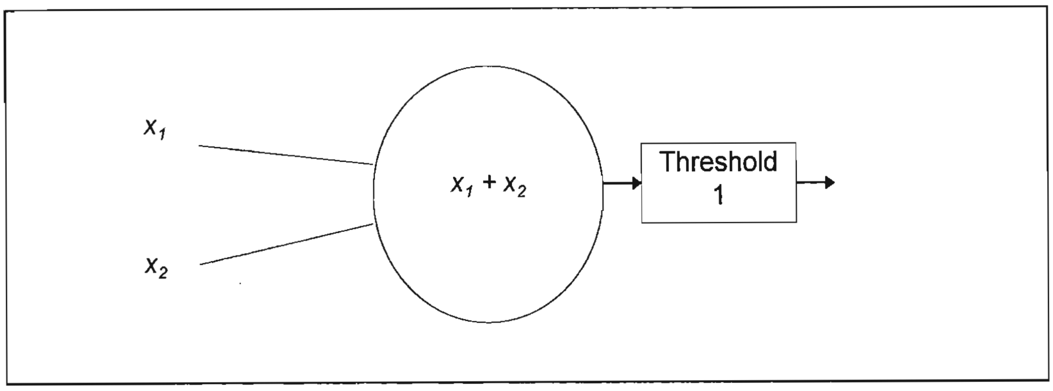


Figure 2.2.1 - Inclusive OR operation performed by a 'McCulloch-Pitts' neuron.

The neuron in figure 2.2.1 has two excitatory inputs, x_1 and x_2, and a threshold value of 1. If any of the excitatory inputs is 1, then $x_1 + x_2 \geq 1$, and the neuron will fire. Similarly the AND operation can be performed if the neuron in fig. 2.2.1 has a threshold of two.

2.2.2 The perceptron

The perceptron (Rosenblatt, 1958) is shown alongside a biological neuron in figure 2.2.2. Some similarities between the perceptron and biological neurons are illustrated in Table 2.1.

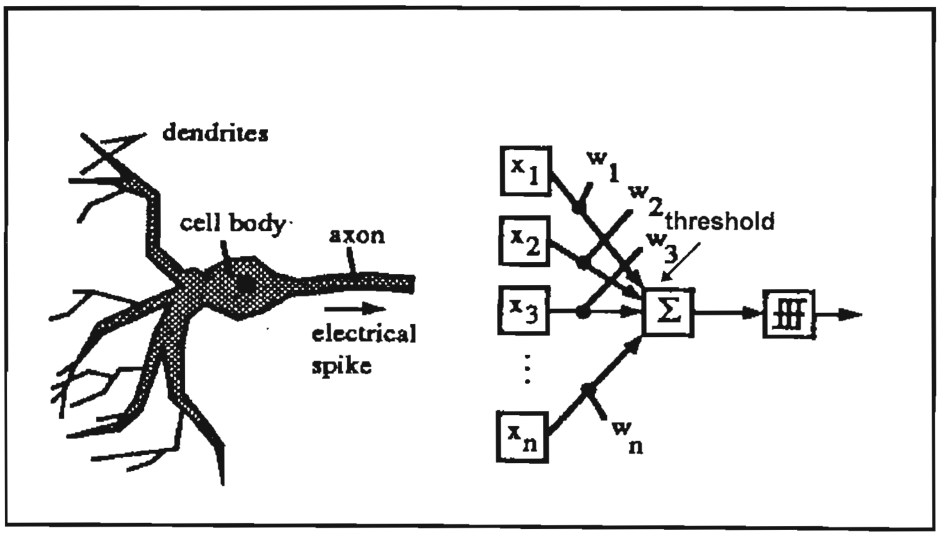


Figure 2.2.2 - A human neuron and a perceptron.

Table 2.1 - Similarities between the perceptron and the biological neuron.

Characteristic	**Perceptron**	**Biological neuron**
Input connections	inputs labelled x_i	dendrites
Processing unit	weighted sum process	cell body
Output connection	has output connection	axon
Variable connections	variable weights of input connections	signal varied at synapses

The perceptron uses the concept of a threshold neuron. A threshold neuron is modelled by taking a weighted sum of its inputs and sending an output of 1 if the sum is greater than some adjustable threshold value and 0 otherwise.

The inputs (x_1 , x_2 , .. , x_n) and corresponding connection weights (w_1 , w_2 , .. , w_n) are usually negative and positive real values and form the weighted sum $\sum_{i=1}^{n} w_i x_i$. If $\sum_{i=1}^{n} w_i x_i > T$, where T is the threshold value, then the neuron fires and if $\sum_{i=1}^{n} w_i x_i \leq T$ then the neuron does not fire. If an input x_i causes the perceptron to fire then the weight w_i will be positive and if the input x_i inhibits the perceptron then its corresponding weight w_i is negative. Each neuron can be considered to act as a function, mapping its domain of the x_i's, to its range {0, 1}. The actual mapping will be dependent on the values of the weights w_i's and the threshold.

Learning

Learning involves modifying the values of the weights to get desired outputs for different input values. Artificial neural networks modify their behaviour in response to their environment. Shown a set of inputs, and in some instances desired outputs, these networks self adjust to produce consistent responses to similar inputs. This property is known as learning. Donald Hebb (1949) was the first to propose a mechanism by which learning could occur. Hebb's rule states that *whenever two neurons are excited at the same time, the connection between them should be strengthened.* The perceptron uses a variation of Hebb's rule for learning. The network begins in a random *initial* state, i.e. all the weights are randomised, and finds a solution state that is the desired mapping of inputs to outputs.

The search space is all the assignments of real values to the weights of the perceptron. An iterative search method is often used. This method is based on the following rule :

If the neuron fires when it should not fire, make each w_i of the input connections smaller by an amount proportional to x_i. If the perceptron fails to fire when it should fire, make each w_i larger by a similar amount.

An in-depth study of the Perceptron was carried out by Minsky & Pappert (1969) who showed that a large class of problems, those that are not linearly separable, cannot be solved by the perceptron. A simple example of a problem that is not linearly separable is the exclusive OR function shown in figure 2.2.3. No linear function of the form $w_1x_1 + w_2x_2$ exists which can separate the black and white points and this problem cannot therefore be solved by the perceptron. The linear seperability problem was eventually overcome by the introduction of a hidden layer into the perceptron and this led to the development of the multilayered perceptron.

x_1	x_2	Desired output x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

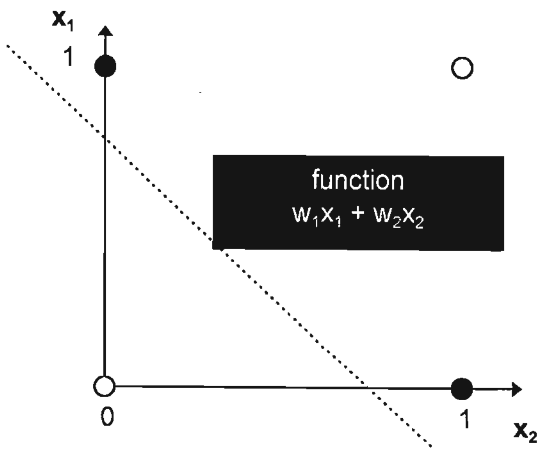


Figure 2.2.3 - The XOR problem.

2.3 THE MULTILAYERED PERCEPTRON

The multilayered perceptron (MLP), illustrated in figure 2.3.1, consists of layers of artificial neurons or processing elements. These neurons have several inputs which are processed to produce the outputs and neurons working together form a layer. The network normally consists of three layers the input layer , the hidden layer and the output layer and it can be shown that the introduction of the hidden layer is primarily responsible for the increase in the computational power of the network. (Hanson & Burr, 1991). There are no connections within layers, and outputs from each layer can only form inputs to the next layer. Thus the network is termed as a feed forward network.

2.3.1 The processing element

The processing element is depicted in figure 2.3.1. Let $\mathbf{x} = (x_1 , x_2 , \dots , x_n)$ represent the inputs applied to the neuron and $\mathbf{w} = (w_1 , w_2 , .. , w_n)$ the corresponding weights. Then the net input, which is the weighted sum of the neuron, at a particular time t is defined as :

$$net(t) = \mathbf{w}^T \mathbf{x} + B,$$

where B is a bias term that is introduced so that the neuron can still fire even if all its inputs are zero. The net input is then processed by an activation function to produce the neuron's output signal O(t). The preferred function is the sigmoid function shown in figure 2.3.2 and defined as

$$f(z) = 1 / (1 + e^{-z}),$$

where z is the net input $net(t)$.

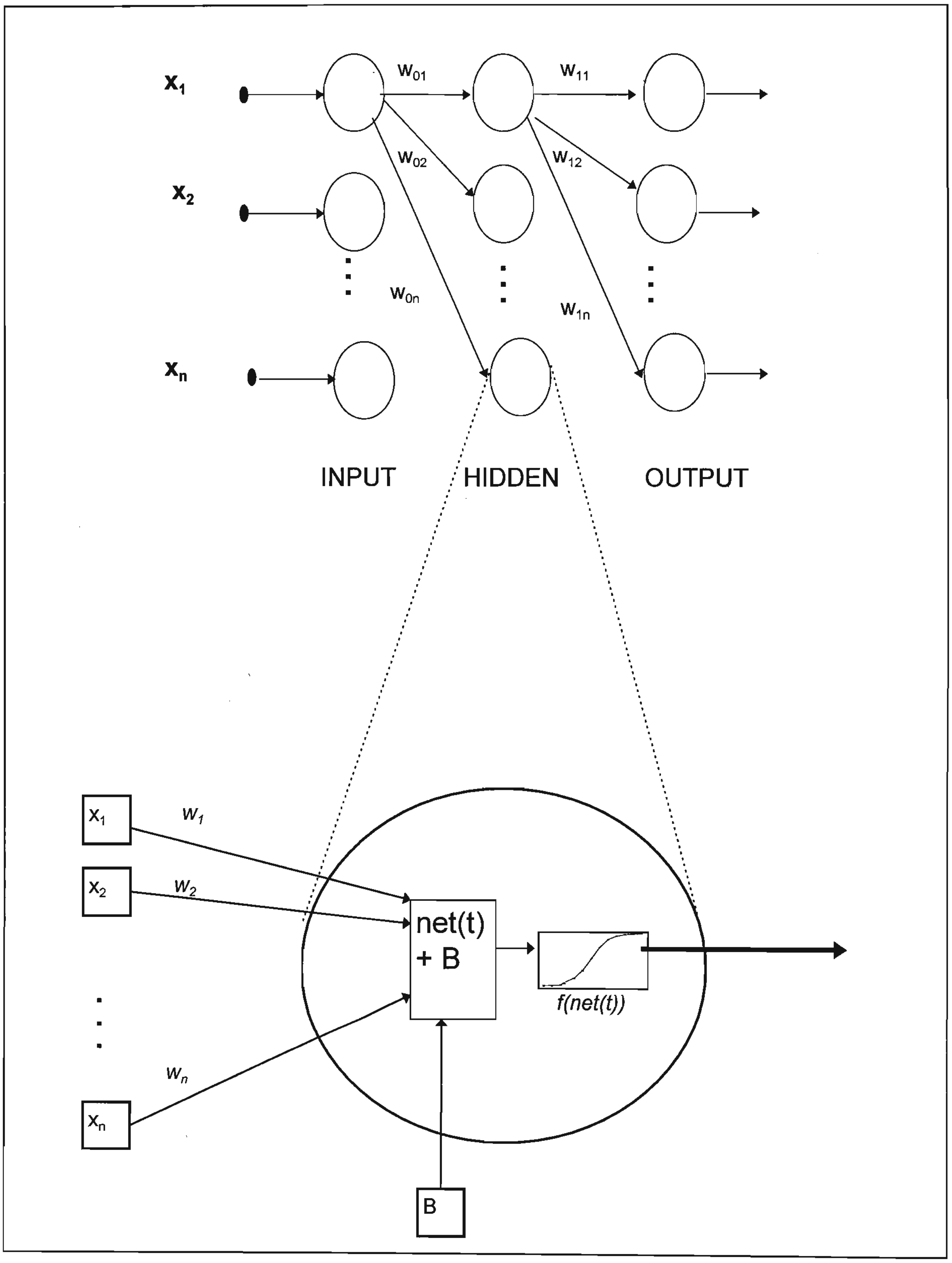


Figure 2.3.1 - The multilayered perceptron and a processing element.

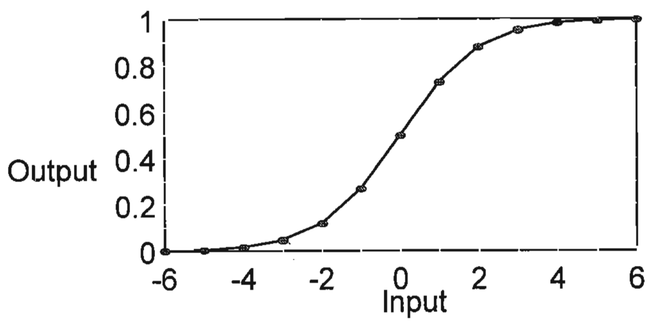


Figure 2.3.2 - The sigmoid function.

Learning in the MLP, which is also known as training, is performed by an algorithm known as the backpropagation algorithm.

2.3.2 Learning by backpropagation

The backpropagation algorithm is used to train the multilayered perceptron (Rumelhart et al. , 1986). The algorithm requires a teacher to supply training data, which consists of input and output patterns.

The weights in the network are initialised to some random values. Input and desired output vectors are then presented to the network. An error E is calculated indicating the difference between the network output and the desired output. This error is propagated back through the network and the weights in each layer are modified so that the error is decreased during the next iteration. This process is repeated until the error E is small enough. The generalised delta rule (Rumelhart et al. 1986), which is based on gradient descent, is used to decrease the error during each iteration. The mathematics of the generalised delta rule are described below.

The generalised delta rule

The application of the generalised delta rule involves two phases. During the first phase the input is presented and propagated through the network to compute the output value for each output neuron. This output is then compared to the target outputs resulting in an error signal for each output neuron. The second phase involves a backward pass through the network, during which the error is propagated back through each neuron in the network and the weights are adjusted accordingly.

The global error E is defined as the squared difference between the output vector and the desired or target output vector and is given by

$$E = \frac{1}{2}\sum_{k=1}^{s}(t_k - o_k)^2, \tag{2.1}$$

where t_k is the target output of the *k*th neuron in the output layer and o_k is the actual output of the *k*th neuron in the output layer. The output o_k is given by

$$o_k = f_k(\sum_{j=1}^{h} w_{jk}^{(HO)} o_j^{(H)}), \tag{2.2}$$

where w_{jk} is the weight of the connection from neuron *j* in the hidden layer to neuron *k* in the output layer, f_k is the activation function of neuron *k* in the output layer and $o_j^{(H)}$ is the output of the *j*th neuron in the hidden layer given by

$$o_j^{(H)} = f_j(\sum_{i=1}^{p} w_{ij}^{(IH)} x_i), \tag{2.3}$$

where w_{ij} is the weight of the connection from neuron *i* in the input layer to neuron *j* in the hidden layer, f_j is the activation function of neuron *j* in the hidden layer and x_i is the input of neuron i in the input layer. Alternatively equations (2.2) and (2.3) can be written as

$$o_k = f_k(\eta_k^{(O)}), \tag{2.4}$$

where $\eta_k^{(O)}$ is the net input to neuron *k* in the output layer and

$$o_j^{(H)} = f_j(\eta_j^{(H)}), \tag{2.5}$$

where $\eta_j^{(H)}$ is the net input to neuron *j* in the hidden layer.

The amount by which the weights of each neuron has to be adjusted is

$$\Delta w_{mn} = l\_coeff \frac{\partial E}{\partial w_{mn}} \tag{2.6}$$

with w_{mn} being the weight of the connection from neuron *m* to neuron *n*. If neuron *n* is in the output layer, then using the chain rule

$$\begin{aligned}\frac{\partial E}{\partial w_{jk}^{(HO)}} &= \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial \eta_k} \cdot \frac{\partial \eta_k}{\partial w_{jk}^{(HO)}} \\ &= -(t_k - o_k).f_k'(\eta_k).o_j^{(H)}; \end{aligned} \tag{2.7}$$

otherwise, if neuron *n* is in the hidden layer then using the chain rule

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}^{(IH)}} &= \sum_{k=1}^{s} \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial \eta_k} \cdot \frac{\partial \eta_k}{\partial \eta_j} \cdot \frac{\partial \eta_k}{\partial w_{ij}^{(IH)}} \\ &= \sum_{k=1}^{s} -(t_k - o_k).f_k'(\eta_k^{(O)}).w_{jk}^{(HO)}.f_j'(\eta_j^{(H)}).x_i \\ &= \sum_{k=1}^{s} e_k^{(HO)}.w_{jk}^{(H)}.f_j'(\eta_j^{(H)}).x_i \,. \end{aligned} \tag{2.8}$$

Equations (2.1), (2.7) and (2.8) illustrate the generalised delta rule. The network learns by making corrections to the weights, based on the error of the output from equation (2.1). which is propagated back through the network during training. As training progresses the global error in (2.1) is minimised by changing the weights of each neuron using equation 2.6. To adjust the weights of neurons in the output layer equation (2.7) is used while equation (2.8) is used to adjust the weights of the neurons in the hidden layer. Thus, the generalised delta rule provides a mathematical explanation for the dynamics of learning and is consistent and reliable in application.

2.3.3 Improvements

Rumelhart et al. (1986), Pao (1989) and Kung et al. (1991) showed that there were improvements that could be made to the backpropagation algorithm. These are discussed briefly here.

Direction of updating weights

The standard approach, captured in equation (2.6), updates the weights in the opposite gradient of the error function. A more sophisticated approach is to add a momentum term which is dependant on the previous change in the weights and to rewrite equation (2.6) as:

$$\Delta w_{ij}(n) = l\_coeff.\frac{\partial E}{\partial w_{ij}} + \alpha\Delta w_{ij}(n-1), \tag{2.9}$$

where α is the momentum constant, $\Delta w_{ij}(n-1)$ represents the change in the weight in the previous iteration and $\Delta w_{ij}(n)$ is the change in the weight for the current iteration.

A variation on the gradient descent approach is the conjugate gradient approach (Hertz, 1991, Kung et al. 1991) where the updating vector is defined by a combination of the gradient and the previous updating vector:

$$\Delta w_{ij}(n) = -\, l\_coeff.\, d(n), \tag{2.10}$$

where d(n) is defined recursively as

$$d(n) = -\frac{\partial E}{\partial w_{ij}}\beta.\, d(n-1) \tag{2.11}$$

and β is chosen so that the new search direction is a compromise between the gradient descent and the previous search direction.

Learning Rate

One of the problems in implementing the backpropagation algorithm is choosing the value of *l_coeff.* A large learning rate results in faster learning but might also result in oscillations in the global error during learning, while a very small learning rate will result in slow learning. One solution is to start with a large *l_coeff* and decrease it gradually so that the initial convergence is rough but fast and gradually the convergence becomes slower but finer. An alternative is to adjust the size of *l_coeff* to best fit each pair of input-output training patterns (Kung et al. , 1991).

Data Presentation

The presentation of data affects the manner in which the weights are updated during training. The entire data set could be presented to the network and a cumulative average error calculated. The weights are then adjusted at the end of this presentation according to this error. This method, known as cumulative backpropagation, minimises the overall or global error function and not each component error function which might increase other component error functions. If the number of elements in the data set, that is, the number of input/output pairs presented during the accumulation, is not too large then the network will converge faster. However it is possible that two errors may cancel each other and cause the network to be stuck in a local minimum (Hertz, 1991). One solution to this problem is to update the weights after each pattern is presented to the network. For this method data elements are presented randomly one at a time from the data set and the weights updated after each presentation. A compromise between the two methods is to present the data in groups which are subsets of the original data set and to update the weights after each group is presented to the network.

Initial weights

If the set of initial weights for the network are chosen to be identical, then all error signals propagated back through the hidden neurons will be identical as they are proportional to the weights. Since the weight changes are dependant on these error signals, the weight updates will also be identical. Thus during each iteration of the learning cycle the weights will be identical and the network may not converge to an optimal solution. This problem is avoided by choosing the initial set of weights to be small random numbers (Rumelhart et al. , 1986).

2.3.4 Classification of the MLP

The multilayered perceptron is classified as a feedforward, supervised, non linear network. Feedforward describes the conduction of signals and the interconnections between neurons, in that the output from each neuron can only form the input to a neuron in the next layer. The network needs a teacher supplying desired output patterns for each input pattern during training and hence supervision is needed during training. Non-linear describes the transfer function that is used, in this case the sigmoid function which is nonlinear. Other neural network architectures incorporate feedback, require no supervision and use nonlinear transfer functions.

2.4 ALTERNATIVE NEURAL NETWORK MODELS

There are currently many different network architectures. A selected few that have been used for image recognition are discussed in this section.

2.4.1 The Neocognitron

The Neocognitron is a three dimensional multilayered feed forward network that simulates the way visual information feeds forward in the cortex of the human brain (Fukushima, 1988). Furthermore the Neocognitron performs feature extraction in the sense that features are extracted at each layer of the network, with simple local features being extracted in the initial layers and complex global features being formed in layers close to the output layer. The Neocognitron is particularly suitable for handwritten character recognition (Fukushima, 1988).

2.4.2 Hopfield networks and associative memory

Recurrent networks have feedback from the outputs of the network to their inputs and are capable of forming associative memories, i.e. only a portion of the pattern is supplied as input and the correct pattern is returned as output. Hopfield showed that such neural networks are capable of exhibiting interesting behaviour such as pattern recall, despite incomplete or garbled input (Hopfield, 1982). Hopfield nets are recurrent, associative memory architectures and are capable of reconstructing images from noisy or blurred data (Hertz et al. 1991). Other associative memory models include the Bi-directional Associative Memory (BAM) networks (Wasserman, 1989).

2.4.3 Adaptive resonance theory (ART)

ARTMAP by Carpenter, Grossberg & Reynolds (1991) consists of two adaptive resonance theory modules, ART_a and ART_b , that are capable of self-organising stable recognition categories in response to input patterns. During training ART_a is presented with a stream of input patterns and ART_b is given the correct prediction. Testing involves presenting the remaining input patterns to ART_a and comparing the predictions from ART_b to the correct predictions. ART and ARTMAP has been used for character recognition (Dimitriadis et al., 1993, Hung & Lin, 1993).

2.5 SUMMARY

In this chapter we have examined a simplified model of biological neural networks, the multilayered perceptron, for image recognition. It is clear that research in artificial neural networks focuses on two areas, the modelling of neural networks in order to get a better understanding of the mammalian nervous system and practical applications such as speech recognition, computer vision and other classification tasks. Some researchers are of the opinion that one approach complements the other. The research in this thesis will concentrate on applying neural networks to image recognition but hopefully will also give the reader some insight into biological visual systems.

Even though the MLP has been used extensively for image recognition applications, it still has some serious limitations. These limitations can be overcome by the introduction of a feature extraction stage and this is described in the next chapter.

3. FEATURE EXTRACTION

3.1 INTRODUCTION

Artificial neural networks have been used widely for digital image recognition (Bouattour et al. 1992, Viennet & Soulie, 1992, Waite, 1992, Khotanzad & Lu, 1991, Gouin & Scofield, 1993, Fraser, et al., 1992). The usual approach involves presenting the network with a digital image and producing an output label for the image. For example, if the system is classifying 11 x 11 binary images, the input vector will have 121 elements, having the value 0 or 1, corresponding to the 121 pixels in the image. If the images are handwritten digits for example, then the output layer will contain ten neurons, corresponding to the ten different outputs ('0', '1', '2', .. , '9'). When an image is presented to the network only one of the output neurons will be activated, thus classifying the image.

Another approach is to extract only relevant information from the pixel representation and to use this information as the input to the neural network. This process is known as feature extraction. The actual information extracted from the pixel representation is termed the feature vector.

3.1.1 Need for feature extraction:

Number of neurons in the input layer

Since the feature vector provides input to the network, the number of neurons in the input layer of the network will be determined by the number of features in the feature vector and not by the size of the digital image. For example using the pixel representation a 64 x 64 digital image will require 4 096 input neurons, but using a feature extraction method such as Zernike moments (Khotonzad and Hong, 1990), 47 features will be extracted from the image and only 47 neurons will be needed in the input layer of the network. It is important to note that larger networks generally require more computing resources and take a longer time to train.

Irrelevant information

Suppose the images to be classified are geometric shapes. Then certain information about the shape such as position and size is not required, but using the pixel representation, two images of the same shape but of different sizes will produce different input vectors to the network and this may well complicate the classification task. It is thus desirable to make the network invariant to these properties. Invariancy in image recognition can be incorporated in three different ways:

1. During training:

   The network is trained on all instances of an object i.e. all translations (positions), rotations and dilations (sizes) of the object. As a result the network learns to classify the image even if the object is rotated, scaled or at different positions in the image. This drastically increases the size of the training data set, resulting in an increase in the training time and the size of the network. The number of input neurons corresponds to the number of pixels in the image and additional hidden neurons are needed for the extra training data. This method is not practically feasible.

2. Feature extraction:

   This is the classical feature extraction approach in which invariant features are extracted from the data. The number of input neurons corresponds to the dimension of the feature vector. This approach seems to be the most popular, as the feature extraction methods that are used are existing methods which have been used previously with traditional classifiers in image recognition. Examples of such methods include moment invariants (Jain, 1989), Zernike moments (Khotanzad and Hong, 1990), Fourier descriptors (Kulkarni, 1994), Gabor filters (Daugman, 1988) and wavelet transforms (Mallat, 1989).

3. Within the network:

   It is possible to incorporate invariancy within the network itself. Current models such as the Neocognitron (Fukushima et al., 1983), which is invariant to location, size and distortions of the image, are based on the early vision feedforward pathways in the human brain The network can be considered as performing feature extraction, but not in the usual sense as described in (2) above. In this instance the pixel representation is used as the input to the network and thus the problem with the size of the network for large images remains.

Feature extraction, as discussed in (2) and (3) above, extracts only relevant information from the image and thus simplifies the classification task.

3.1.2 Properties of feature extraction methods:

Ideally the feature extraction stage of image recognition should process an input image so that irrelevant information such as size and location is removed and the dimension of the input vector, which normally consists of the pixel values of the image, is reduced. The resultant feature vector then forms the input to the network. The size of the network is reduced, the complexity introduced in order to respond to shape, scale and position is removed and the network is dedicated to differentiating between the different output classes. This is known as invariant image recognition. The properties to which the system is invariant are determined by the nature of the application.

The most important criterion for the extracted features is that they must retain much of the discriminatory information present in the original data. A good set of features can be used to reconstruct the original image. If the reconstructed image is close enough to the original image, then the feature vector provides a good representation. However, in practice some feature extraction methods produce features which cannot be used to reconstruct the original image e.g. invariant moments (Teh & Chin 1988). For these methods, if the features of similar images yield numerically close values and features of different images yield numerically different values then the features are acceptable.

Another important characteristic of feature sets is the handling of redundancy of information. This occurs when more than one element in the feature vector encodes the same characteristic of the image and results in the elements not being independent of each other. This creates difficulties when trying to reconstruct the image from its features as each element does not make a unique contribution to the original image. However, redundancy increases robustness of the representation in the sense that if an element is corrupted then the characteristics encoded by that element may still be available from another element. Orthogonal feature extraction methods produce features whose elements are totally independent of each other and thus exhibit no redundancy. These methods therefore result in easier reconstruction of the image as each element in the feature vector makes a unique contribution to the image.

Feature extraction methods can be classified by the origin of the method, and in particular whether the method was inspired by the visual structures in the brain or whether it has origins in signal processing. Methods such as Gabor filters (Daugman, 1988) and the Neocognitron (Fukushima et al., 1983) have been inspired by the neural structures in early mammalian vision. Others such as Zernike moments (Khotanzad and Hong, 1990), Fourier descriptors (Kulkarni, 1994) and wavelet transforms (Mallat, 1989) have their origins in signal processing.

An overview of feature extraction methods has been given. Specific feature extraction methods such as moment invariants (Jain, 1989), Zernike moments (Khotanzad and Hong, 1990), Fourier descriptors (Kulkarni, 1994), Gabor filters (Daugman, 1988), wavelet transforms (Mallet, 1989) and the Neocognitron (Fukushima et al., 1983) have been identified, and these will now be described separately below.

3.2 Moment invariants

Let f(x, y) denote the brightness of a pixel at position (x, y) with x denoting the column and y the row of the pixel in a digital image. Then the moments of a digital image f(x, y) are given by :

$$m_{pq} = \sum_{x}\sum_{y} x^p y^q f(x,y) \tag{3.2.1}$$

where p and q refer to the order of the moments. If f(x, y) represents a binary image , i.e. only take on values of 0 or 1, then m_{00} will represent the total number of pixels that contribute to the shape of the object. To make the object translation (position) invariant, the origin is moved to the centroid of the object which is determined by m_{10} and m_{01} and the moments

$$m'_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

where (3.2.2)

$$\bar{x} = \frac{m_{10}}{m_{00}} \text{ and } \bar{y} = \frac{m_{01}}{m_{00}}.$$

are introduced. Scale invariance can be obtained by standardizing the moments of equation 3.2.2 to give the central moments:

$$\mu_{pq} = \frac{m'^{\gamma}_{pq}}{m_{00}}, \qquad (3.2.3)$$

where $\gamma = (p+q)/2 + 1$, which are invariant to both scale and translation. To achieve a better understanding of central moments an analogy to the moments for rigid bodies is illustrated in table 3.1 (Schalkoff, 1989).

Table 3.1 - Analogy of moments with rigid bodies (adapted from Schalkoff, 1989).

Central moment	**Interpretation**
μ_{20}	horizontal variance about the centroid of the body
μ_{02}	vertical variance about the centroid of the body
μ_{11}	"diagonality"; indication of which quadrant, with respect to the centroid, has the most "mass".
μ_{12}	"horizontal divergence"; compares the extent of the mass on the left to extent of the mass on the right of the centroid
μ_{21}	"vertical divergence" ;compares the extent of the mass at the bottom to the mass at the top of the centroid.

The central moments (3.2.3) can be used to derive the seven moment invariants shown in table 3.2. Moment invariants are used to extract rotational, scale and translation invariant features from a digital image (Jain, 1989; Schalkoff, 1989). This method is used as an intermediate feature extraction stage between the pixel representation and the neural network in that the pixel representation f(x, y) is processed as described in equations 3.2.2 and 3.2.3 to give the seven moment invariants ($\phi_1 .. \phi_7$), shown in table 3.2, which are then used as input to the neural network.

Table 3.2. - Seven moment invariants derived from central moments (Jain, 1989; Schalkoff, 1989).

$\phi_1 =$	$(\mu_{20} + \mu_{02})$
$\phi_2 =$	$(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2$
$\phi_3 =$	$(\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2$
$\phi_4 =$	$(\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2$
$\phi_5 =$	$(\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12}) \cdot [(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] +$
	$(3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03}) \cdot [3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]$
$\phi_6 =$	$(\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03})$
$\phi_7 =$	$(3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12}) \cdot [(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] -$
	$(\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03}) \cdot [(3\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]$

The advantage of using these moments for feature extraction is that their calculation is straightforward in practice and that they are invariant under rotation, scale and translation. Further the dimension of the input data is significantly reduced as only the seven features in table 3.2 are used to represent the image. However there are some disadvantages to using this method. Firstly, the moments are not strictly invariant as they fluctuate slightly with variations in position, size and orientation of an image. A possible explanation for this is round-off errors introduced during computation. Secondly, the seventh moment undergoes a sign change when the image is reflected. A third problem is that the transformation is not 1:1 in that is does not provide a representation from which the original image can be reconstructed.

These moments have been used with a back propagation network for aircraft recognition (Kulkarni 1994). The moments were shown to be invariant to rotational and translational variations but did not respond well to scale changes. Reiss (1991) showed formally that in fact these moments are invariant to scale and rotation but not to translational variations and presented a set of moments that were invariant to scale, rotation and translation as well as to changes in contrast or illumination.

3.3 Zernike moments

In 1990, Khotanzad & Hong used a set of complex moments of an image as feature extractors for invariant handwritten character recognition. The features are invariant to scale, rotation and translation and are termed 'Zernike Moments'.

Zernike moments are based on an orthogonal set of complex polynomials defined over the interior of the unit circle, $x^2 + y^2 = 1$, and were introduced by Zernike in 1934. Let the set of these polynomials be denoted by $\{ V_{nm}(x,y) \}$. Then the form of these polynomials is :

$$V_{nm}(x,y) = V_{nm}(\rho, \theta) = R_{nm}(\rho) exp(im\theta), \qquad (3.3.1)$$

where

n : positive integer or zero

m: positive and negative integers subject to constraints $n - | m |$ even, $| m | \leq n$

ρ: length of vector from origin to (x, y) pixel given by $(x^2 + y^2)^{½}$.

(e.g. a pixel centred at (0.5, 0.5) has $\rho = (0.5 + 0.5)^{1/2} = 0.707$).

θ: angle between vector and x axis given by $\theta = \tan^{-1}(y/x)$.

i: $\sqrt{-1}$

$R_{nm}(\rho)$: radial polynomial defined as

$$R_{nm}(\rho) = \sum_{s=0}^{\frac{n-m}{2}} \frac{(-1)^{s} [(n-s)!] \rho^{n-2s}}{s!(\frac{n+|m|}{2} - s)!(\frac{n-|m|}{2} - s)!} .$$

Figure 3.3.1 shows ρ and θ for a pixel at location (x, y).

The complex conjugate of $V_{nm}(\rho, \theta)$ is defined as:

$$V_{nm}(\rho, \theta)^{*} = R_{nm}(\rho) exp(i(-m\theta)) \qquad (3.3.2)$$

where * denotes the complex conjugate and $R_{n,-m}(\rho) = R_{nm}(\rho)$.

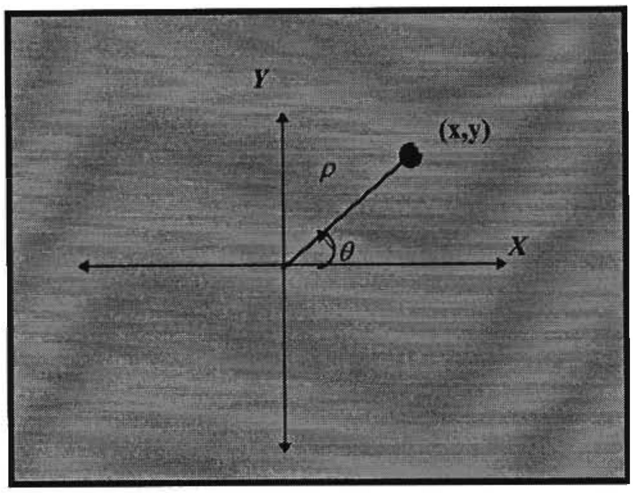


Figure 3.3.1 - θ and ρ for a pixel (x, y).

These polynomials are orthogonal since they satisfy

$$\iint_{x^2+y^2\le 1} [V^*_{nm}(x,y)]V_{pq}(x,y)dxdy = \frac{\pi}{n+1}\delta_{np}\delta_{mq} \tag{3.3.3}$$

with

δ_{ab} = 1 if a=b and δ_{ab} = 0 otherwise.

In words, this expression is 0 for all polynomials V_{nm} and V_{pq} except when n=p and m=q, that is when V_{nm} and V_{pq} are the same polynomial, in which case it is $\pi/(n+1)$. Elements in orthogonal sets are independent of each other, i.e. a change in one element does not necessarily result in a change in another element.

The complex Zernike moments, A_{nm}, for a digital image f(x,y) are calculated as:

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x,y) V_{nm}^*(\rho,\theta) \qquad x^2 + y^2 \leq 1 \qquad (3.3.4)$$

$$= \frac{n+1}{\pi} \sum_x \sum_y f(x,y) V_{nm}^*(\sqrt{x^2+y^2}, \tan^{-1}\frac{y}{x}),$$

where

n : denotes the order and is a positive integer or zero and is normally ≤ 12.

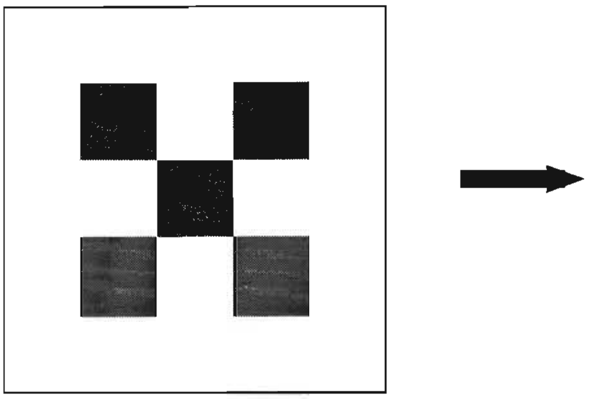
m: denotes the repetition and takes values of positive and negative integers subject to constraints n- | m | even, | m | ≤ n

To compute the Zernike moments of an image, the centre of the image is taken as the origin and pixel coordinates are mapped to the range of the unit circle $x^2 + y^2 \leq 1$. Suppose the digital image is a 5 x 5 binary image. The image can be represented in the form of a matrix indicating active pixels with a 1 and inactive pixels with 0 as illustrated in figure 3.3.2. The image is then mapped onto the unit circle as shown in figure 3.3.3. Only points that are within the unit circle $x^2 + y^2 \leq 1$ are included, while points outside the unit circle, denoted by * in figure 3.3.3, are not included in the calculation.

The magnitude of these complex moments $|A_{nm}|$ can be taken as rotation invariant features of an image, but not as scale and translation invariant features. To achieve translation invariancy, the image is moved to its centroid $(\bar{x},\bar{y})$, which is calculated as

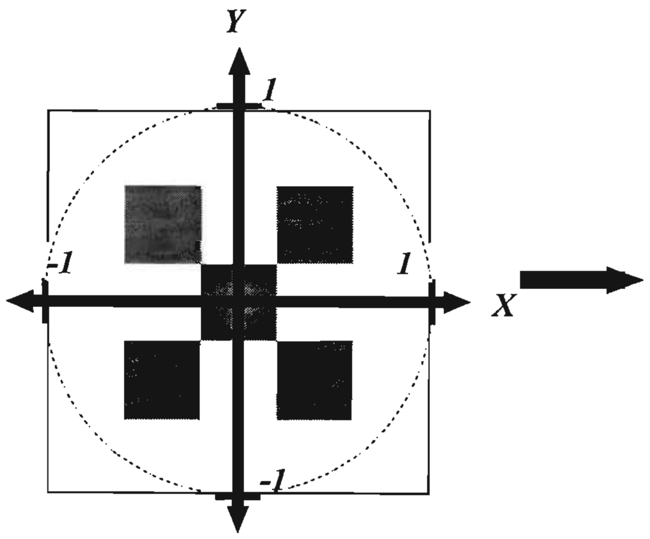
$$\bar{x} = \frac{m_{10}}{m_{00}} \; and \, \bar{y} = \frac{m_{01}}{m_{00}},$$

where m_{pq} is a regular moment as defined earlier in (3.2.1), to give a new image $f(\bar{x}+x, \bar{y}+y)$ which is standardized with respect to translation.



$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 \\
3 & 0 & 1 & 0 & 1 & 0 \\
4 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Figure 3.3.2 - Binary image represented as a matrix.



$$\begin{array}{c|ccccc}
 & -1 & -.5 & 0 & .5 & -1 \\
\hline
1 & * & * & 0 & * & * \\
.5 & * & 1 & 0 & 1 & * \\
0 & 0 & 0 & 1 & 0 & 0 \\
-.5 & * & 1 & 0 & 1 & * \\
-1 & * & * & 0 & * & *
\end{array}$$

Figure 3.3.3 - Binary image mapped to the unit circle.

Scale invariancy is achieved by setting the first moment m_{00}, which is actually the number of black pixels in the digital image, to a predetermined value β by defining a factor

$$a = \sqrt{\frac{\beta}{m_{00}}}.$$

The image $f(\frac{x}{a}, \frac{y}{a})$ which is standardized with respect to scale can then be generated.

Combining the above transformations, the image

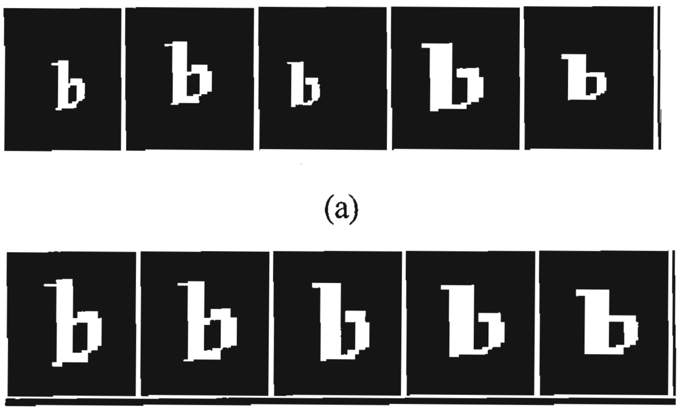
$$g(x,y) = f(\bar{x} + \frac{x}{a}, \bar{y} + \frac{y}{a}) \tag{3.3.5}$$

can be defined which is standardized with respect to both scale and translation.

The Zernike moments defined in equation (3.3.4) are calculated for the standardized images obtained from (3.3.5), and their magnitudes, denoted as $|A_{nm}|$ can be used as rotational invariant features of the image. The scale and translation standardization process affects two of the Zernike features, $|A_{00}|$ and $|A_{11}|$ which remain constant for any standardized images at β/π and 0 respectively. Therefore these moments are not utilized in the classification task and only those of second order or higher, i.e. $n \geq 2$, are used. Since the Zernike moments are calculated for the standardized image, the magnitudes of the Zernike moments are scale, translational, as well as rotational invariant features of the original digital image f(x, y).

Example 1

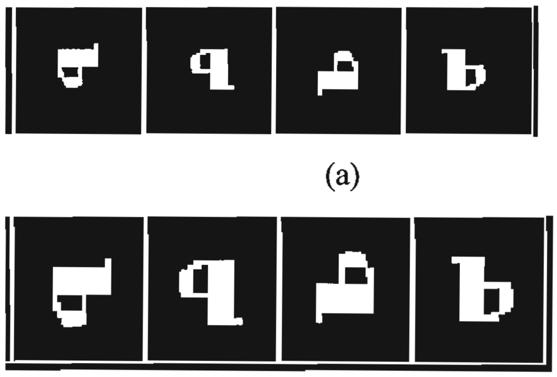
To illustrate the use of Zernike moments as an invariant feature extraction method, five 64 x 64 binary images of the character "b" were standardized with respect to scale and position with β set to 500. These characters are shown before and after standardization in figure 3.3.4. The images were then rotated by 90°, 180° and 270° as shown in figure 3.3.5. The magnitudes of 4 Zernike moments for the standardized images were calculated and are shown in Table 3.3. They were found to be equal.



(a)

(b)

Figure 3.3.4 -Five instances of the character b (a) before and (b) after standardization.



(a)

(b)

Figure 3.3.5 - The letter b rotated through angles of 90°, 180°, 270°.

Table 3.3. - The magnitudes of 4 Zernike moments for the rotated images in figure 3.3.5.

	90°	180°	270°	0°
|A[2,0]|	207.935751	207.935751	207.935751	207.935751
|A[2,2]|	3.246915	3.246915	3.246915	3.246915
|A[3,1]|	52.198420	52.198420	52.198419	52.198419
|A[3,3]|	2.201566	2.201566	2.201566	2.201566

Zernike moments have been used in conjunction with the multilayered perceptron for handwritten character recognition (Khotanzad & Lu , 1991; Perantois , 1992). Khotanzad & Lu used 32 x 24 binary images of handwritten numerals and a training set of 13 250 images were generated by selecting 1 325 characters of each numeral at random from a database of 86 000 images. A test set of 3 300 characters was created by randomly selecting 330 characters of each numeral from the remaining characters in the data base. The input to the MLP was 47 Zernike features of order 2 to 12. The accuracy rate obtained with β=150 and using 50 hidden nodes was 83.8%. This was compared to the accuracy rates of conventional classification methods of 83.45% for nearest-neighbour and 58.31%. for minimum-mean-distance. Considering the large amount of variation and distortion the results obtained are very good.

3.4 FOURIER DESCRIPTORS

Image processing methods that involve the direct manipulation of the pixel values from a digital image are known as spatial domain methods. Another important domain in image processing is the frequency domain in which the Fourier transform of an image is used to represent it (Schalkoff, 1989; Lewis, 1990, Jahne, 1993).

3.4.1 The Fourier transform

Instead of storing brightness values at each position (x,y), the Fourier transform stores the amplitudes of spatial frequencies. Spatial frequency is a measure of the brightness variations in an image. Areas of an image that contain fine detail or texture exhibit high spatial frequencies. An area which consists of smoothly changing brightness has little high frequency component. Increasing the low frequency component of an image is equivalent to blurring it and increasing the high frequency component of an image makes the image appear sharper (Lewis, 1990).

The two dimensional discrete Fourier transform of an mxn digital image f(x, y) is given by :

$$F(u,v)=\frac{1}{mn}\sum_{x=0}^{m-1}\sum_{y=0}^{n-1} f(x,y)\exp[-2\pi i(\frac{ux}{m}+\frac{vy}{n})], \tag{3.4.1}$$

for u = 0,1,2, ... , m - 1 and v = 0,1,2, ..., n - 1

where

u , v : spatial frequency variables in the frequency domain with u being the spatial frequency component along the x axis, and v being the spatial component along the y axis in the spatial domain.

x, y : position variables in the spatial domain.

The Fourier representation of an image may be transformed back to a spatial representation using the inverse discrete Fourier transform.

$$f(x,y)=\sum_{u=0}^{m-1}\sum_{v=0}^{n-1} F(u,v)\exp[2\pi i(\frac{ux}{m}+\frac{vy}{n})] \tag{3.4.2}$$

for x = 0,1,2, ... , m - 1 and y = 0,1,2, ..., n - 1

3.4.2 Invariant feature extraction

The Fourier representation has the following interesting characteristics:

1. The magnitude of the Fourier transform, |F(u, v) |, is not affected by translation or shift in origin of u and v and is known as the Fourier spectrum.
2. The Fourier distribution is symmetric in the (u, v) plane.
3. The Fourier distribution rotates with rotations of the input image.
4. Points closer to the origin represent lower spatial frequencies and points further away from the origin represent higher spatial frequencies.

These characteristics can be used for invariant feature extraction (Kulkarni, 1994). Since the distribution is symmetrical, only half of the (u, v) plane is needed to capture all the information of the Fourier representation. Thus the (u, v) plane is split into angular and radial bins as shown in figure 3.4.1.

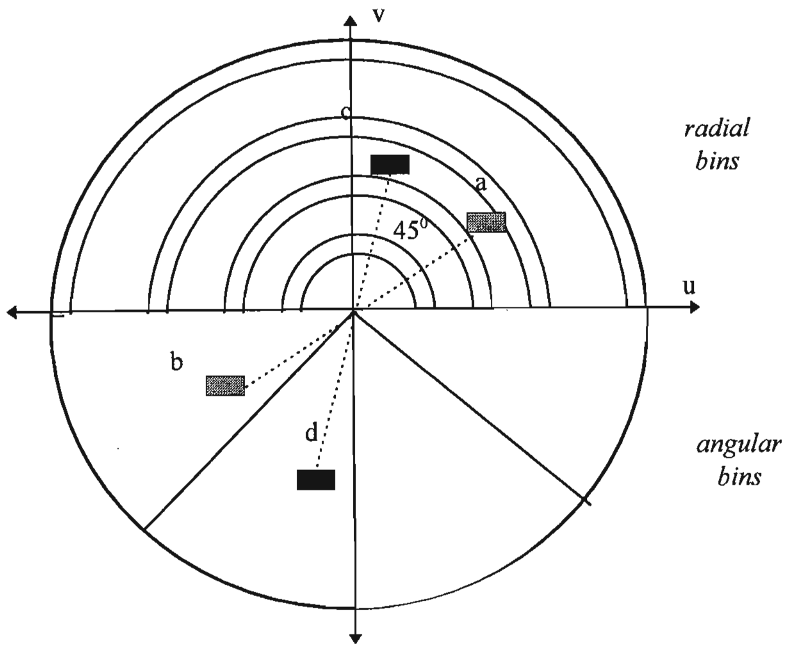


Figure 3.4.1 - Radial and angular bins in the frequency domain (Kulkarni, 1994).

The distance from the origin to a point in the frequency domain determines into which radial or ringed shaped bin the point falls, and since this is a measure of spatial frequency or coarseness of the image, the radial bins capture the coarseness or scale of the image. However radial bins are insensitive to rotation of the input image as the Fourier representation undergoes the same rotation. Suppose the Fourier transform of a digital image produced the response (a) and (b) in the frequency domain indicated by the grey areas in figure 3.4.1. Since the Fourier descriptor rotates with rotations in the input image, the Fourier transform of the image rotated 45^0 in a counter-clockwise direction would then produce the responses (c) and (d) shown as black areas in figure 3.4.1. Notice that the area (c) falls within the same radial bin as (a) but the area (d) falls in a different angular bin from (b). This illustrates how radial bins are insensitive to rotational changes.

On the other hand, wedge shaped angular bins capture orientation information about the image as the angular bin in which a point falls will be determined by the orientation of the image. Rotation of the image will cause the point to fall into a different angular bin. However angular bins are insensitive to scale or coarseness of the image. If the coarseness of the image is increased uniformly, then the grey areas, (a) and (b), will move closer to the origin, thus falling in the same angular bin but a different radial bin.

Radial and angular bins can be combined to extract rotational and scale invariant features from an image, which together with the magnitude of the Fourier transform can be made to be translational invariant.

3.5 GABOR FILTERS

Different patterns in images can be distinguished by their preferred direction or orientation in the image or by their spatial frequency[1], that is, how fine (more detail) or how coarse (less detail) the pattern is. The role which orientation and spatial frequency play in distinguishing amongst these patterns, more commonly known as textures, is described below.

3.5.1 Orientation and spatial frequency of texture

Local orientation plays an important role in an image. This is illustrated in figure 3.5.1. The number 4, is of the same colour as the background, has the same spatial frequency as the background but is in a different orientation, which allows us to identify it from the background. "Local" indicates a specific area in the image, in this example the area of the '4', and not the whole image.

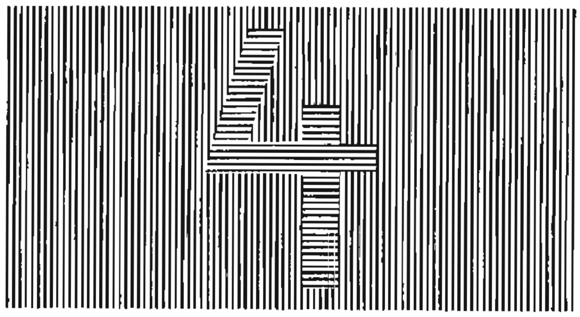


Figure 3.5.1 - Image illustrating importance of local orientation in images.

[1] In a binary image (2 colours, black and white) the spatial frequency of a small area can be thought of as how fast the pixels change from black to white in that area. Thus areas with fine detail will have high spatial frequencies

A texture can be characterised by its preferred direction (orientation) or spatial frequencies (coarseness). The two textures in figure 3.5.1 have the same spatial frequency but different orientations while the two textures in figure 3.5.2 have different spatial frequencies, but the same orientation.

Texture can be used in the analysis of images in several ways, for example in the segmentation of objects in images e.g. in figure 3.5.1 the '4', an object, can be separated or segmented from the background, another object, using their textural differences, in the classification of surface materials and in the computation of shape (Ghosh & Bovik, 1991).

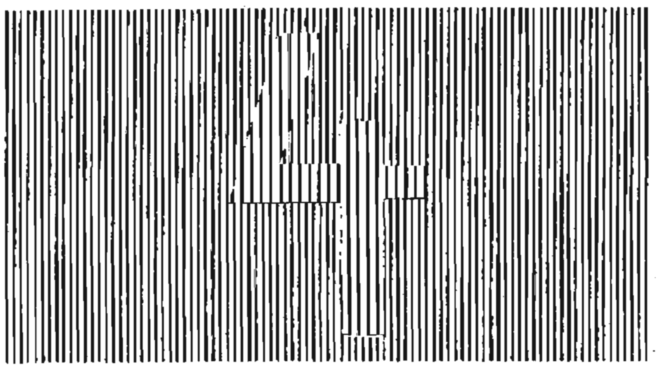


Figure 3.5.2 - Two textures with the same orientation, but different frequencies.

Currently there is no formal definition of texture, but intuitively texture descriptors provide measures of properties such as coarseness, smoothness and regularity (Kulkarni, 1994; Ghosh & Bovik, 1991). Different objects in an image generally differ in texture. If we are able to distinguish between areas of uniform textures in an image it will be easier to identify objects in an image. The process of identifying regions of uniform or consistent textures in an image is known as texture segmentation.. Transitions in texture can either be sudden or gradual. Sudden texture transitions, which usually occur because of orientation changes between texture types as shown in figure 3.5.1, can be identified more easily in the spatial domain (the common two dimensional pixel array representation) while gradual variations in texture, which are normally caused by variations in spatial frequency, are more difficult to detect spatially and are more suited to the spatial frequency domain. This is illustrated in figure

3.5.3 which depicts two textures, with the central texture created by a sudden change in spatial frequency. Notice how difficult it is to distinguish spatially between the two textures. However there are instances where textures with different spatial frequency are detected easily in the spatial domain e.g. figure 3.5.2, but generally, detection of variations in spatial frequency are more suited to the frequency domain.

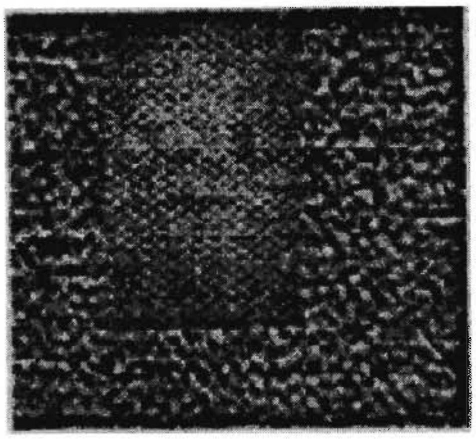


Figure 3.5.3 - Two textures with the same orientation but different frequencies, which can not be distinguished easily in the spatial domain..

Since both the spatial and frequency domains offer advantages in texture segmentation, the problem as to whether to represent the image in the spatial or spatial frequency domain arises. The solution is to use a representation that incorporates both the spatial and spatial frequency attributes of an image. Daugman showed that two dimensional Gabor filters provide a conjoint representation of a two dimensional image in both the spatial and spatial frequency domain (Daugman & Kammen, 1987), that is, each filter responds to a specific range of orientations and frequencies[2]. Gabor filters have been used extensively for texture segmentation (Dunn et al. , 1994; Bovik et al, 1990; Lisboa, 1992, ; Lu et al. , 1991; Greenspan et al. 1991; Bisio et al, 1993; Ghosal & Mehrotra, 1993; Kulkarni, 1994; Ghosh & Bovik, 1991).

[2] frequency refers to spatial frequency .

3.5.2 Biological motivation

The use of Gabor filters for two dimensional image representation was inspired by the way images are represented in the mammalian primary visual cortex. Since both Gabor filters and neural networks are biologically inspired, a combination of them would seem to be plausible.

Gabor filters have been shown to resemble receptive fields of simple cells in the mammalian primary visual cortex (Daugman & Kammen, 1987). This is illustrated in figure 3.5.4 where the receptive field profiles of simple cells in the visual cortex of a cat are compared with the best fitting two dimensional Gabor filter. The top row of figure 3.5.4. shows the inhibitory and excitatory effects that small spots of light on a 16x16 grid have on a simple cell. A high peak signifies a high excitatory effect, while a low peak signifies a high inhibitory effect. The Gabor filters shown in the middle row were obtained using equation 3.5.1 on page 40.

Simple cells are stimulated by a specific orientation of an edge or blob of light in their receptive fields. The localised orientation selectivity and spatial frequency selectivity of these cells can be simulated by a two dimensional Gabor function as illustrated in figure 3.5.5.

The Gabor function is localised[3] in both the spatial domain (orientation) and the frequency domain (frequency selectivity). In other words a Gabor filter responds to a limited range of orientations in the spatial domain and a limited range of frequencies in the frequency domain, as shown in figure 3.5.5, and thus constitutes a two dimensional bandpass filter (Daugman & Kammen, 1987). Two dimensional bandpass filters can be used to decompose an image into sub-images, each containing different ranges of orientations and frequencies, as accomplished by the simple cells in the mammalian visual cortex.

3.5.3 Definition

Gabor functions consist of sinusoids with quadratic envelopes, parameterized by two constants α and β together with expansion centers (x_0,y_0) in space and (u_0,v_0) in the frequency domain.

[3] A function is *local* if most of its area is concentrated in a compact region. An example of a local function is the normal distribution.

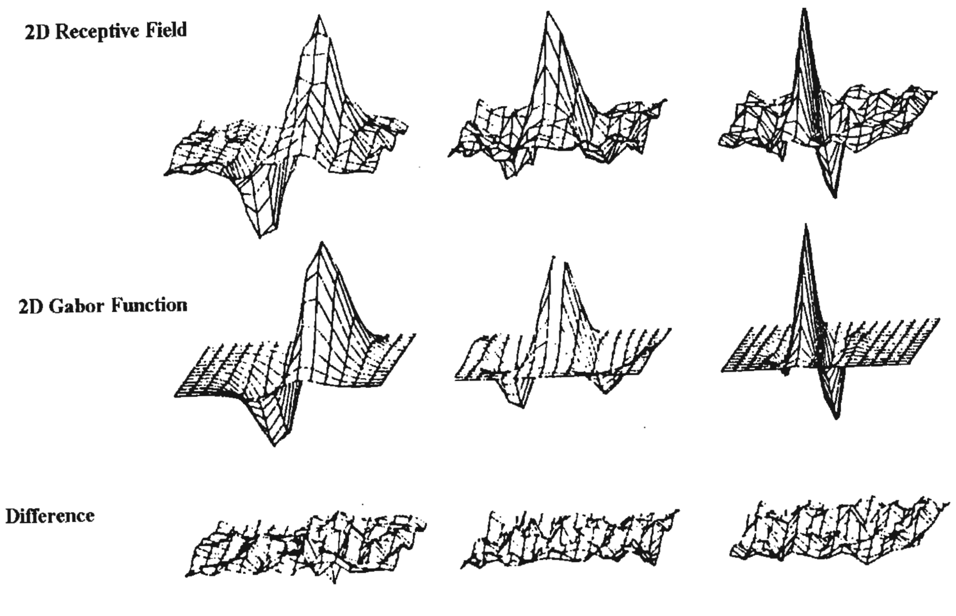


Figure 3.5.4 - Comparison of the receptive field profile of simple cells in the cat visual cortex to best fitting two dimensional Gabor functions. The bottom row shows the residual errors which are considered to be random (from Daugman & Kammen, 1987).

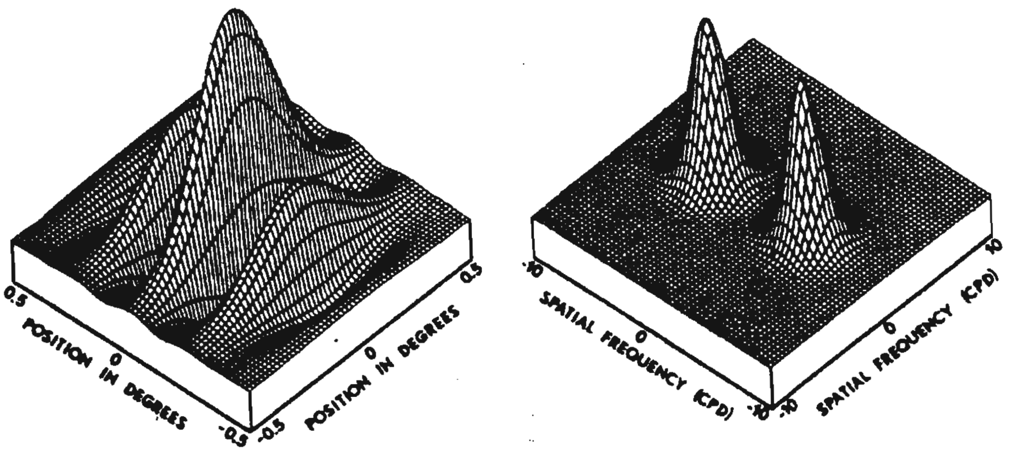


Figure 3.5.5 - An example of a two dimensional Gabor function showing orientation selectivity in the spatial domain (left) and frequency selectivity in the frequency domain ,the Fourier transform (right) from Daugman & Kammen,1987.

In the spatial domain the Gabor function is defined as:

$$G(x,y) = exp\{ -\pi [(x-x_0)^2.\alpha^2 + (y-y_0)^2.\beta^2] - 2\pi i[u_0.(x-x_0)+v_0.(y-y_0)]\} \qquad (3.5.1)$$

and in the frequency domain as :

$$F(u,v) = exp\{ -\pi [(u-u_0)^2 / \alpha^2 + (v-v_0)^2 / \beta^2] -2\pi i[x_0.(u-u_0)+y_0.(v-v_0)]\}. \qquad (3.5.2)$$

It can be shown through the uncertainty principle (Daugman, 1988) that there is an inverse relationship between the orientation ranges and frequency ranges to which each Gabor filter responds. A Gabor filter that responds to a small orientation range (orientation bandwidth) will respond to a large frequency range (frequency bandwidth). This means that the filter is more sensitive to orientation than frequency. In other words it responds to a large range of frequencies and cannot distinguish small changes in frequency but responds to a small range of orientations and can thus distinguish small changes in orientation. Further discussion on the uncertainty principle may be found in Wilson & Granland (1984), and its applicability to the two dimensional Gabor filters may be found in Maclennan (1991) and Navarro et al. (1995).

Families of Gabor filters

A family of Gabor filters can be generated using the elementary Gabor function G(x,y) in equation 3.5.1 as

$$\begin{aligned} &G_{pqm\theta}(x,y) = 2^{-m}.G(x',y') \\ &where \\ &x' = 2^{-m}.((x-p)\cos\theta + (y-q)\sin\theta) \\ &y' = 2^{-m}.(-(x-p)\sin\theta + (y-q)\cos\theta), \end{aligned} \qquad (3.5.3)$$

where

p, q	: the (x ,y) coordinates of a spatial expansion center.
m	: integer labeling the tiers, in figure 3.5.5, radially starting with m= 0 at the

outer ring of ellipses.

θ : orientation of the filter, angle of the line joining the origin and the center of the ellipse in figure 3.5.6.

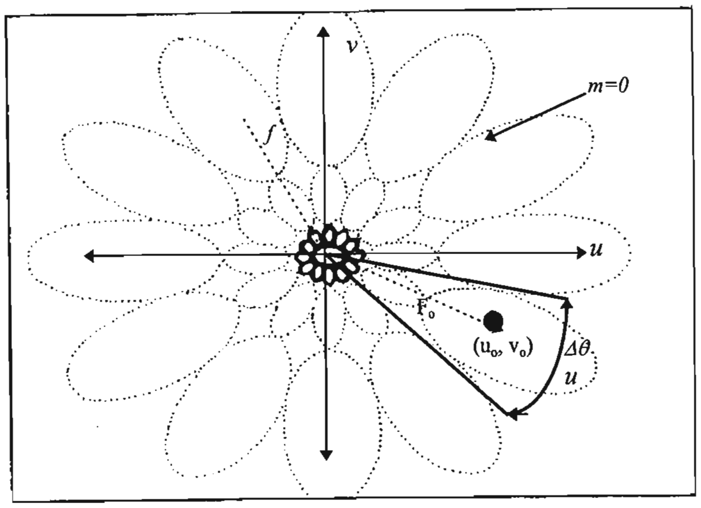


Figure 3.5.6 - Log polar distribution of two dimensional Gabor filters (adapted from Lisboa 1992).

Figure 3.5.6 depicts the log polar distribution of these Gabor filters in the frequency domain. The frequency distribution is symmetric about the origin and thus a single Gabor filter is depicted by a pair of symmetric ellipses. For example, consider an ellipse in tier m=0. Then the ellipse directly opposite it in the same tier (m=0) is its symmetric counterpart. These two ellipses constitute a Gabor filter. Each pair of ellipses is centred at a specific distance from the origin. This distance is the central frequency, F_0, of the Gabor filter, where $F_0 = (u_0 + v_0)^{-1/2}$. The angle of the line joining the origin and the centre of an ellipse to the u-axis is the orientation, θ, of the filter, where $\theta = \tan^{-1}(v_0 / u_0)$. Each ellipse spans an angle, $\Delta\theta$, which is the orientation bandwidth of the filters while the length of the ellipse is the frequency bandwidth, Δw, of the filter.

In figure 3.5.6, the shape of the ellipses are determined by the aspect ratio $ar \equiv \alpha / \beta$, where α and β are the constants which appear in equation 3.5.1. The trade-off between the orientation half bandwidth, $\Delta\theta_{1/2}$, which is half the orientation bandwidth, $\Delta\theta$, shown in figure 3.5.6, and the frequency bandwidth, Δw, is determined by the aspect ratio, *ar*, of the filter and is defined as:

$$(1-\Delta w) / (1+\Delta w) = (ar.\ \tan\theta_{1/2}) / (1+ ar^2.\ \tan^2 \theta_{1/2})^{1/2}. \qquad (3.5.4)$$

3.5.4 Implementation of Gabor filters

A digital image f(x,y) can be approximated by h(x,y) which is a linear combination of Gabor elementary functions, that is, a digital image can be split up into sub images, where each sub-image contains a range of orientations and frequencies as determined by its Gabor filter. Thus

$$h(x,y) = \sum_{i=1}^{n} a_i.G_i(x,y), \qquad (3.5.5)$$

where

$h(x,y)$	: approximation for the digital image f(x,y)
G_i	: the family of n Gabor filters, $G_{pqm\theta}$ generated from (3.5.3)
a_i	: weightings of Gabor filters G_i.

The *n* elementary Gabor functions in (3.5.5) are said to form a basis for *h* and the coefficients $\{a_i\}$ can be thought of as the weight of the orientations and frequencies of the *i*th Gabor filter in the input image.

Since these Gabor elementary functions do not form an orthogonal set, the calculation of the coefficients is difficult. Lisboa (1992) proposed an iterative method using least squares to represent a digital image with a combination of elementary Gabor functions. If the original image *f(x,y)* is not exactly recovered in the expansion (3.5.5) then an error can be defined between the original image f(x,y) and the calculated image h(x,y) as

$$\frac{1}{2}(f(x,y) - h(x,y))^2. \qquad (3.5.6)$$

Defining the projection

$$< f|G_{pqm\theta} > \equiv \sum_{x,y} f(x,y)G_{pqm\theta}(x,y)\,, \tag{3.5.7}$$

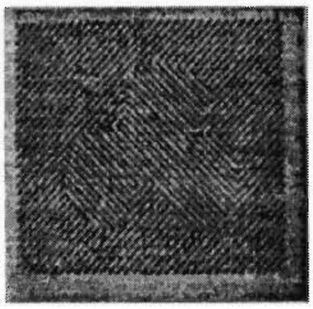
then the algorithm for updating the expansion coefficients may be written in the form

$$\begin{aligned}\Delta a_i &= < (f-h)|G_i > \\ &= \sum_{xy} (f(x,y) - h(x,y)).G_{pqm\theta}(x,y)\end{aligned} \quad . \tag{3.5.8}$$

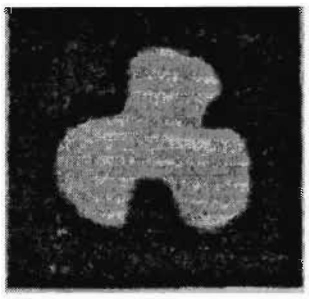
A similar approach was presented by Daugman (1990) and later modified by Ghosh and Bovik (1991) for a three layered neural network which is used to obtain the expansion coefficients, $\{a_i\}$ in equation 3.5.5, for a two dimensional Gabor transform.

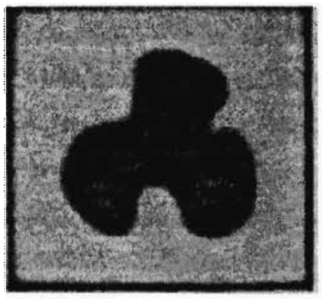
3.5.5 Examples

The following figures adapted from Bovik et al. (1990) illustrate the operation of Gabor filters. Figure 3.5.7(a) shows an image of two different textures, of the same frequency but different orientations. The responses of Gabor filters tuned to orientations of 42^0 and -43^0 are shown in (c) and (d) respectively. The dark areas indicate a response while the light areas indicate no response. However local orientation is not restricted to continuous lines. Figure 3.5.8(a) shows another image of two textures of the same frequency but different orientations. The response of a Gabor filter tuned to an orientation of 45^0 is shown in (b) and (c) is the result of segmentation using post-filtering (Bovik et al. 1990). The above two examples illustrate the orientation selectivity of the Gabor filters. Figure 3.5.9(a) shows an image with two different textures, of the same orientations but now with different frequencies. The response of a Gabor filter tuned to a frequency of 80 and a bandwidth of 0.7 is shown in (b) while (c) shows the response of a Gabor filter with a lower frequency, 40, but a larger bandwidth ,1.3. It can be seen that the areas of different frequencies cannot be clearly identified as a result of this larger bandwidth, that is, the filter is less sensitive to changes in frequency. This illustrates the frequency selectivity of Gabor filters. The above examples clearly show how Gabor filters can be used to segment textures according to their different orientations and frequencies.



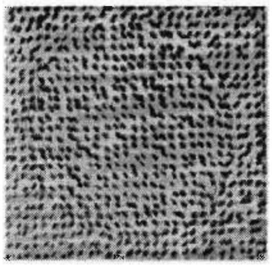
(a)

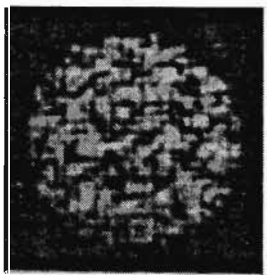




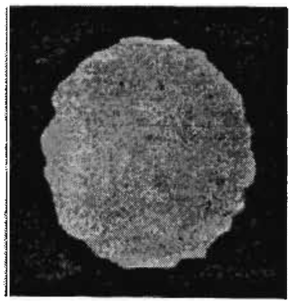
(b   (c)

Figure 3.5.7 - Segmentation of different orientations using two Gabor filter, (a) original image; (b) amplitude of Gabor filter (F, B, θ) =(50, 0.7, 42^0); (c) amplitude of Gabor filter (F, B, θ) =(50, 0.7, -43^0) adapted from Bovik et al. (1990).





(a)   (b)



(c)

Figure 3.5.8 - (a) original image; (b) amplitude of Gabor filter (F, B, θ) =(40,1, 45^0); (c) postfiltered image (adapted from Bovik et al. 1990).

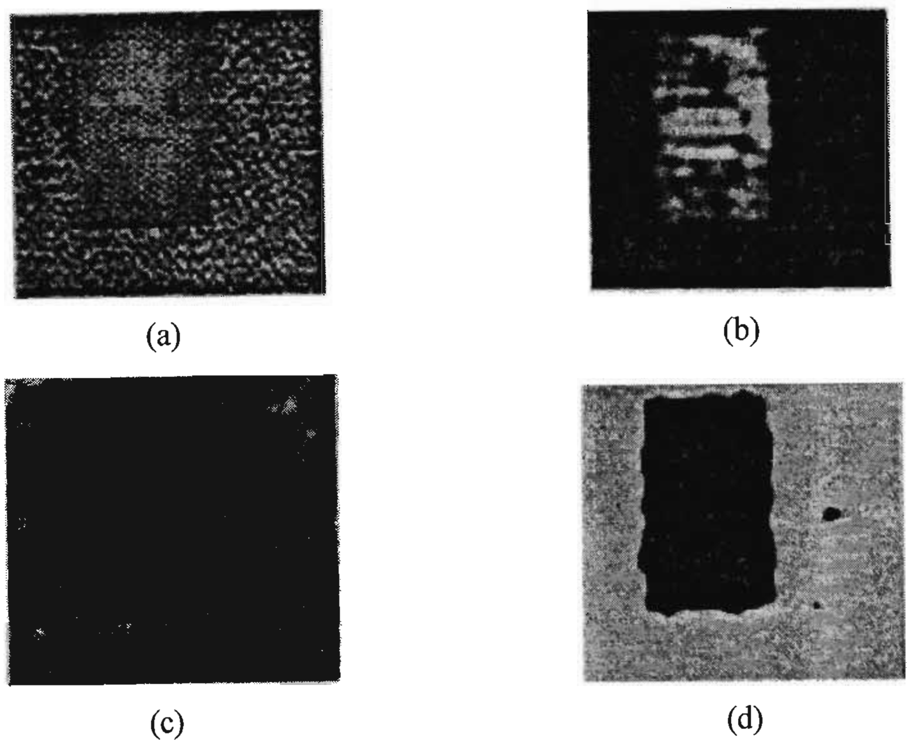


Figure 3.5.9. - (a) original image; (b) amplitude of Gabor filter (F, B, θ) =(40,1, 89^0); (c) amplitude of Gabor filter (F, B, θ) =(40,1.3, 89^0); (d) segmentation (adapted from Bovik et al. 1990).

3.5.6 Characteristics

The Gabor filter family provides the optimum trade-off between resolution for orientation and resolution for spatial frequency which is determined by the aspect ratio of the filter. Biological neurons tend to favour orientation over spatial frequency by a factor of two. Consequently the aspect ratio is usually set to 2:1.

The problem of whether to characterize images by spatial templates or frequency signatures is resolved as Gabor functions (3.5.1) allow intermediate representations, ranging from pure pixel representations (spatial templates) at the one extreme when the constant $\beta \Rightarrow \infty$ in (3.5.1), to Fourier transformations (frequency signatures) when $\alpha \Rightarrow 0$, on the other. Furthermore the optimum representation in both the spatial and frequency domains is achieved. The family of two dimensional Gabor filters have similar forms in the spatial

(3.5.1) and the frequency (3.5.2) domains which allow simple transformations between domains. A problem with Gabor filters is, that they are not orthogonal. This complicates their computation and may result in redundancy of information in the representation. However, this redundancy increases the robustness of the representation.

The above characteristics of Gabor filters are ideal for texture classification. The simultaneous localisation in both domains is important, since it facilitates identification of sudden spatial transitions between texture types in the spatial domain, which is important for segmenting images based on texture and detection of gradual variations within a textured region in the frequency domain, which is important for computing deformations arising from surface defects. Furthermore the Gabor representation results in the reduction in the size of the pixel representation of a digital image as, the coefficients of the Gabor filters ($\{a_i\}$ in (3.5.5)) can be used to represent the image.

Thus the method of Gabor filters can be used as a feature extraction method, which takes in the pixel representation of an image and produces a feature vector which can than be processed by a neural network.

3.6 THE MULTIRESOLUTION WAVELET REPRESENTATION

Mallat (1989) introduced the idea of using wavelets for a multiresolution representation of a digital image. This involves the extracting of the difference in information of an image at different resolutions. The description given here is brief and intended to convey just the concept behind the method. For further information see Mallat (1989) and Navarro et al. (1995).

Wavelets like Gabor functions are responsive to both orientation and spatial frequency. Wavelets are a family of functions that take the form:

$$\psi^{(a,b)}(x) = |a|^{-1/2}\, \psi(\,(x-b)/a\,), \tag{3.6.1}$$

where

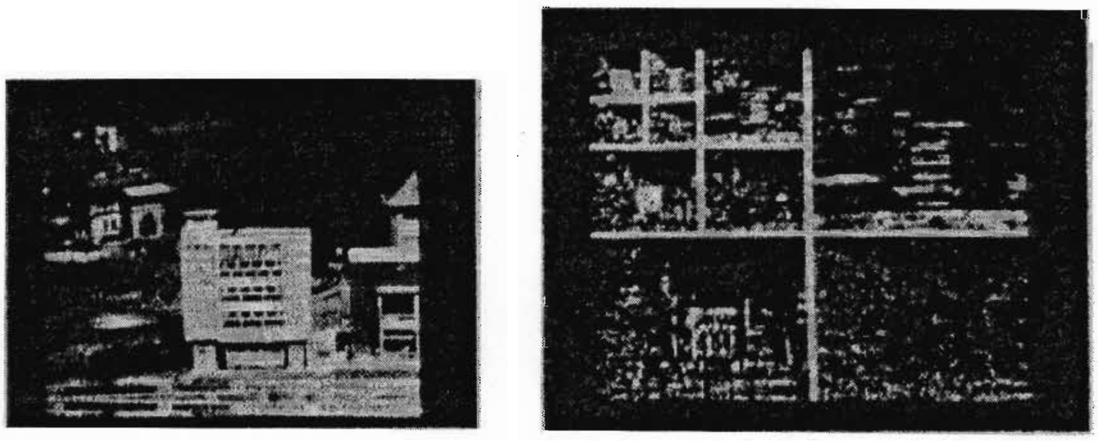
a : the dilation factor on the original function or mother wavelet $\psi(x)$,

b : translation factor on the original function.

By varying the parameters a and b, the scale (frequency) and position (space) content of the wavelet function respectively can be controlled.

The multiresolution approach

If a digital image consists of a 20 x 20 array of pixels, then the resolution of the image can be reduced by one quarter by representing the image with a 10 x 10 array of pixels. By reducing the number of pixels that represent an image, the resolution of the image and the detail in the image is decreased. Mallat's (1989) multiresolution approach to image representation involves the extraction of the difference in information from an image at multiple resolutions. The image is first divided into large neighbourhoods (coarse or low resolution) where global attributes are extracted. The resolution is then increased by a power of 2 at each iteration to give fine resolution and regional attributes are extracted. This is illustrated in figure 3.6.1 which shows an image (a) and its multiresolution decomposition (b).



(a) (b)

Figure 3.6.1 - An image (a) and its multiresolution representation (b) (adapted from Mallat, 1989).

To represent a digital image f(x, y) at a particular resolution 2^j as shown in figure 3.6.1b, four terms, calculated using the wavelets defined in (3.6.1), are required:

1. $A^d_{2^j} f(x,y)$
2. $D^1_{2^j} f(x,y)$
3. $D^2_{2^j} f(x,y)$
4. $D^3_{2^j} f(x,y)$

The letter j indicates the level of resolution e.g. resolutions of 1, ½ and ¼ correspond to j=0, -1, -2 respectively. The first term (1) is the best approximation of the original image f(x, y) at the resolution 2^j. Terms (2), (3) and (4) denote the differences between the image at the previous resolution, 2^{j+1}, and the current resolution 2^j where (2), (3) and (4) correspond to different frequencies and orientation, that is, (2) responds to vertical edges, (3) to horizontal edges and (4) to both vertical and horizontal edges. The absolute values of the wavelet coefficients (2), (3) and (4) are shown in (b) at the top right, bottom left and bottom right respectively. Note that the dark areas indicate a response and the lighter areas no response. The top left of figure 3.6.1b shows (1) further decomposed at the next resolution. Note the vertical edges, horizontal edges and both horizontal and vertical edges detected in the top right the, bottom left and the bottom right of figure 3.6.1b respectively.

The log polar Gabor representation in section 3.5 have been referred to as Gabor-wavelets (Daugman, 1988) because the equation used to generate them, equation (3.5.5), bears a resemblance to the generation of wavelets shown in equation (3.6.1).

It has also been suggested that wavelets resemble elements in biological visual systems (Boubez, 1993). The building blocks at the lower levels of the visual system are the receptor neurons, which react to stimuli within a relatively small neighbourhood of their input space, called their receptive field. In many systems these receptors are connected in a hierarchical structure to form increasingly complex receptive fields. Wavelets are a family of functions which exhibit similar properties of local support and hierarchical structure.

Laine & Fan (1993) discussed wavelet representations, based on Mallat's (1989) multiresolution approach, as feature extractors for texture classification. Feature sets ranging from 17 (standard wavelets) to 341 (complete wavelet scheme) were used for the representation of 128 x 128 digital images. Close to 100% accuracy was achieved using a backpropagation network, thus justifying the use of wavelets for texture segmentation.

3.7 THE NEOCOGNITRON

The image recognition models that have been reviewed so far consist of two distinct stages, the feature extraction stage e.g. Zernike moments, and the classification stage in which the multilayered perceptron is used. The Neocognitron (Fukushima et al. 1983, Fukushima 1988) is a multilayered network, based on the biological visual system, that incorporates the feature extraction process within the network

3.7.1 Structure

The Neocognitron is a three dimensional network containing a series of two dimensional layers as illustrated in figure 3.7.1.

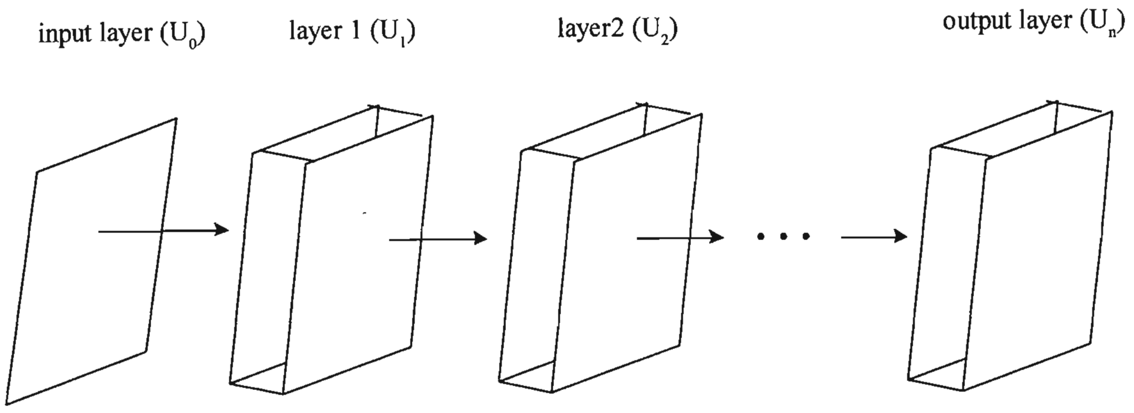


Figure 3.7.1 - Three dimensional structure of the Neocognitron.

A two dimensional digital image forms the input layer and provides inputs to the next layer, layer 1 which in turn provides inputs to layer 2 and so on until the output layer is reached.

Layers are denoted by the letter U with an index denoting the position of the layer. Thus U_0 is the input layer, U_1 denotes layer 1 and so on. Figure 3.7.2. depicts a single layer in the Neocognitron. Each layer consists of two sub layers, each containing two dimensional arrays of cells called planes, the first containing simple cell planes which consist of simple cells and the second, complex cell planes consisting of complex cells. The cells are termed simple and complex after the simple and complex cells in biological vision and have two states, active and inactive. For the sake of simplicity the letters *S* and *C* are used to denote simple cells and complex cells respectively and thus U_{s1} and U_{c1} will refer to the simple and complex cell planes of layer 1 respectively.

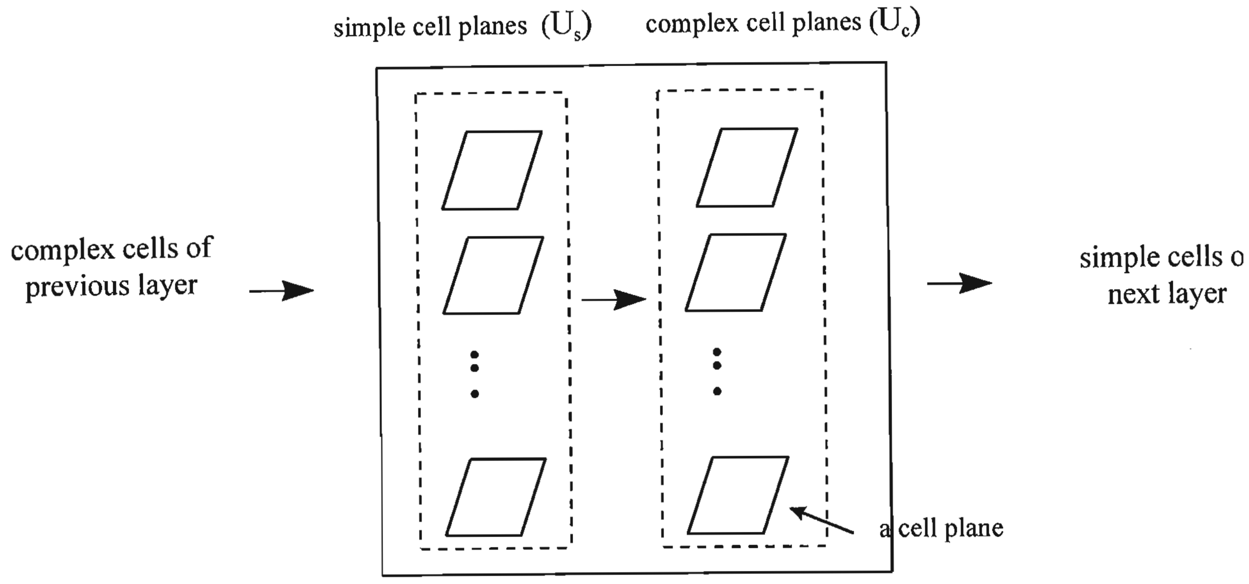

simple cell planes (U_s)
complex cell planes (U_c)
complex cells of previous layer
simple cells o
next layer
a cell plane

Figure 3.7.2 - A layer in the Neocognitron containing sub layers of simple cell planes and complex cell planes.

3.7.2 Operation

Figure 3.7.3 shows the structure of a Neocognitron which is used for the classification of 19x19 images of handwritten characters and which is used here to discuss the operation of the Neocognitron.

<u>Input (U_0) to simple plane (U_{s1})</u>

U_0 is a 19 x 19 digital image which forms the input to U_{s1}. U_{s1} consists of 12 planes, where each plane is a 19x19 array of simple cells. The cells in each plane receive inputs from a 3x3 neighbourhood of pixels in U_0. The position of this neighbourhood, called the receptive field of the cell, is determined by the position of the cell in its plane, in that the coordinates of the cell in the plane are used as the centre of its 3x3 input neighbourhood. These 19x19 neighbourhoods actually cover a 21x21 region. However since U_0 is only a 19x19 array of cells, the neighbourhoods at the edges of U_0 cover an extra imaginary layer of cells.

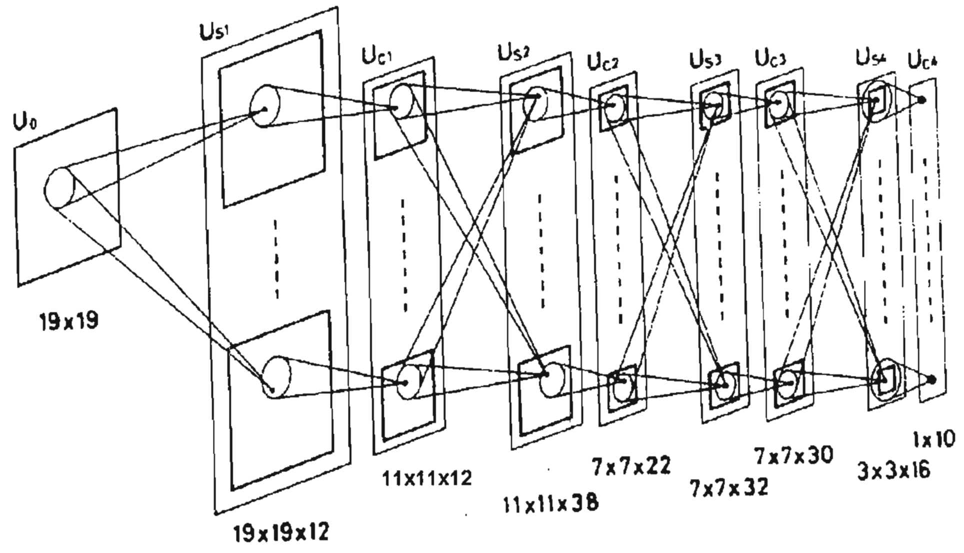


Figure 3.7.3. - Structure of a Neocognitron for classification of handwritten numerals (Fukushima et al., 1983).

This means that adjacent cells in a plane receive inputs from neighbourhoods in U_0 that are centered one pixel apart. Simple cells at the same coordinates but in different planes, receive inputs from the same neighbourhood of cells in U_0.

The simple cells in U_{s1} are responsive to specific features in the input image determined by the plane to which they belong. These feature take the form of 3x3 masks. Suppose that the cells in the first plane in U_{s1} respond to the horizontal edge shown in figure 3.7.4. Then a cell in this plane fires only if this pattern occurs in its corresponding 3 x 3 neighbourhood in U_0.

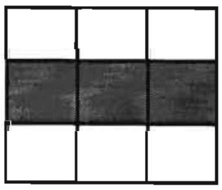


Figure 3.7.4 - Example of a 3x3 feature or mask to which cells in a simple plane respond.

Connections converging to simple cells are modifiable and are strengthened during learning. Each plane of simple cells responds to a particular feature which is determined during learning. In other words the simple cells in the same plane respond to the same feature but at slightly different positions.

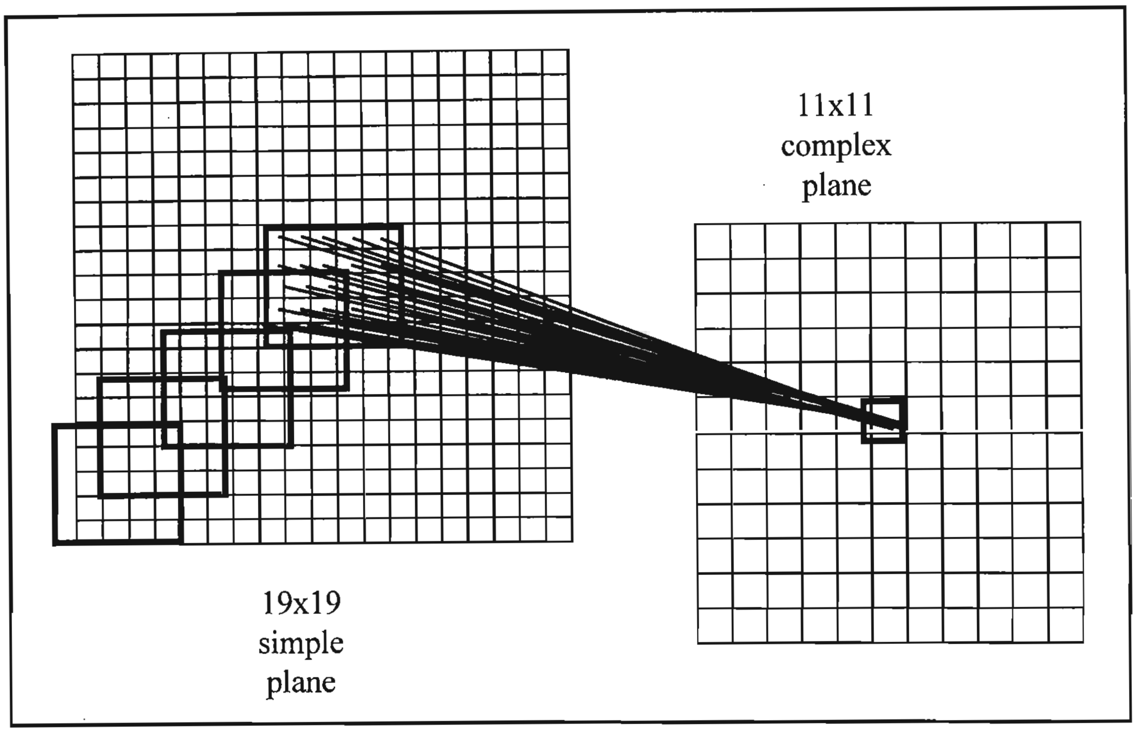


Figure 3.7.5 - Input neighbourhoods of complex cells.

<u>Simple plane (U_{s1}) to complex plane (U_{c1})</u>

Simple and complex planes occur in pairs i.e. each complex plane has a corresponding simple plane with the cells in a complex plane only receiving inputs from the cells that plane. A 5x5 neighbourhood of simple cells in the corresponding plane in U_{s1} forms the input for each complex cell in U_{c1}. These input neighbourhoods in U_{s1} are shown in figure 3.7.5. A complex cell fires if any of the cells in its input neighbourhood are active. Thus if simple cell S in U_{s1}, which provides input to complex cell C in U_{c1}, responds to a specific feature e.g. the edge in figure 3.7.4, and fires, then complex cell will also fire. If this edge is shifted a little in position in U_0, then another simple cell R in the same plane in U_{s1} and near S will fire. If S falls within the input neighbourhood of C, then C will fire. In this way complex cells respond to the same features as its simple cells in the corresponding plane, but are over a wider area in the input image, U^0.

Complex (U_{c1}) to simple (U_{s2})

As shown in figure 3.7.3 cells in U_{c1} form the input to cells in U_{s2}. The determining of the input neighbourhoods is exactly as described above for the simple cells in U_{s1}, the only difference being that cells in U_{s2} received inputs from all 12 planes in U_{c1}. For example, a cell at position (i, j) in the first plane in U_{s2} will receive inputs from twelve 3x3 neighbourhoods of complex cells centered at (i, j) in each of the 12 planes in U_{c1}. As previously, each plane of simple cells in U_{s2} detect the same feature, at slightly different positions in the complex planes of U_{c1}. Furthermore, since the cells in U_{s2} receive inputs from all the planes in the U_{c1}, they are have the ability to detect combinations of the features which are detected in U_{c1} over a larger area in U^0.

This trend continues through the network with cells in higher layers responding to more complex features over larger neighbourhoods in U^0. At the output layer, U_{c4} in figure 3.7.3 each plane consists of a single cell which responds to the entire input image, that is, the eventual neighbourhood in the input image to which the cell responds, is the entire image.

Figure 3.7.6 depicts a simple network which captures the basic operation of the Neocognitron. The network consists of the input layer and layer 1 through to layer n, the output layer. For the sake of simplicity the layers are not divided into complex and simple sub layers. Each cell in layer 1 corresponds to a 3 x 3 neighbourhood in the input image centered at the same coordinates. If the pattern shown on the left of layer 1 is detected in this neighbourhood then the cell fires, i.e. has a value of 1 (grey); otherwise it is 0 (white). Cell *a* in the top plane of layer 1 in figure 3.7.3 is activated because a horizontal edge, denoted by *p1*, was detected in its corresponding neighbourhood, denoted by *n1*, in the input image.

The cells in the middle plane respond to a corner pattern. Thus the corner pattern of the seven in the input image causes the cell *b* at the corresponding position in the middle plane to respond. Similarly the diagonal edge patterns in the input image cause the corresponding cells *c* and *d* in the third plane to respond. Thus active cells in layer 1 indicate the presence of a specific feature at the corresponding position in the input image. The input to cells in layer 2 are neighbourhoods of cells in the first layer. However the cells in layer 2 receive inputs from all the planes in the first layer. Thus a cell in layer 2 will receive inputs from three 3 x 3 neighbourhood from 3 planes in layer 1. Thus the planes in layer 2 can respond to more complex features, which are a combination of the features found in the layer 1, e.g. a combination of an horizontal edge and a corner.

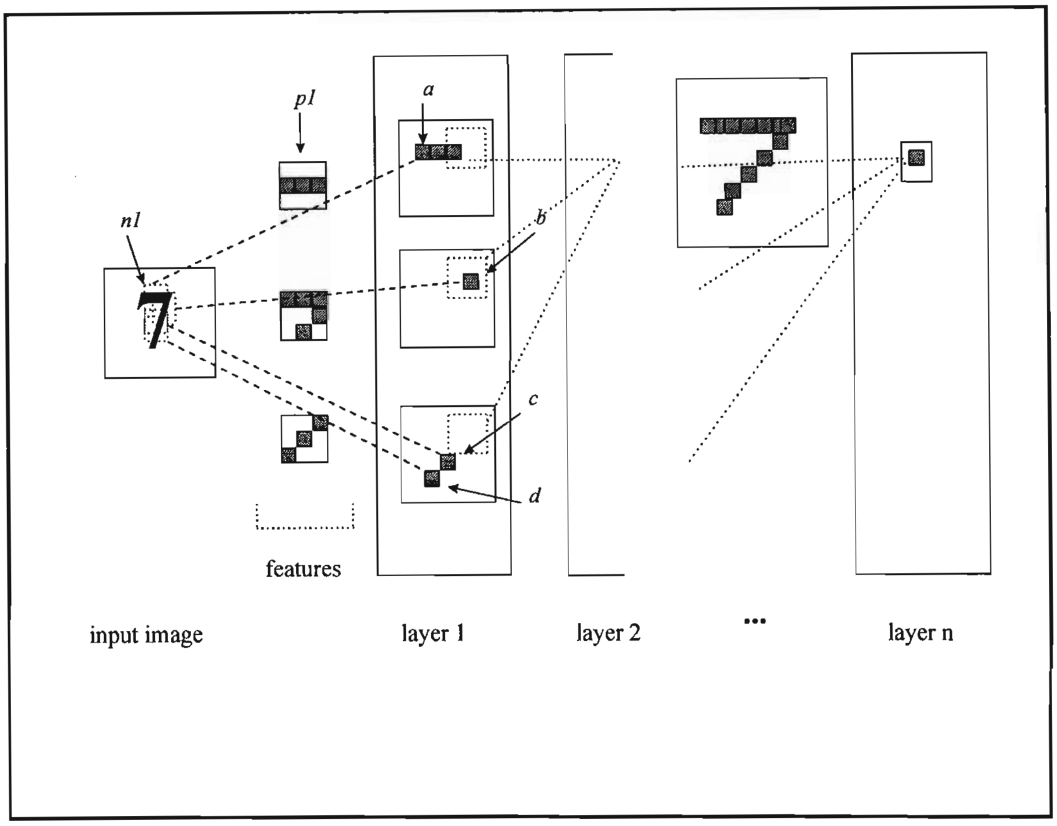


Figure 3.7.6. - Illustration of the operation of the Neocognitron.

Each cell in layer 2 receives input from a 3 x 3 neighbourhood of cells in layer 1 which in turn received input from a 3 x 3 neighbourhood in the input image. Since 3 horizontally adjacent 3x3 neighbourhoods cover a 3x6 area in the input image, then 3x3, 3x3 neighbourhoods, will cover a 6x6 area in the input image. Thus cells in layer 2 are actually responsive to an area twice the area of the receptive field for layer 1.

The features detected in layer 2 may be a long horizontal edge pattern, represented by two active cells next to each other in layer 1, e.g. cell *a* and the cell on the right of it, a corner pattern which is a single active cell, shown by cell *b* in the middle plane of layer 1 and a longer diagonal edge pattern e.g. cell *c* and the cell diagonal to it in the bottom plane of layer 1. The output layer can combine the long horizontal edge pattern, the corner pattern and the extended diagonal pattern detected in previous layers to detect a "7" .

3.7.3 Implementation

The implementation of a simple cell will now be discussed. A simple cell receives two inputs, an excitatory input, E, which is the weighted sum of the outputs of the corresponding complex cells from the previous layer, and a further inhibitory input, I, from an inhibitory cell which receives input from the same complex cells in the previous layer as shown in figure 3.7.7.

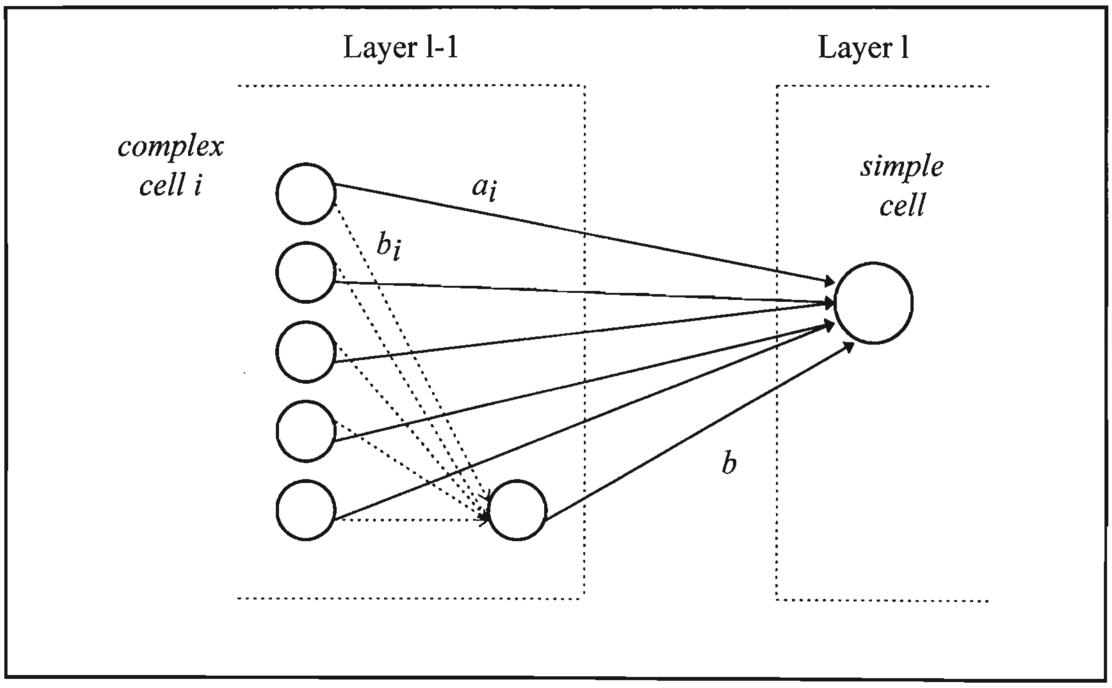


Figure 3.7.7 - The inhibitory and excitatory inputs of a simple cell.

The excitatory input is defined as:

$$E = \sum_i a_i u_i \, , \tag{3.7.1}$$

where a_i is the weight of the input connection to the ith complex cell and u_i is the output of the complex cell. The inhibitory input is defined as:

$$I = bv \, , \tag{3.7.2}$$

where b is the weight of the connection from the inhibitory cell and v is the output of the inhibitory cell defined as:

$$v = \sqrt{\sum_i (b_i u_i)^2}$$

with b_i being the weight of the connection between the *i*th complex cell and the inhibitory cell. The weights b_i are fixed so that they are strongest at the center of the connection region and decrease outwards. Also the sum of these weights must be one, that is, $\Sigma b_i = 1$.

The overall input, *net,* to the cell is given by:

$$net = \frac{1+E}{1+I} - 1$$

A threshold linear function is then applied to this input to provide the output of the cell:

$$out_S = \begin{cases} net & net > 0 \\ 0 & net \leq 0 \end{cases}$$

The complex cell fires if at least one cell in its input neighbourhood is active.

3.7.4 Training

The features detected by simple cells in each level are determined during training. The simple cells have variable input connections which are reinforced during training according to the feature that they are required to detect. Learning can be performed either with a teacher, that is supervised learning (Fukushima, 1988) where the features to be detected are provided by the teacher during training, or without a teacher, that is unsupervised learning (Wasserman, 1989) where the network self organizes to repetitions of a specific pattern and features are determined automatically by the network during training.

Training involves the changing of the excitatory weights a_i and the inhibitory weight b for each simple cell . The weights, b_i, of the input connections to the inhibitory cell, which are shown by the broken lines in figure 3.7.7, remain fixed.

Unsupervised training

Unsupervised training involves presenting an input pattern to the network and adjusting the weights of the inputs to the simple cells layer by layer, starting with those in U_{s1}, where the weights of the input connections from cells in U_0 are adjusted. The weight of the connection between a simple cell *S* and a complex cell *C* from the previous layer, is increased when *C* is responding, and *S* is responding more strongly than its immediate neighbours. S then serves as a representative for the other cells in its plane and their weights are adjusted in a similar way, thus ensuring that all cell in that plane detect the same feature but at different positions. Initially simple cells near each other might have similar outputs, but slight variations will cause one cell to fire more strongly than the others. The excitatory weights of that cell are strengthened while the inhibitory weights of the other cells are increased. In this way the weights of the inputs to simple cells are adjusted so that the cells respond only to patterns which are experienced frequently during training. Thus familiar patterns produce large outputs, while unfamiliar patterns result in smaller outputs.

Supervised training

Supervised learning is tedious in that it requires the teacher to supply the features for the simple planes in each layer of the network. The weights of the inputs to the simple cells are modified layer by layer starting with layer 1 through to the output layer. The simple planes in each layer are trained one at a time and the next layer is only trained if the previous layer is responding correctly. A representative cell, known as the seed cell, is chosen from the first plane in layer 1. The input pattern is then presented to the input image. The weights of the seed cell are modified repeatedly until it responds to the feature corresponding to the first plane. Since the other cells in the plane must respond to the same feature but at different positions, the weights of the inputs to the seed cell are used to determine the weights of their inputs. The second plane is trained similarly for a different feature. Once all the planes in the first layer are responding correctly to their corresponding features, the planes in the second layer are trained. This procedure is followed for each layer through to the output layer.

Unsupervised learning involves self organizing of the network and bears resemblance to biological visual systems as it does not require a teacher for learning. The main advantage of this method is that the decision of which features are to be detected by the simple planes, in each layer, is avoided. Supervised learning can be viewed more from a pattern recognition perspective, with the network trained for a specific application e.g. handwritten character recognition. Since the features, e.g. edges in layer 1, are supplied by the teacher, the network can be made more tolerant to distortions in the input pattern (Fukushima et al., 1983, Fukushima, 1988). However this can be tedious and time consuming. Fukushima & Wake (1992) have suggested an improved learning algorithm for the Neocognitron which combines both supervised and unsupervised learning. In this method the network automatically determines the features for each simple plane as carried out in unsupervised learning, but providing the target outputs for each training pattern as carried out in supervised learning. This method avoids the time consuming process of providing features for each plane, while still increases the network's tolerance to distortions of the input patterns.

3.7.5 Biological motivation

Each layer in the Neocognitron contains simple cells and complex cells which correspond to simple or lower-order hypercomplex cells, and complex or higher order hypercomplex cells in the biological visual system respectively. In the visual cortex of the brain, neurons respond selectively to local features of a visual pattern, such as lines and edges in a particular orientation. In the area higher than the cortex, cells exist that respond selectively to certain figures like circles, triangles, squares, or even human faces (Fukushima, 1988). The Neocognitron also has hierarchical structure as in biological visual systems since cells in higher stages are responsive to more complicated features, have a larger receptive field and are insensitive to shifts in position of the input patterns while cells in the lower stages are responsive to simple features and have smaller receptive fields.

3.8 SUMMARY

Seven feature extraction methods, viz. moment invariants, Zernike moments, Fourier descriptors, Gabor filters, wavelets and the Neocognitron have been described and table 3.4 summarizes the properties of these methods.

Moment invariants, Zernike moments Fourier descriptors and the Neocognitron are suitable for two dimensional shape recognition. The method of moment invariants is easier to implement than Fourier descriptors and Zernike moments as the former involves the calculation of the Fourier transform while the latter requires calculations using complex polynomials. The Neocognitron requires the largest amount of computations.

Gabor filters and wavelets are suitable for texture classification. However the computation of the Gabor filters of an image is difficult compared to wavelets which can be calculated using the multiresolution approach.

In the next chapter an image recognition application, that of handwritten character recognition, is used to illustrate the practical role of feature extraction.

Table 3.4 - Summary of feature extraction methods.

Methods	Origin/ Implementation	Calculation	Applications	Objectives of features	Dimension of feature set
Moment invariants	⇒ image processing ⇒ separate stage	simple to calculate	2-D shape recognition	invariant to: ⇒ scale ⇒ orientation ⇒ position	seven moments
Zernike moments	⇒ image processing ⇒ separate stage	calculations involve complex numbers	2-D shape recognition	invariant to: ⇒ scale ⇒ orientation ⇒ position	coefficients of Zenike moments usually 40 or 47
Fourier descriptors	⇒ image processing ⇒ separate stage	Fourier transform is needed, complex numbers involved	2-D shape recognition	invariant to: ⇒ scale ⇒ orientation ⇒ position	radial and rotational bins in Fourier distribution
Gabor filters	⇒ biologically inspired ⇒ separate stage	⇒ very complicated ⇒ difficult to calculate	texture classification	decomposes textured areas in image, simultaneously into different frequency and orientation ranges	coefficients of Gabor filters 72 for $\Delta\theta=30^\circ$ and frequency bandwidth increasing at 1.5 octaves
Wavelets	⇒ origins in biological vision as well as in image processing ⇒ separate stage	⇒ easier to calculate than Gabor filters	texture classification	decomposes textured areas in image, simultaneously into different frequency and orientation ranges, but not as efficiently as Gabor filters	four values at each resolution
Neocognitron	⇒ biologically inspired ⇒ within the network	⇒ complicated structure ⇒ massive computations	2-D shape recognition	invariant to: ⇒ scale ⇒ position ⇒ distortions	number of features correspond to the number of simple planes in each layer

4. IMPLEMENTATION

In the previous chapter, different feature extraction methods were described. These methods can be categorized according to the applications in image processing for which they are suited, namely, shape recognition and texture recognition. The application considered in this thesis is handwritten character recognition, which falls under the area of two dimensional shape recognition. The feature extraction methods in table 3.4 which are suitable for two dimensional shape recognition are Moment invariants, Zernike moments, Fourier descriptors and the Neocognitron. Zernike moments have been used for handwritten character recognition (Khotanzad and Lu , 1991; Perantois , 1992) and are less sensitive to noise than moment invariants (Teh & Chin 1988, Khotanzad and Hong 1990). The Neocognitron was originally designed to handle the distortions and variations in size and positions of handwritten characters (Fukushima et al., 1983).

Zernike moments and the Neocognitron were implemented in an handwritten character recognition system and compared to a handwritten character recognition system without any feature extraction. Zernike moments form the basis for a feature extraction method implemented as a separate stage in the system while the Neocognitron combines the feature extraction stage with a neural network classifier.

Figure 4.1. shows the overall structure of an handwritten character recognition system.

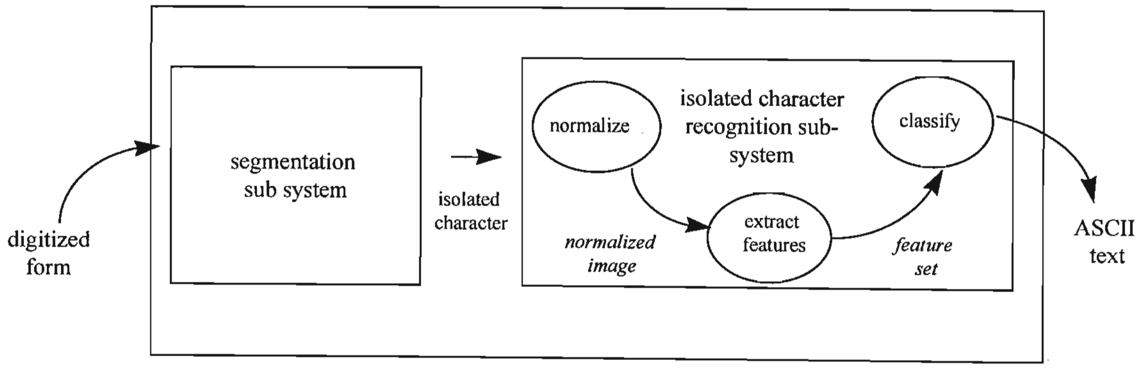


figure 4.1. - The components of a handwriting recognition system

The input to the system is a form. A form can be considered as a page with marked areas within which a person is allowed to write some text. A simple example is an A4 page with a centered rectangle within which the text can be written. A digitized form is the computer version of the form which can be stored electronically by the system. The physical form is usually fed into a device known as a digital scanner, which then produces the electronic equivalent, that is, a digitized form. The latter form is fed into the segmentation sub-system which analyzes the writing and splits up words into images of isolated characters to form the input to the isolated character recognition sub system. This system processes these images, decides which character is depicted in each image, and generates the appropriate ASCII characters as the system output. The recognition sub-system can be broken down further into three components, (1) a preprocessing stage in which the image is normalized with respect to size e.g. 32 x 32 pixel images are scaled down to 19 x 19 pixels, (2) the feature extraction stage in which important attributes or features are extracted from the image and (3) the classification stage in which the feature vector is used to categorize the image into one of the output classes, '0','1', '2', ..., '9'. The overall output of the system is a file containing ASCII text of the handwriting appearing on the original form.

Feature extraction serves two purposes, that of reducing the size of the input vector and that of making the system invariant to certain properties which are specific to each application.

Three factors can cause variations in handwritten characters. These are:

1. The size of the writing.
2. The position of the character in the image. It is highly improbable that all characters have been started at exactly the same position in the specified area on the form. The scanning and segmentation process may also result in variations in the position of the characters in the image.
3. The orientation of the characters. Variations occur in the orientation of the characters written by different writers, e.g. one writer may write with a slant to the right and another to the left. Furthermore the characters are acquired using a digital scanner so that when the scanning process is carried out the pages will not be at exactly the same

position on the scanner each time and a slight slant of the page affects the orientation of the characters .

For example the digit '7' must still be classified as a seven even if:

- the '7' covers the whole image or only a small portion of the image,
- the '7' occurs at the top of the image or at the bottom,
- the '7' is slanted either to the right or to the left.

Invariancy in the system can be accomplished in three ways. The first involves the data which is presented to the network. In particular all instances, involving orientation, size and translation, of a handwritten character can be used to train the network, thus allowing the network to be invariant to these properties. This method is not practically feasible as the training set can become extremely large. The second way involves using a feature extraction method to extract important features from the data, which means that the data will undergo some processing before reaching the network. This processing strips the data of parameters such as size, scale and orientation before it reaches the network. The third way to incorporate invariancy within the system is to incorporate it within the network architecture, that is to allow the network to extract appropriate features from the data.

Two methods of implementing invariancy are illustrated in this thesis, and these are compared to a system without invariancy in the following experimental applications:

Experiment I: system with no invariancy

The pixel values of digital images of handwritten digits are used to train and test a multilayered perceptron. This experiment illustrates image recognition without any distinct feature extraction stage.

Experiment II: separate feature extraction stage

The Zernike moments of the digital images are calculated and used to train and test a multilayered perceptron. This experiment illustrates the use of a distinct feature extraction stage.

Experiment III: invariancy within network architecture

The pixel values of the digital images are used to test and train the Neocognitron. This experiment illustrates the incorporation of the feature extraction process within the network architecture.

These applications are described in detail below. However it will be useful to first examine the general software engineering requirements of each of the applications to be developed (Sommerville, 1989).

4.1 GENERAL REQUIREMENTS DEFINITION FOR A CHARACTER RECOGNITION SYSTEM

4.1.1 Functional requirements

The system is required to perform the functions of the isolated character recognition subsystem depicted in figure 4.1. The input to the system is a digitized image of an isolated numeric handwritten character, that is [‘0’,.., ‘9’]. The image must be a binary image, that is the image must only contain two colours, black and white. The output of the system must be the corresponding ASCII representation of the character. A further requirement is to be able to evaluate the efficiency of the system.

4.1.2 Database requirements

A training set of data is required to train the system. This must consist of a database of handwritten digits from different writers. A test set is also required to evaluate the performance of the system. The test set must consist mainly of characters from writers who have not contributed to the training set.

4.1.3 Non-functional requirements

The non-functional requirement is generally the set of constraints imposed on a proposed system (Sommerville, 1989). In the present case the system is intended for research purposes and not for continuous use. As a result no constraints are placed on the speed, efficiency and effectiveness of the software. Since the emphasis of the research is on feature extraction rather than neural networks existing neural network software is used. In this respect software for the multilayered perceptron and the Neocognitron need to be evaluated.

4.2 DATABASE

A sample database, the *fl3* database, was obtained from the National Institute of Technology and Standards (NIST) in the United States of America. This database was used for the training and testing of the systems.

The *fl3* database consists of 3,471 isolated handwritten digits from 49 different writers. The images of the characters have been spatially normalized to be 32 X 32 pixels and are arranged according to the 49 different writers. The files are in the form of IHEAD raster images and C source code was included with the database to manipulate these images. Further code was written in C++ to convert these images into text format. Figure 4.2.1 shows an example of an image in text format with the characters '*' and '.' signifying active and inactive pixels respectively.

```
..................XXX...........
.................XXXX...........
................XXXX............
..............XXXXXXXX..........
.............XXXXXXXXXXX........
.............XXXXXXXXXXXX.......
............XXXXXXXXXXXXXX......
...........XXXXX..XXXXXXXXX.....
.........XXXXXX...XXXXXXXXXX....
.........XXXXX....XXXXXXXXXX...
.........XXXX.........XXXXXXX...
........XXXXX............XXXXX..
........XXXXX.............XXXX..
.......XXXXX..............XXXX..
......XXXXX...............XXXX..
.....XXXXXX...............XXXX..
....XXXXXX................XXXX..
....XXXXX................XXXXX..
...XXXXX.................XXXXX..
...XXXXX................XXXXX...
..XXXXXX................XXXXX...
..XXXXX.................XXXXX...
..XXXXX.................XXXX....
..XXXXX...............XXXXX.....
..XXXXX............XXXXXXX......
..XXXXX...........XXXXXXX.......
..XXXXXX........XXXXXXXX........
...XXXXXX...XXXXXXXXXXX.........
....XXXXXXXXXXXXXXXXX...........
.....XXXXXXXXXXXXX..............
......XXXXXXXXXX................
.............X..................
```

Figure 4.2.1 - An example of an handwritten digit from the fl3 database in 32 x 32 text format.

4.3 EXPERIMENT 1: THE MULTILAYERED PERCEPTRON

The multilayered perceptron (MLP) is a three layered neural network architecture which can be trained using the backpropagation algorithm. In this experiment the MLP was implemented to recognize handwritten digits and the overall structure of the system is shown in figure 4.3.1.

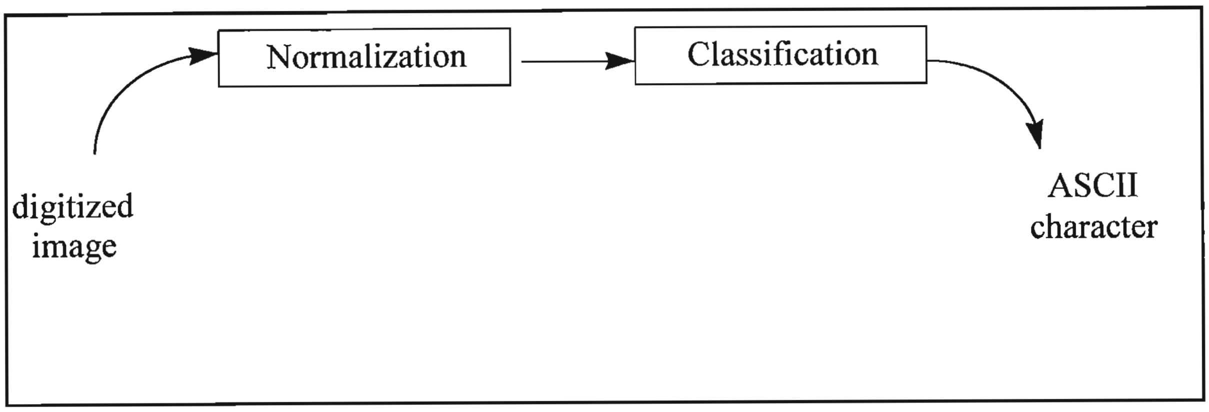


Figure 4.3.1 - Experiment 1 - handwritten digit recognition with no feature extraction.

Several software packages were evaluated to implement the MLP and a selection of these are described in the next section.

4.3.1 Software

<u>Selection</u>

Two commercially available packages, BrainMaker 2.5 and NeuralWorks Explorer, and several public domain packages were evaluated. The software package BrainMaker 2.5 was selected for implementing the multilayered perceptron in this experiment.

BrainMaker was chosen over the other packages because:

1. BrainMaker is a commercial package and thus enjoys the usual advantages over public domain packages in that it has extensive but simple documentation in the form of manuals, it is less prone to crashes and to bugs, and it has been extensively tested.
2. BrainMaker was found to be easier to learn and use then the other commercial package examined, NeuralWorks Explorer. The latter in fact caters for other network architectures besides the multilayered perceptron.
3. NeuralWorks Explorer has a limit of 150 on the number of neurons in the network while BrainMaker can accommodate networks of more than 1500 neurons.

BrainMaker 2.5 - The multilayered perceptron

BrainMaker 2.5 is a commercially available software package developed by California Scientific Software in 1992 and can be used for designing, building, training, testing and running neural networks. The network structure used by BrainMaker 2.5 is exclusively that of the multilayered perceptron network architecture described in Chapter 2.

BrainMaker has the following limitations on the size of the network:

1. The maximum number of neurons in any layer is 512
2. The maximum number of layers allowed in the network is 8.

The limitation on the number of layers has no effect on the experiments as the multilayered perceptron comprising of only three layers, the input layer, the hidden layer and the output layer is used here. However the limitation on the number of neurons in a layer does pose some problems. In particular, the number of neurons in the input layer is limited to 512. If the pixel values of digital images are used as input to the network then the largest image that can be handled by BrainMaker is a 22 x 22 pixel image, resulting in an input vector of size 484 which falls below the 512 limit. As a result the 32x32 images in the *fl3* database were normalized to 19x19 pixels. The choice of normalizing to 19x19 pixels resulted also from the need for consistency in the three experiments especially experiment III, the neocognitron, where 19x19 images are normally used.

4.3.2 Normalization

A program was written in C++ (appendix I) to normalize the 32x32 images in the *fl3* database to 19x19 pixels. Firstly the image was normalized to 16 x 16 pixels using the simple averaging technique summarized in equation 4.1 (Schalkoff, 1989).

$$f_{16}(i,j) = \frac{1}{4}\sum_{p=0}^{1}\sum_{q=0}^{1} f_{32}(2i+p, 2j+q) \ , \qquad i, j = 0,\ 1,\ 2,\ ..,\ 18. \qquad (4.1)$$

In particular, the image was divided into 2 x 2 windows to give 16 x 16 windows which completely cover the image. Each pixel in the new image corresponded to a 2 x 2 window in the original image with brightness the average of the brightness of the 4 pixels in its corresponding 2 x 2 window.

To convert the image from 16 x 16 pixels to 19 x 19 pixels, 3 rows and 3 columns of inactive pixels were added to the image as follows: 1 row at the top, 2 rows at the bottom, 1 column at the left and 2 columns at the right. Figure 4.3.2 depicts the image shown in figure 4.2.1 normalized to 19 x 19 pixels.

```
...................
...................
...........XX......
.........XXXX......
........XXXXXX.....
........XXXXXXX....
......XXX..XXXXX...
......XXX....XXXX..
......XX.......XX..
.....XXX.......XX..
....XXX........XX..
...XXX........XXX..
...XXX........XXX..
...XXX.......XXX...
...XXX.....XXXX....
...XXX..XXXXXX.....
....XXXXXXXX.......
.....XXXXX.........
...................
```

Figure 4.3.2 - 19 x 19 normalized image of the digit '0' in figure 4.2.1.

The overall structure of the normalization process is summarized in figure 4.3.3.

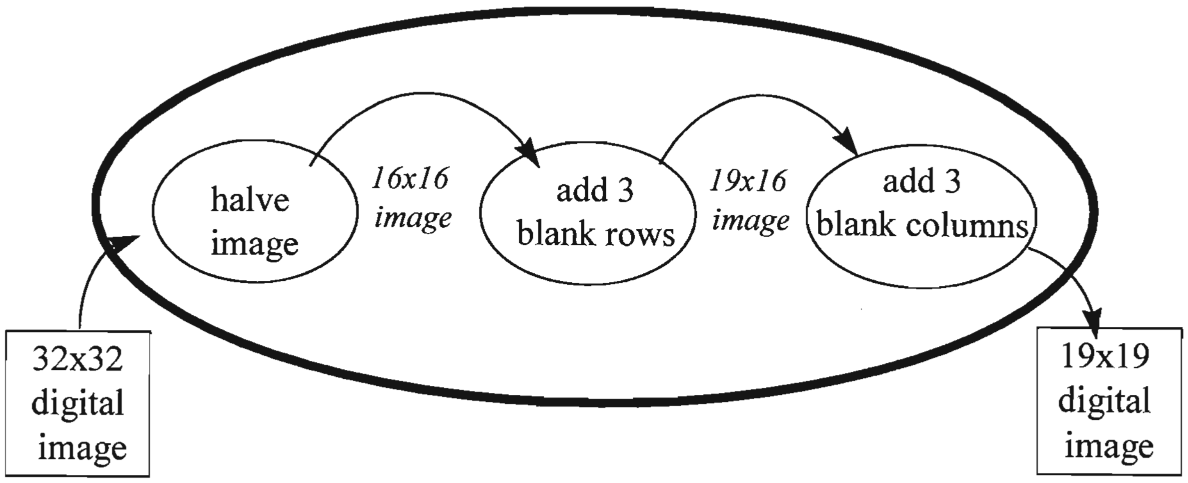


Figure 4.3.3 - The normalization process used to scale down the 32x32 pixel images from the fl3 database to 19x19 pixels.

The isolated characters provided in the *fl3* database were obtained by segmenting them from words and in some cases the segmentation was not completely successful. As a result parts of other characters occur in certain of the images. This is illustrated in figure 4.3.4 which shows a small selection of images from the *fl3* database. Notice also the extent of the distortions of some of the characters.

4.3.3 Partitioning of the database

Data is required for both the training and testing of neural network architectures, and thus the data must be partitioned into a training set and a test set. There are three ways in which this can be done:

1. Split the data so that for each character in the test set, there exists characters in the training set from the same writer. Thus at least one set of characters from all 49 writers are in the training set. In this case, the classification performed would be writer dependent, as the test set will contain writing styles on which the network has been previously trained. The accuracy of recognition would be expected to be high.
2. Split the data according to the writers, e.g. the characters from 40 writers in the training set and the characters from the remaining 9 writers in the test set. The classification performed here would be writer independent, that is, the writing styles on which the network is being tested have not been processed by the network. This would require the network to be less sensitive to slight distortions and slight variations in the orientation of the characters.
3. Split the data so that the test set contains characters from both writers who have contributed to the training set, as well as characters from writers who have not contributed to the training set, e.g. 41 writers used for the training set, 7 writers used for the test set and the last writer will contribute to both the sets.

In this study method 3 was selected.

```
...................                  ...................
...................                  ...................
..XX........XX.....                  ..........XXX......
..XX........XX.....                  ...........XXXX....
..XX........XX.....                  .............XXX...
..X........XXX.....                  ..............XX...
..X........XXX.....                  .............XX....
..X........XXX.....                  ...........XX......
..........XXXX.....                  ..........XX.......
.........XXXXX.....                  ........XXXXXXXX...
.........XXXX......                  ...................
.........XXXX......                  ...................
........XXXX.......                  ...................
........XXXX.......                  ..............XX...
.......XXXX........                  .............XX....
......XXXX.........                  ...........XXX.....
......XXXX.........                  ......XXXXX........
.......XX..........                  ...XXXXXX..........
...................                  ...................
1                                    3

...................                  ...................
...................                  ..........XXXX.....
......XXX.X...XX...                  .........XX...X....
.....XXXXXXXXXXXX..                  ........X..........
.....XX...XXXXXX...                  ........X.....X....
....XX......XXXX...                  ........X...XX.....
...XXX......XXX....                  .......XX...X......
...XXX......XX.....                  .......XX.XX.......
....X......XXX.....                  ....XXX............
..........XXX......                  ....XX.............
.........XXX.......                  ...XXX.............
........XXX........                  ..XX.X.............
........XXX........                  ..X..XX..........X.
......XXXX.........                  ..X..XX..........X.
......XXX..........                  ..X..XX..........X.
.....XXXX..........                  ..X..X.............
.....XXX...........                  ...XX..............
.....XX............                  ...................
...................                  8
7

...................                  ...................
...................                  ...................
...........XX......                  .........XXXX......
..........XXX......                  .......XX....X.....
.........XX........                  ......XX.....X.....
........XX.........                  .....XX.....X......
.......XXX.........                  .....X.....XX......
.......XX...X......                  .....X...XXX.......
.......XXXXXX......                  .....XXXX.X........
.......XXXXXX......                  ......X..XX........
.........XXXX......                  .........X.........
..........XXX......                  ........X..........
..........XXX......                  .......XX..........
.........XXX.......                  .......X...........
.........XX........                  ......X.....XX.....
........XXX........                  .....X........X....
........XX.........                  .....X.............
........XX.........                  ....X..............
...................                  ...................
9                                    9

...................                  ...................
...................                  ...................
........XXXXX......                  ..............XXXX.
.........XXXXXXXX..                  ......XXX...XXXXXX.
..........XXXXXXXX.                  .....XXXX..XXXXXXX.
................XX.                  ....XXXXX..XXXX.XX.
................XX.                  ....XXXX...........
...............XXX.                  ...XXXXX...........
............XXXXX..                  ..XXXXX............
..........XXXXXX...                  ..XXXX.............
.........XXXXXXXX..                  ..XXXX.............
......XXXXXXX......                  ..XXX............X.
....XXXXXXXX.......                  ..XXX...........XX.
...XXXXXXX.........                  ..XXX..........XXX.
..XXXXXX...........                  ..XXXXXXXXXXXXXXXX.
..XXXX.............                  ..XXXXXXXXXXXXXXXX.
...................                  ....XXXXXXXXXXXXX..
...................                  .....XXXXXXXXXX....
2                                    ...................
                                     0
```

Figure 4.3.4 - 8 normalized 19 x 19 images from the fl3 database. Notice the segmentation error in the first image and the high degree of distortion evident in the other images.

4.3.4 Training and testing the MLP

The pixel values of the digital image formed the input to the network. Thus the input layer contained 361 neurons corresponding to the 19 x 19 pixels in the digital image. Since the images are binary images, that is, the pixels were either '*' (active) or '.' (inactive), each neuron in the input layer received either 0 ('.') or 1 ('*') from their corresponding pixel in the input image. The output layer contained 10 neurons corresponding to the digits '0'..'9'. On presentation of an input image, a single neuron in the output layer is required to output a a 1, indicating the digit which is depicted in the input image. However in practice the output neurons do not output either a 0 or a 1. Usually all the neurons have values above 0 and below 1 and the neuron with the highest value indicates the output class.

<u>Data set I</u>

Using method 3 from section 4.3.3, the characters of the 49 writers in the *fl3* database were allocated to a training and a test set as follows:

- 41 writers contributed characters to the training set,
- 7 writers contributed characters to the test set,
- 1 writer one contributed characters to both the training and the test sets.

The above training and test set was labeled as data set I.

The size of the MLP used for data set I was :

- 361 neurons in the input layer corresponding to the 361 pixels.
- 80 neurons in the hidden layer.
- 10 neurons in the output layer corresponding to the 10 output classes '0' .. '9'.

The training set consisted of 2445 images of handwritten characters. The training tolerance was set to 0.1. This means that if the correct neuron outputs a value of more than 0.9 then the image is considered to be classified correctly. The training rate or learning coefficient, *l_coeff*, was set to 1. The network required 33 iterations to learn these images. The number of images that were correctly and incorrectly classified for each iteration is shown in table 4.1 and were used to plot the training curve of the network which is shown in figure 4.3.5. Notice the initial steepness of the learning curve which flattens out after the 10th run. In fact almost 90% of the images were learnt by the 10th run and only 300 images were learnt over the last 20 runs. These were probably images that were inconsistent as a result of distortions or noise and were therefore difficult to learn. For this experiment training ended when all the images in the training set were classified correctly.

Table 4.1 - Number of facts learnt at each iteration during training of the MLP on the fl3 data set.

Run	TotFacts	Good	Bad	% good	Tolerance	hh:mm:ss
1	2445	703	1742	28.75%	0.1	00:01:52
2	2445	1462	983	59.80%	0.1	00:03:04
3	2445	1692	753	69.20%	0.1	00:04:07
4	2445	1817	628	74.31%	0.1	00:05:06
5	2445	1897	548	77.59%	0.1	00:06:02
6	2445	1967	478	80.45%	0.1	00:06:55
7	2445	2067	378	84.54%	0.1	00:07:45
8	2445	2133	312	87.24%	0.1	00:08:33
9	2445	2135	310	87.32%	0.1	00:09:20
10	2445	2195	250	89.78%	0.1	00:10:05
11	2445	2243	202	91.74%	0.1	00:10:50
12	2445	2251	194	92.07%	0.1	00:11:33
13	2445	2291	154	93.70%	0.1	00:12:15
14	2445	2295	150	93.87%	0.1	00:12:57
15	2445	2328	117	95.21%	0.1	00:13:38
16	2445	2325	120	95.09%	0.1	00:14:19
17	2445	2321	124	94.93%	0.1	00:15:00
18	2445	2357	88	96.40%	0.1	00:15:39
19	2445	2378	67	97.26%	0.1	00:16:18
20	2445	2361	84	96.56%	0.1	00:16:58
21	2445	2401	44	98.20%	0.1	00:17:36
22	2445	2398	47	98.08%	0.1	00:18:14
23	2445	2412	33	98.65%	0.1	00:18:53
24	2445	2419	26	98.94%	0.1	00:19:30
25	2445	2431	14	99.43%	0.1	00:20:07
26	2445	2435	10	99.59%	0.1	00:20:44
27	2445	2439	6	99.75%	0.1	00:21:21
28	2445	2441	4	99.84%	0.1	00:21:58
29	2445	2442	3	99.88%	0.1	00:22:34
30	2445	2440	5	99.80%	0.1	00:23:11
31	2445	2443	2	99.92%	0.1	00:23:48
32	2445	2443	2	99.92%	0.1	00:24:24
33	2445	2445	0	100.00%	0.1	00:25:01

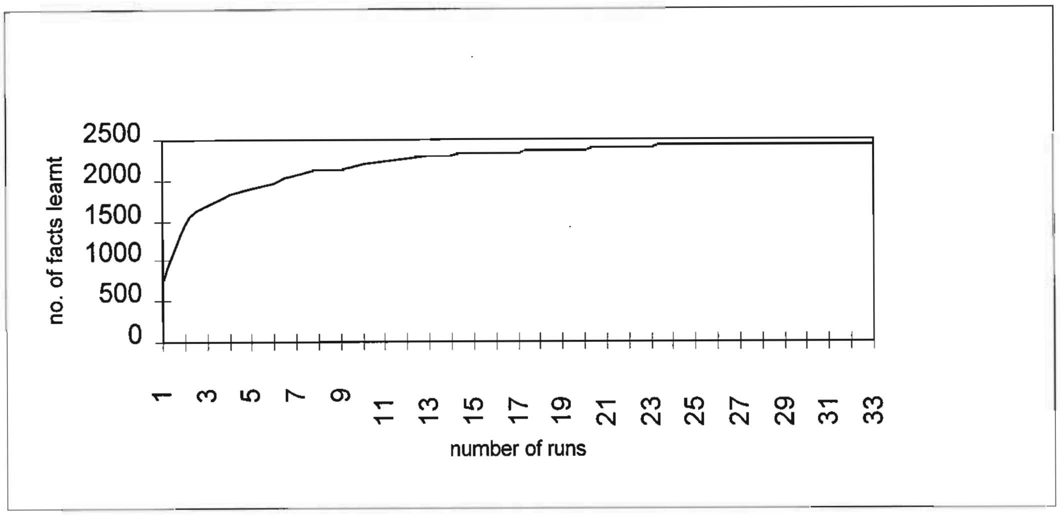


Figure 4.3.5 - Learning curve for the MLP on fl3 data set: number of correctly learnt facts (y axis) versus number of runs (x axis) from table 4.1.

The performance of the network was evaluated by presenting a test set to the network. The test tolerance was set to 0.4. This means that an image is deemed to be correctly classified when the correct neuron in the output layer, produced an output of at least 0.6. The testing set consisted of 872 images. These characters were from both writers who had contributed to the training set and from writers who had not contributed to the training set. Of these, 731 were correctly classified and 141 were not correctly classified. This gives an accuracy rate of 83.8%.

The multilayered perceptron thus performed extremely well with the characters from the *fl3* database. However it was noticed that these characters, a sample of which is shown in figure 4.3.6, vary very little in size, location and orientation.

<u>Data set II</u>

A second data set, data set II, was constructed. One hundred images, from two different writers, were taken from the *fl3* database. A further 20 characters from 2 other writers consisting of the digits '0' .. '9' , which are shown in Figure 4.3.7, were also taken.

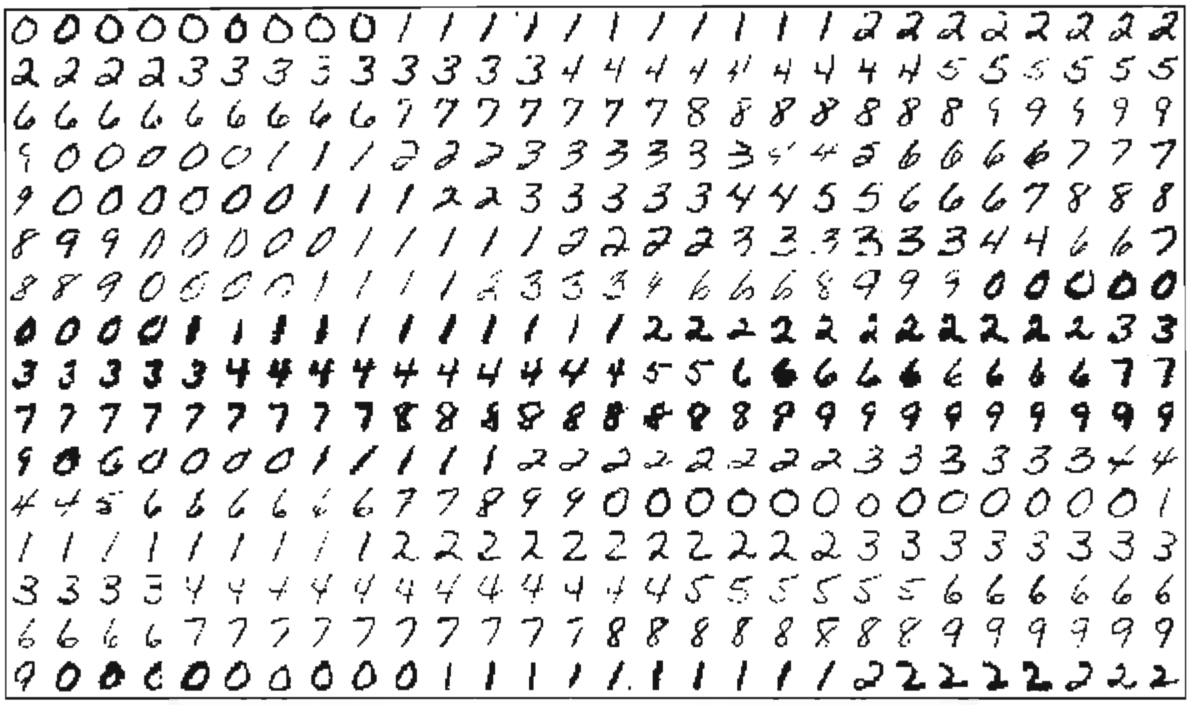


Figure 4.3.6 - Characters from the fl3 database which vary very little in size, shape and orientation.

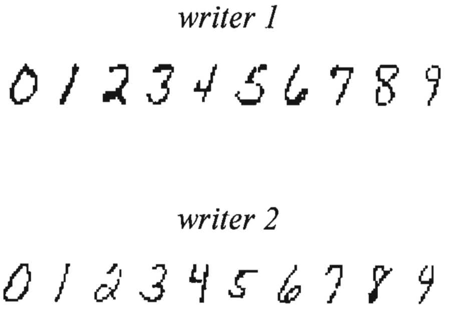


Figure 4.3.7 - Handwritten digits from two different writers.

These characters (figure 4.3.7) were rotated through angles of 90, 180 and 270 degrees and these rotated images were than combined with the other 100 images. The digit nine was excluded from the data set as it resembles a six on rotation through 180 degrees.

The resultant data set, data set II consisting of 169 images, is shown in random order in figure 4.3.8.

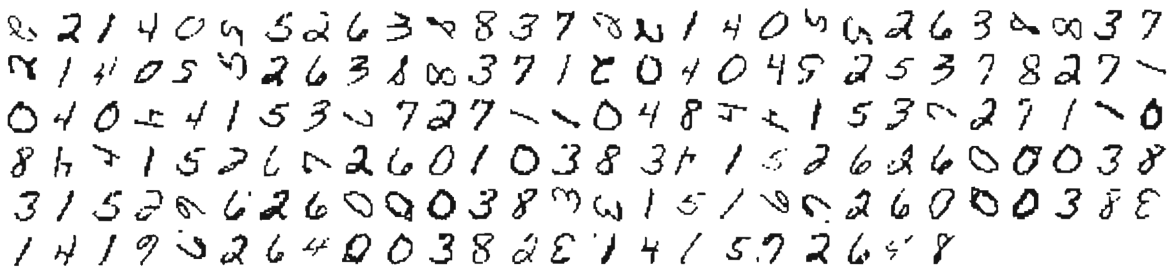


Figure 4.3.8 - Data set II containing normal as well as rotated digits.

This data set was divided into a training and test set. The first 139 images in figure 4.3.8 formed the training set and the remaining 30 were used as the test set. The network took 15 iterations to completely learn the training set at a training tolerance of 0.1. Table 4.2 shows the number of images learnt at each iteration and figure 4.3.9 shows the learning curve of the network.

Table 4.2 - Number of images learnt at each iteration during training of the MLP on data set containing rotated digits.

Run	Total	Good	Bad	% good	Learn	Tolerance	hh:mm:ss
1	139	1	138	0.7%	1	0.1	0:00:09
2	139	7	132	5.0%	1	0.1	0:00:14
3	139	22	117	15.8%	1	0.1	0:00:18
4	139	43	96	30.9%	1	0.1	0:00:22
5	139	62	77	44.6%	1	0.1	0:00:26
6	139	68	71	48.9%	1	0.1	0:00:29
7	139	96	43	69.1%	1	0.1	0:00:32
8	139	128	11	92.1%	1	0.1	0:00:34
9	139	130	9	93.5%	1	0.1	0:00:36
10	139	128	11	92.1%	1	0.1	0:00:38
11	139	135	4	97.1%	1	0.1	0:00:40
12	139	137	2	98.6%	1	0.1	0:00:42
13	139	138	1	99.3%	1	0.1	0:00:44
14	139	138	1	99.3%	1	0.1	0:00:46
15	139	139	0	100.0%	1	0.1	0:00:47

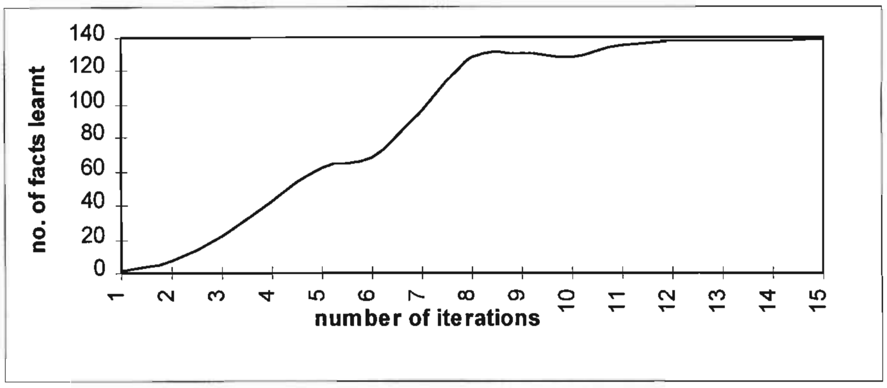


Figure 4.3.9 - Learning curve for the MLP on data set II, which contains the rotated digits: number of correctly learnt facts (y axis) versus number of runs (x axis) from table 4.2.

The testing tolerance was set at 0.4. Fourteen of the 30 images in the test set were classified correctly and 16 images were not classified correctly, resulting in an accuracy rate of 46.67%.

4.4 EXPERIMENT II: ZERNIKE MOMENTS

Figure 4.4.1 shows an image recognition system which includes a feature extraction stage. For the recognition of handwritten digits the method of Zernike moments is used.

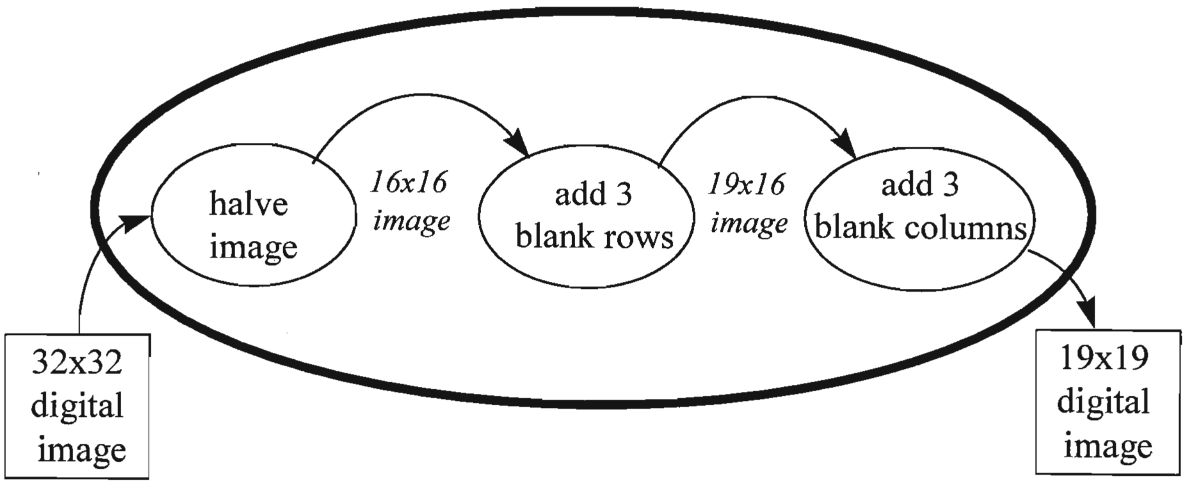


Figure 4.4.1 - Structure of an image recognition system that contains a feature extraction stage.

Zernike moments are used to remove rotational, scale and translational invariant features from digital images. Moreover Zernike moments reduce the size of the input vector to the network. For example, a 19 x 19 digital images which produces 361 input elements, the 361 pixels, can now be represented using the magnitudes of 47 Zernike moments. Furthermore these 47 values will remain constant even if the object, depicted in the image, is rotated, scaled or moved to a different position.

4.4.1 System design and implementation

Figure 4.4.2 depicts the steps involved to produce an invariant representation of a digital image. The image is first standardized with respect to scale and position, using equation (3.3.5) on page 30. The Zernike moments of the standardized image are then calculated. Since Zernike moments are invariant to rotational changes and calculated on the normalized image, the resultant representation is invariant to rotation as well as to scale and to position.

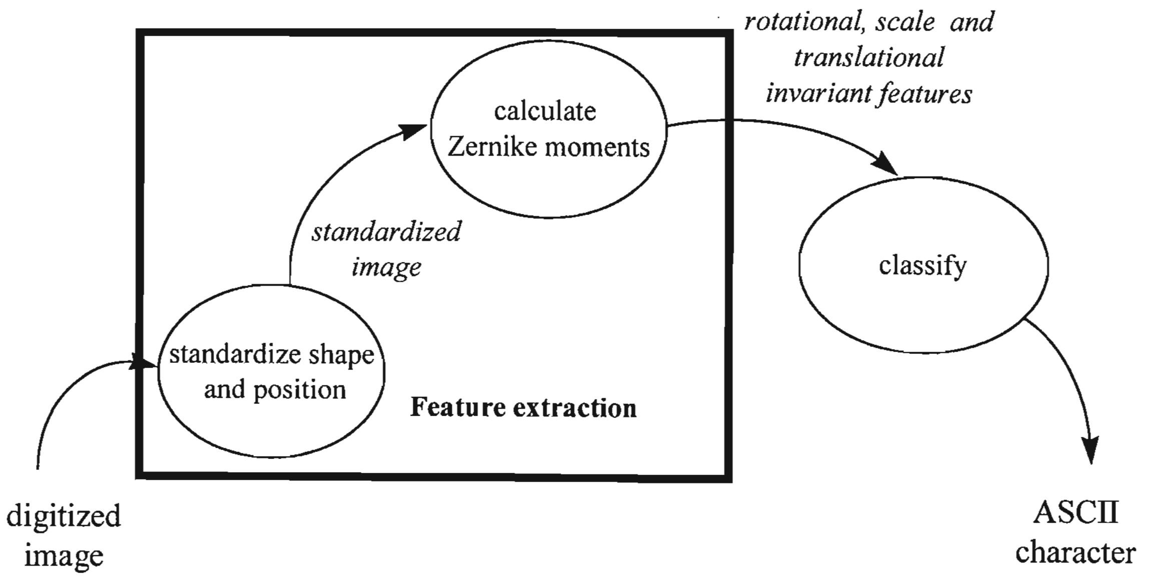

rotational, scale and translational invariant features
calculate Zernike moments
standardized image
classify
standardize shape and position
Feature extraction
digitized image
ASCII character

Figure 4.4.2 - General breakdown of feature extraction method of Zernike moments.

A program was written in Borland C++ (appendix II) to calculate the Zernike moments of a digital image. The input to the program, the digital image, was used to calculate the magnitudes of the 47 Zernike moments ($|A_{nm}|$) of order ranging from 2 to 12, which formed the output of the program. The 47 Zernike moments are depicted in Table 4.3 and the structure of the program used for their calculation is shown in figure 4.4.3.

Table 4.3 - The 47 Zernike moments, A_{nm}, used to represent an image.

Order	Moments
2	A_{20} ,A_{22}
3	A_{31}, A_{33}
4	A_{40} ,A_{42}, A_{44}
5	A_{51}, A_{53}, A_{55}
6	A_{60} ,A_{62}, A_{64} ,A_{66}
7	A_{71} ,A_{73}, A_{75}, A_{77}
8	A_{80}, A_{82}, A_{84}, A_{86}, A88
9	A_{91} ,A_{93}, A95 ,A_{97} ,A_{99}
10	$A_{10,0}$,$A_{10,2}$,$A_{10,4}$,$A_{10,6}$,$A_{10,8}$, $A_{10,10}$
11	$A_{11,1}$,$A_{11,3}$,$A_{11,5}$,$A_{11,7}$,$A_{11,9}$,$A_{11,11}$
12	$A_{12,0}$,$A_{12,2}$, $A_{12,4}$,$A_{12,6}$,$A_{12,8}$,$A_{12,10}$, A12,12

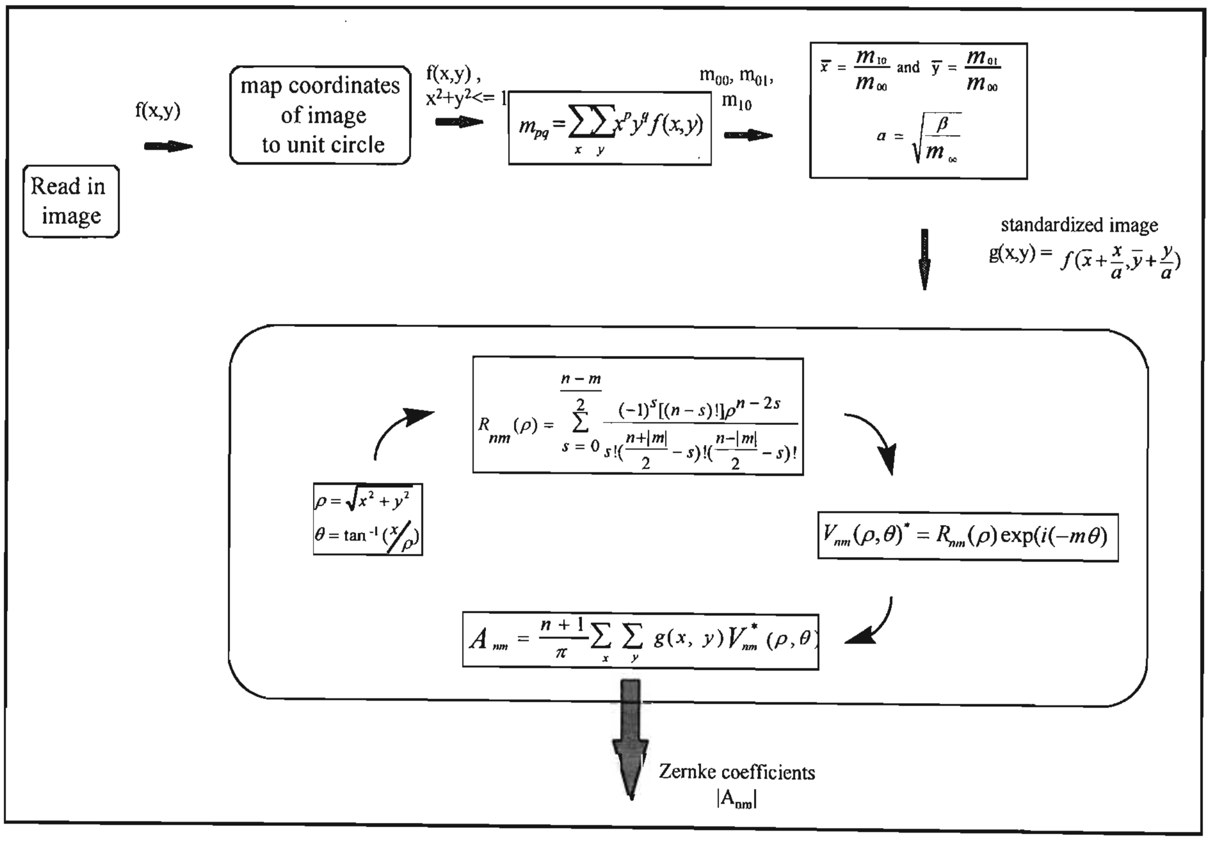

Read in image
f(x,y)
map coordinates of image to unit circle
f(x,y), $x^2+y^2<=1$
$m_{pq}=\sum_x\sum_y x^p y^q f(x,y)$
m_{00}, m_{01}, m_{10}
$\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$
$a = \sqrt{\frac{\beta}{m_{00}}}$
standardized image
g(x,y) = $f(\bar{x}+\frac{x}{a},\bar{y}+\frac{y}{a})$
$\rho=\sqrt{x^2+y^2}$
$\theta=\tan^{-1}(x/\rho)$
$R_{nm}(\rho)=\sum_{s=0}^{\frac{n-m}{2}}\frac{(-1)^s[(n-s)!]\rho^{n-2s}}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!}$
$V_{nm}(\rho,\theta)^*=R_{nm}(\rho)\exp(i(-m\theta)$
$A_{nm}=\frac{n+1}{\pi}\sum_x\sum_y g(x,y)V_{nm}^*(\rho,\theta)$
Zernke coefficients
$|A_{nm}|$

Figure 4.4.3. - Structure of program used to calculate the magnitude of the Zernike moments, $|A_{nm}|$.

4.4.2 Training and testing

The magnitudes of the Zernike moments were calculated for each of the 19 x 19 images used in experiment I. In this experiment these magnitudes were used as the input to the multilayered perceptron.

Data set I

The *fl3* data set was split, as in the first experiment using, method 3 in section 4.2.3, into a training set consisting of the magnitudes of the Zernike moments for 2 608 images and a test set consisting of the magnitudes of the Zernike moments for 739 images. The learning coefficient and the training tolerance were set at 1 and 0.1 respectively and the test tolerance was set at 0.4.

The size of the MLP was :

- 47 neurons in the input layer corresponding to the magnitudes of the 47 Zernike moments used to represent the image.
- 512 neurons in the hidden layer
- 10 neurons in the output layer corresponding to the 10 output classes '0' .. '9'.

Despite the additional hidden neurons, the network was not able to completely learn the magnitudes of the Zernike moments of the training set. The network was trained several times. The number of images learnt during training varied from 65% to 75% while the performance of the network on the test set varied from 45% to 55%. The number of iterations required during training ranged from 800 to over 2000. A possible explanation was that the network was having difficulty learning images depicting the digits '6'and '9' as they are similar at rotations through 180 degrees. All instances of the digit '9' were then removed from both the test and training sets and the number of neurons in the output layer was reduced to 9. However the performance of the network did not improve.

Data set II

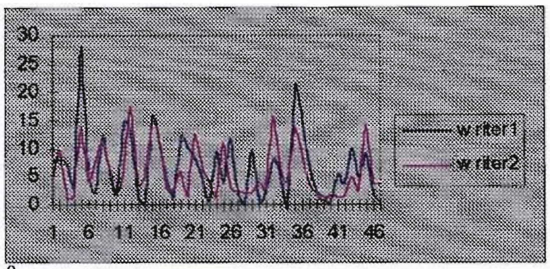
The network faired better on the magnitudes of the Zernike moments of the data set containing the rotated images. The magnitudes of the Zernike moments for each digit '0' .. '9' from the two writers, as shown in figure 4.3.7 on page 75, are depicted in figure 4.4.3. Each graph in figure 4.4.4 shows two instances of a digit, one by writer 1 and the other by writer 2.

To be consistent with the previous experiment, all instances of the digit '9' were removed from the data set, and the learning coefficient was set to 1, the training tolerance was set to 0.1 respectively and the test tolerance was set to 0.4.

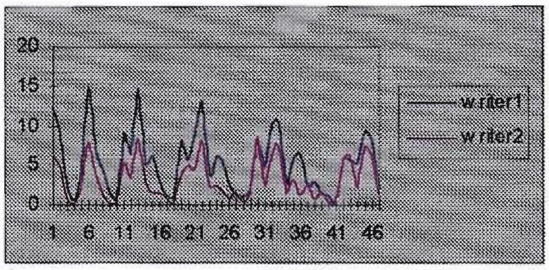
The size of the MLP was :

- 47 neurons in the input layer corresponding to the magnitudes of the 47 Zernike moments used to represent the image.
- 40 neurons in the hidden layer
- 9 neurons in the output layer corresponding to the 9 output classes '0' .. '8'

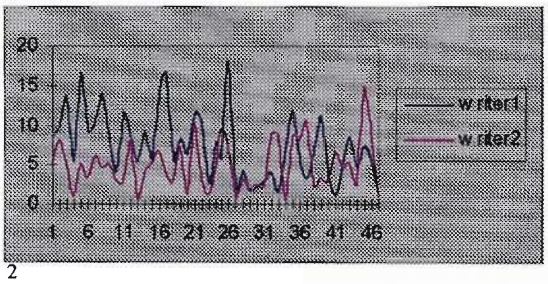
After 60 iterations the network had learnt all the images in the training set. The number of images learnt at each iteration is shown in Table 4.4 and plotted as the learning curve of the network in figure 4.4.5. The network performed well on the test set and classified 66.67% of the images correctly.



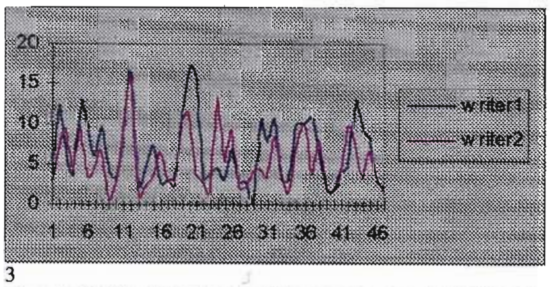
0



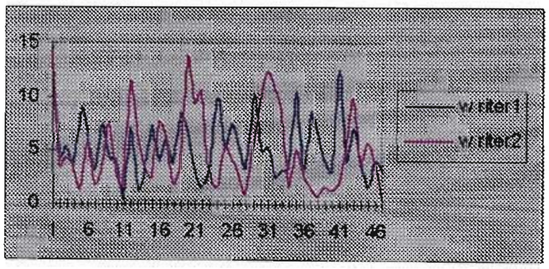
1



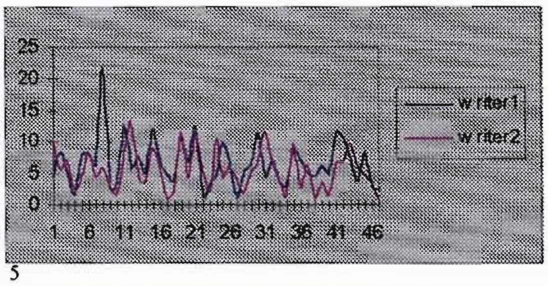
2



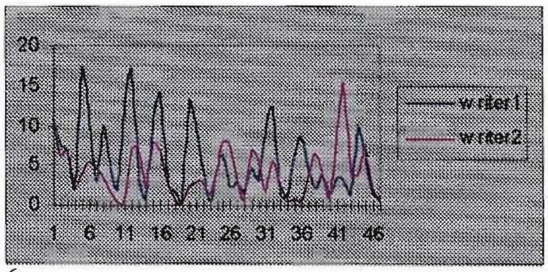
3



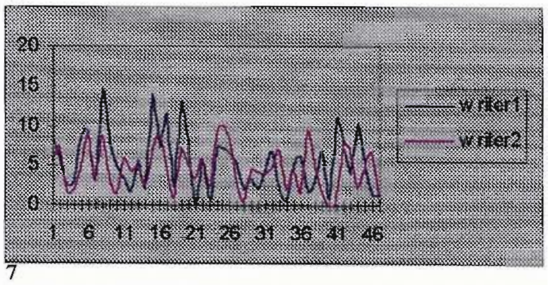
4



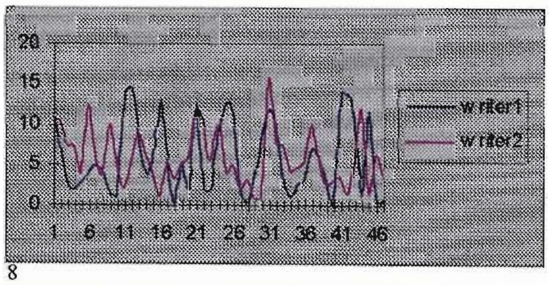
5



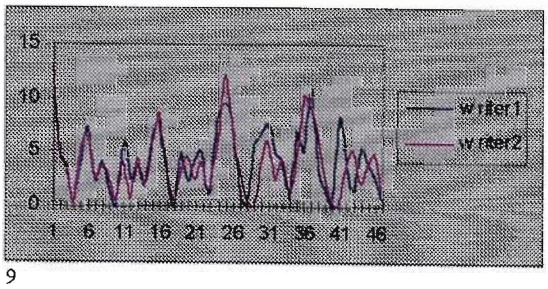
6



7



8



9

Figure 4.4.4 - coefficients of 47 Zernike moments of digits '0' - '9' from figure 4.3.7 on page 75.

Table 4.4 - Number of images learnt at each iteration using the Zernike moments of data set II, which contained the rotated digits.

Run	TotFacts	Good	Bad	Learn	Tolerance	hh:mm:ss
1	139	0	139	1	0.1	0:00:01
2	139	0	139	1	0.1	0:00:02
3	139	0	139	1	0.1	0:00:02
4	139	0	139	1	0.1	0:00:03
5	139	0	139	1	0.1	0:00:04
6	139	5	134	1	0.1	0:00:07
7	139	11	128	1	0.1	0:00:08
8	139	16	123	1	0.1	0:00:09
9	139	16	123	1	0.1	0:00:09
10	139	16	123	1	0.1	0:00:10
11	139	20	119	1	0.1	0:00:13
12	139	22	117	1	0.1	0:00:13
13	139	24	115	1	0.1	0:00:14
14	139	30	109	1	0.1	0:00:15
15	139	31	108	1	0.1	0:00:15
16	139	32	107	1	0.1	0:00:18
17	139	36	103	1	0.1	0:00:19
18	139	45	94	1	0.1	0:00:19
19	139	44	95	1	0.1	0:00:20
20	139	39	100	1	0.1	0:00:20
21	139	45	94	1	0.1	0:00:23
22	139	47	92	1	0.1	0:00:24
23	139	58	81	1	0.1	0:00:24
24	139	58	81	1	0.1	0:00:25
25	139	60	79	1	0.1	0:00:25
26	139	59	80	1	0.1	0:00:28
27	139	61	78	1	0.1	0:00:28
28	139	67	72	1	0.1	0:00:29
29	139	68	71	1	0.1	0:00:29
30	139	71	68	1	0.1	0:00:30
31	139	73	66	1	0.1	0:00:32
32	139	72	67	1	0.1	0:00:33
33	139	72	67	1	0.1	0:00:33
34	139	75	64	1	0.1	0:00:34
35	139	79	60	1	0.1	0:00:34
36	139	85	54	1	0.1	0:00:37
37	139	89	50	1	0.1	0:00:37
38	139	90	49	1	0.1	0:00:38
39	139	109	30	1	0.1	0:00:38
40	139	108	31	1	0.1	0:00:38

Run	TotFacts	Good	Bad	Learn	Tolerance	hh:mm:ss
41	139	107	32	1	0.1	0:00:41
42	139	121	18	1	0.1	0:00:41
43	139	128	11	1	0.1	0:00:41
44	139	125	14	1	0.1	0:00:42
45	139	131	8	1	0.1	0:00:42
46	139	136	3	1	0.1	0:00:44
47	139	134	5	1	0.1	0:00:44
48	139	134	5	1	0.1	0:00:45
49	139	138	1	1	0.1	0:00:45
50	139	138	1	1	0.1	0:00:45
51	139	134	5	1	0.1	0:00:47
52	139	138	1	1	0.1	0:00:47
53	139	138	1	1	0.1	0:00:48
54	139	138	1	1	0.1	0:00:48
55	139	134	5	1	0.1	0:00:48
56	139	137	2	1	0.1	0:00:50
57	139	138	1	1	0.1	0:00:51
58	139	136	3	1	0.1	0:00:51
59	139	137	2	1	0.1	0:00:51
60	139	139	0	1	0.1	0:00:52

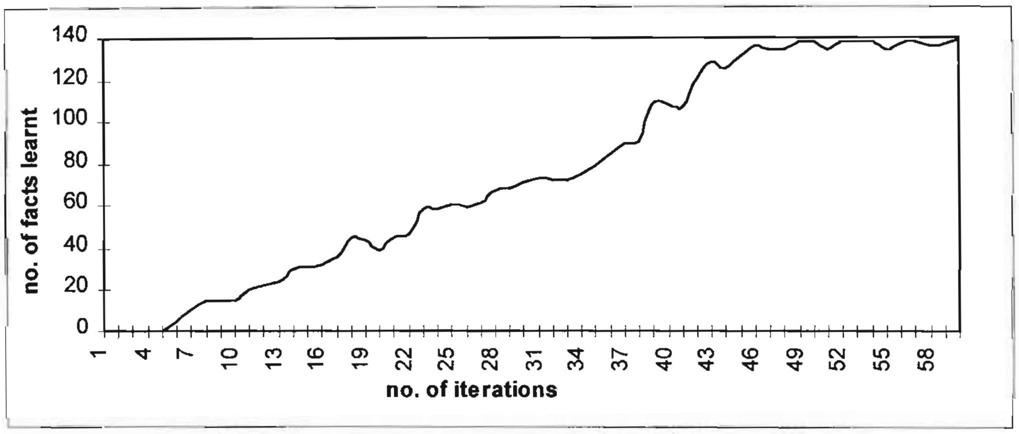


Figure 4.4.5 - Learning curve for the MLP on the Zernike moments of data set II which contained the rotated digits: number of correctly learnt facts (y axis) versus number of runs (x axis) from table 4.4.

4.5 EXPERIMENT III: THE NEOCOGNITRON

Figure 4.5.1 shows an image recognition system which incorporates the feature extraction process within the network.

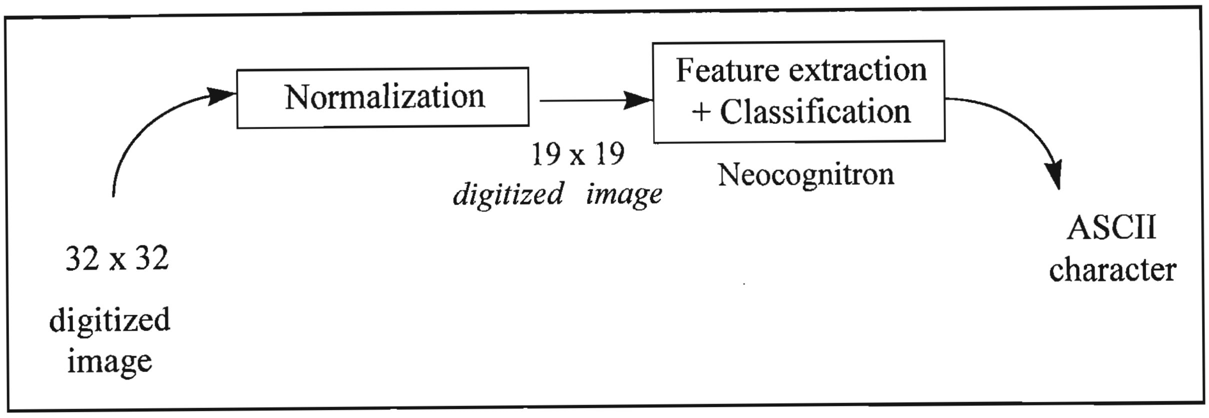


Figure 4.5.1 - Structure of an image recognition system which uses the Neocognitron.

The Neocognitron was implemented to illustrate the incorporation of the feature extraction process within the network architecture. The Neocognitron is a multilayered neural network architecture which is invariant to changes in scale, position and distortions in the input patterns.

4.5.1 Design and implementation

Two freeware software packages were evaluated for implementing the Neocognitron. These were:

- NeoC Explorer Version 1.0 by Szabolcs Szakacsits (Szakacsits, 1994).
- Fukushima's Neocognitron by Frank Schorrenberg (Schorrenberg, 1992).

NeoC Explorer has a simple but friendly graphical user interface designed solely for testing and analysis. However no modifications of the program can be made as the source code was not available. For Fukushima's Neocognitron, the C source code is supplied with the software and the program can be modified, but the program has no user interface and all instructions are supplied on the command line. Furthermore the software does not incorporate inhibitory cells in the network. NeoC Explorer was chosen over Fukushima's Neocognitron for implementing the Neocognitron.

The network used in this experiment consisted of 5 layers, the input layer U_0, U_1, U_2, U_3 and the output layer U_4. The number of planes in each sub layer, the number of cells in each of these planes and the receptive field of the cells in each layer are shown in Table 4.5. It must be noted that the structure of the Neocognitron is determined by the nature of application and the size of the input images. The structure used here is based on that of Fukushima (Fukushima et al., 1983).

Table 4.5. - Structure of the Neocognitron used in the experiment.

layer	**sub layer**	**no. of planes**	**no. of cells in plane**	**receptive field**
input - U_0	C_0	1	19x19	-
U_1	S_1	20	19x19	5x5
	C_1	20	15x15	5x5
U_2	S_2	20	13x13	5x5
	C_2	20	11x11	5x5
U_3	S_3	20	9x9	5x5
	C_3	20	7x7	5x5
output - U_4	S_4	10	3x3	3x3
	C_4	10	1x1	3x3

The pixel values of the 19x19 digital images are used as the input layer U_0. The output layer consists of 10 planes with each plane containing a single cell. These 10 cells represent the output classes '0 .. '9' and usually take on values between 0 and 1, with values close to 1 indicating an active cell and in this way the output class of the image.

The training set consists of representatives of each of the 10 output classes, that is, 10 images depicting the digits '0' .. '9'. Training consisted of repeatedly presenting each digit to the network until the network was responsive to that digit. For example, suppose the digit '0' was presented to the network, then '0' will be presented repeatedly to the network until it causes the cell in the first plane in the output layer to respond. Training is completed when all 10 digits invoke responses from their corresponding cells in the output plane, U_{C4}. Testing involved the presentation of images to the input layer U_0 and monitoring the corresponding response from the cells in U_{C4}. If the correct cell responded then, the character had been recognized correctly.

4.5.2 Training and testing

Data set I

As the Neocognitron requires only 10 images in the training set, the training set used for data set I in experiments 1 and 2 was not used for training. The 10 images shown in figure 4.5.2 were used to train the Neocognitron.

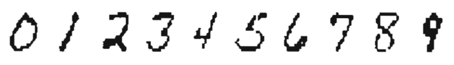


Figure 4.5.2 - 10 images used to train the Neocognitron for data set I.

The test set consisted of 100 of the 739 images used in the test set in experiments I and II and contained 10 images of each digit. The Neocognitron trained network classified 40% of the images in the test set correctly, 52% were classified incorrectly and the network was undecided for the remaining 8%, that is more than one output cell was active.

Data set II

For data set II the digit '9' was omitted from the training set which is shown in figure 4.5.3. The structure of the network was not changed, but U_{c4} in the output layer consisted of only 9 cells, corresponding to the digits '0' .. '8'.

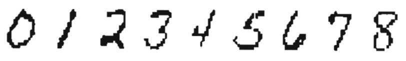


Figure 4.5.3 - 9 images used to train the Neocognitron for data set II.

The test set consisted of 36 images, 4 images of each digit, of which 25 were taken from the test set used in experiments I and II for data set II. The trained Neocognitron network classified 28% of the images in the test set correctly, 57% were classified incorrectly and the network was undecided for the remaining 15%.

4.6 SUMMARY OF RESULTS

Table 4.6 and table 4.7 summarize the performance of the three experiments on data set I, and data set II respectively. The accuracy rate for the Neocognitron on the training set is not relevant as the network only requires a single representative of each digit in the training set, in data set I the digits '0 .. '9' and data set II the digits '0' .. '8'.

Table 4.6 - Performance of the three experiments on data set I.

Experimental application	Accuracy rate on training set	Accuracy rate on test set
I. MLP	100%	83.8%
II. Zernike moments	55-65%	50 - 60%
III. Neocognitron	-	41%

Table 4.7 - Performance of the three experiments on data set II.

Experimental application	Accuracy rate on training set	Accuracy rate on test set
I. MLP	100%	47%
II. Zernike moments	100%	67%
III. Neocognitron	-	28%

The MLP was the easiest to implement. Even though this implementation was used to illustrate a system without any feature extraction, a normalization procedure (Schalkoff, 1989) had to be used to scale down the images for use with BrainMaker. This may be considered to some extent to be a form of feature extraction. The MLP performed surprisingly well on data set I, but not as well on data set II which contained the rotated images.

Zernike moments required some effort to calculate. However variations in scale, position and orientation were eliminated and the dimension of the input data reduced, thus reducing the number of neurons in the input layer from 361 to 47. Although the maximum number of hidden neurons, i.e 512, was used for data set I, the network was not able properly learn the training set and did not fair as well as the MLP on the test set. This can be attributed to the loss of discriminatory information resulting from the reduction of the input data from 361 elements to just 47 elements. However Zernike moments are especially useful when the input images are really large, e.g. 32 x 32 pixels. In this instance BrainMaker was not able to handle the images because of the 512 neuron limit on the input layer, but the Zernike moments of the image could be calculated and used to represent the images. Zernike moments performed better than the MLP on data set II, which illustrates the inability of the MLP to be cope with certain irrelevant properties of the input data.

The Neocognitron produced disappointing results on both data sets I and II. However, It was noticed that the network was undecided for a larger proportion, 15%, of data set II than the 8% of data set I. The network is not invariant to rotation and thus it was expected that the network would be confused with the rotated images in data set II. Overall, the Neocognitron can be considered as a highly complex research model which needs to be further investigated in order to produce better results:.

5. EVALUATION AND CONCLUSION

This thesis has investigated the area of digital image recognition. A typical image recognition system involves three stages:

1. the pre-processing stage
2. the feature extraction stage
3. the classification stage.

Image recognition tasks are performed naturally by biological systems and the architecture of these systems have been emulated by artificial neural networks. It is therefore appropriate to use an artificial neural network, more specifically the multilayered perceptron, for the classification stage.

The artificial neural network which was discussed in chapter 2 performs well under noise and distortions of the image, but is sensitive to other parameters e.g. size, position and orientation of the image. These parameters are often redundant and provide no discriminatory information to the classifier. Furthermore, variations in these parameters may cause confusion in the classification stage. The feature extraction stage removes these parameters from the image and only 'extracts' important features from the image.

A selection of feature extraction methods were reviewed in chapter 3:

<u>Moments invariants</u>
Moment invariants provide a positional, size and orientation invariant representation of an image. They have been used successfully for shape recognition and provide a representation consisting of seven features which are calculated from the geometric moments of an image. A significant disadvantage is that there is no measure of how good a representation of the original image these moments provide.

<u>Zernike moments</u>
Zernike moments have been used previously for shape recognition tasks. The representation consists of 47 positional, size and orientation invariant features which are

the magnitudes of the complex valued Zernike moments of a digital image. The original image can be reconstructed from these moments, thus giving an indication of the quality of the representation.

Fourier descriptors

Fourier descriptors provide a positional, size and orientation invariant representation of a digital image. This method has been investigated for 2-D aircraft recognition (Kulkarni, 1994). New ways in which to use Fourier descriptors for image representation are currently being investigated (Kauppinen & Seppanen,1995).

Gabor filters

Gabor filters operate similarly to structures in biological systems, that is, they categorise areas of an image into different ranges of orientations and spatial frequencies. It is one of the few representations that can simultaneously distinguish between orientations and spatial frequencies which is important for texture classification. However the calculation of the Gabor filters for a digital image is difficult.

Wavelets

The wavelet representation uses the differences between images at different resolutions to represent a digital image. This representation also, but to a lesser extent, is able to simultaneously distinguish areas of the image according to different orientations and spatial frequencies. Calculation of this representation is less difficult than for Gabor filters.

The Neocognitron

The Neocognitron is a multilayered neural network architecture, based on the feed forward architectures in biological systems, which combines the feature extraction and classification stage. The Neocognitron is insensitive to variations in position and distortions in the input image. This architecture has been used predominantly for shape recognition tasks.

It is sometimes necessary to introduce a pre-processing stage. The nature of the pre-processing is dependant on the type of images to be classified and addresses the practical limitations of the feature extraction and classification stages. An example of such a limitation is the dimension of the images to be classified. If the feature extraction or the classification stage can only accept images of a certain size, then the pre-processing stage will scale all images entering the system to this size.

To illustrate the practical significance of feature extraction, an application for handwritten digit recognition was implemented. Since handwritten characters can be distinguished by differences in their shapes, the feature extraction methods that are applicable, from table 3.4 on page 60, are moment invariants, Zernike moments, Fourier descriptors and the Neocognitron. Three experimental applications were developed to illustrate the role of feature extraction in handwritten digit recognition. The results of these experiments are discussed below.

5.1 THE MULTILAYERED PERCEPTRON

The multilayered perceptron was implemented in experiment I and represented a system with no feature extraction.

5.1.1 Strengths

The MLP is easy to implement. The pixel values of the digital images are used as input to the network. As a result no modification of the image occurs. Thus the best representation of the image, the image itself, is used as the feature vector. Since no processing is carried out on the image the chances of introducing noise or corrupting the image are reduced. The MLP performed very well on the data set I, which contained images which varied little in orientation, position and size of the characters. The MLP required 33 runs to completely learn the training set. Considering that the network had to learn 2445 images from different writers the MLP did well to learn the entire training set in only 33 runs. Moreover, the accuracy rate of 83.8% on the test set was good, despite the fact that this set consisted mostly of characters from different writers.

5.1.2 Weaknesses and limitations

A significant limitation of the MLP is the limitation on the number of neurons in the input layer. This may be overcome by scaling down the image as was done in experiment I. Since the original images were 32x32 pixel images, the image was scaled down by a factor of 2. Scaling down of images causes loss of information in the image. In experiment I this was not serious since the image was only scaled down by a factor of 2. However if the original image had comprised of 400x400 pixels, the scaling process would have resulted in a significant loss of information which may have been needed during classification to discriminate between patterns.

The MLP performed poorly for data set II which contained the rotated digits. The different orientations of the digits confused the network and illustrated the inability of the MLP to be invariant to certain irrelevant properties of the input data, in particular the orientation of the digits in data set II.

5.2 ZERNIKE MOMENTS

The feature extraction method of Zernike moments was implemented in experiment II and represented a system with a separate feature extraction stage.

5.2.1 Strengths

Zernike moments form an orthogonal set over the unit circle and therefore there is no information redundancy in the representation which makes it possible to reconstruct the original image from the representation. The reconstructed image can then be compared to the original image and the quality of the representation can be appraised. During reconstruction it is possible to identify the contribution of each Zernike moment to the image. This is important in deciding how many moments to use to represent the image, that is, if the reconstructed image is close enough to the original then the number of moments used is sufficient to represent the image. Zernike moments have been shown to perform better than moment invariants in the presence of noise (Teh & Chin 1988, Khotanzad and Hong 1990).

This method performed better on data set II than the MLP in experiment I and confirms its ability to extract rotational invariant features from the images. Zernike moments not only produce an invariant representation of an image which simplifies the classification task, but also significantly reduces the size of the feature vector. Thus the additional pre-processing stage used in the experiments to normalise the images to 19x19 pixels is not necessary.

5.2.2 Weaknesses

The reduction in the size of the input images results in the loss of information which may have been important during classification. This was apparent for data set I, where the MLP performed better than Zernike moments.

The calculation of the Zernike moments for an image requires a large number of complex operations. As a consequence, unless the code is optimised and implemented on a fast machine, this method is not suitable for practical use. The method is also application dependant and not suitable for shape recognition tasks where size, position or orientation of the shape provides discriminatory information.

5.3 THE NEOCOGNITRON

The Neocognitron was implemented and tested in experiment III. However the architecture was deemed to be too complicated for practical use. The reasons for this are discussed below.

5.3.1 Strengths

The Neocognitron incorporates the feature extraction stage within the network and this in itself is both convenient and important. The introduction of a separate feature extraction stage in experiment II results in uncertainty about the actual amount of processing carried out by the neural network as the feature extraction stage may have oversimplified the classification process. This may undermine the classification capabilities of the neural network in the system. The Neocognitron is insensitive to scale, changes in position and even distortions of objects in the input pattern.

The unsupervised training used for the Neocognitron resembles closely the processes in biological visual systems. Even though this training method does not produce good results in practice, it does give insight into the operation of biological visual systems. To produce better results, but moving away slightly from the biological paradigm, supervised training methods have been developed. These are more suited to pattern recognition and allow the teacher to control which features are to be detected by the simple planes.

5.3.2 Weaknesses

The training set consists of a single representative from each output class. It is very difficult to find the ideal representative for each output class especially when there are many variations in each class. In particular for data set II it was difficult to chose a single instance of a digit which represented all rotations of the digit.

The training process is not straightforward. In supervised training the teacher has to supply the features which each simple plane must detect. A network has on average about 50 simple planes. To determine appropriate features for each of these involves careful analysis of the data. This process is tedious and time consuming.

The structure and operation of the Neocognitron involves a large number of computations. For the network implemented in experiment III, 1520 cells were used, which is quite large.

Unsupervised training is based on biological visual systems and is not intended specifically for pattern recognition while supervised training is designed specifically for pattern recognition as the features that are detected are determined by the teacher. A compromise between these two paradigms is to allow the network to determine the features to be detected by the simple cells during training but to supply the expected output for each input pattern during training. This training method removes the tedious task of determining the features to be detected by each simple plane, but since the network is provided with the target output during training, the system performs better for pattern recognition applications (Fukushima, 1992).

Even though the Neocognitron has been shown in some instances to perform well for handwritten character recognition, it is still a research model and is not as yet efficient for practical use.

5.4 FUTURE RESEARCH

A few areas have been identified for further development or extension.

- The experimental applications could be optimised and extended into a handwritten character recognition system, which is suitable for practical use. This would involve introducing a segmentation stage which would acquire images using a digital scanner. Areas where there is writing would be identified, parsed and the sentences broken down into words, which would be further broken down into isolated characters. The segmentation process is quite important in that if it is carried out optimally, perfect segmentation of characters would be obtained. This would significantly reduce the complexity of the recognition task. If the segmentation process is not carried out optimally then factors such as noise, distortions of the characters and variations in the size and position of the character would be introduced. These factors would increase the complexity of the recognition task.

- Data set II consisted of digits which varied in orientation but varied little in size and position. The results obtained in experiments II and III will therefore not reflect the full capabilities of the Neocognitron. This method needs to be tested on data with more variability in size and position which usually arise from incorrect segmentation.

- The structure of the Neocognitron needs to be investigated further. This may result in the development of simpler training and implementation methods and will simultaneously provide more information about biological visual system which is still the optimal image recognition system.

- Improved feature extraction methods, e.g. Fourier descriptors (Kauppinen et al., 1995), have been developed and tested with conventional classifiers. Since artificial neural networks are superior, in some instances, to conventional classifiers, the feasibility of combining these methods with artificial neural networks should be investigated.

- The properties of Gabor filters have been shown to be ideally suited for texture classification. However the computation of the Gabor representation for a digital image is difficult. New methods need to be developed to overcome this problem.

- This study has analysed the effectiveness of feature extraction methods for two dimensional shape recognition of binary images. The effectiveness of feature extraction methods on more complex image recognition tasks, using three dimensional images and colour information, needs to be analysed.

5.5 CONCLUSION

An artificial neural network, the multilayered perceptron, has been investigated for digital image recognition applications. The limitations of the multilayered perceptron, namely, unnecessary processing of redundant information and the large number of neurons needed in the input layer for large digital images have been addressed by the introduction of a feature extraction stage. Existing feature extraction methods which have been used previously with the multilayered perceptron, namely, Zernike moments, Fourier descriptors, Gabor filters, wavelets and the Neocognitron have been described and evaluated.

The application of handwritten digit recognition was chosen to practically illustrate the role of feature extraction in image recognition. Two feature extraction models, Zernike moments and the Neocognitron, were chosen as suitable for two dimensional shape recognition. Three experimental applications were designed and implemented to demonstrate the significance of introducing a feature extraction stage into the handwritten digit recognition system:

- Experiment I consisted of a system with no feature extraction.
- Experiment II consisted of a system with a feature extraction separate feature extraction stage using Zernike moments.

- Experiment III consisted of a system which combined the feature extraction and the classification stages in the form of the Neocognitron.

These experiments were tested on data obtained from the National Institute of Standards and Technology (NIST) in the United States of America. The data consisted of isolated handwritten digits. Even though in some instances the characters were cut off or parts of two characters were found in some images (segmentation error), the segmentation process used to produce the isolated characters was considered overall to be quite good since:

- the characters had minimal variations in size.
- the characters had little or no variations in position as all the characters filled the entire image.

A selection of these characters were rotated and a new data set, data set II, was constructed which contained a combination of the rotated and the normal characters. The experiments were also tested on data set II.

The results of these experiments can be summarised as follows:

- The introduction of a feature extraction method in the image recognition system, developed in experiment II, reduces the size of the feature vector and thus the amount of resources needed for implementing, training and testing of the artificial neural network. This is especially noticeable in the number of neurons needed for the input layer.
- The feature extraction process allows the system to be invariant to certain irrelevant properties of the input data.
- The disadvantages of introducing a feature extraction stage into the system, is the additional processing that is required and the loss of information that occurs when reducing the input data. This must be weighed against the advantages of invariancy and reduction in size of the feature vector which feature extraction provides. It can thus be concluded that if there are negligible variances in parameters such as size and position, then the only advantage of the feature extraction stage is reduction in size of the feature vector.

There are many applications where feature extraction can certainly play a major role in contributing to the effectiveness of the overall recognition system, and it is hoped that this research has provided enough insight in order to make better decisions regarding their design and development.

REFERENCES

Bisio, G. M. , Caviglia, D. D. , Indiveri, G. , Luigi, R. , Sabatini, S. P. (1993), A Neural Network Architectural Model of Visual Cortical Cells for Texture Segregation, In in *Proceedings of the IEEE International Conference on Neural Networks, San Francisco, California, March 28- April 1, 1993*, pp. 755- 760.

Bouatour, H. , Soulie, F. F. , Viennet, E. (1992), Solving the Human Face Recognition Task using Neural Networks, In *Artificial Neural Networks 2 - Proceedings of the 1992 International Conference on Artificial Neural Networks*, pp. 1595-1598.

Boubez, T. I. (1993), Receptive field partitioning for wavelet networks, In *Artificial Neural Networks for Speech and Vision*, edited by Mammone, R. J. , pp. 79- 96, Chapman & Hall, London.

Bovik, A. C. , Clark, M. , Geisler, W. S. (1990), Multichannel Texture Analysis Using Localized Spatial Filters, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, no. 1, pp. 55- 73.

Carpenter, G. A, Grossberg, S. , Reynolds, J. H. (1991), A Self-Organizing ARTMAP Neural Architecture for Supervised Learning and Pattern Recognition, In *Neural Networks Theory and Applications*, edited by Mammone, R. J. and Zeevi, Y. , pp. 43 - 80, Academic Press, San Diego.

Daugman, J. G. (1988), Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 36, no. 7, pp. 1169- 1179.

Daugman, J. G. , Kammen, D. M. (1987), Image Statistics, Gases, and Visual Neural Primitives, In *Proceedings of the IEEE First International Conference on Neural Networks, San Diego, California, June 21- 24,* , pp. 163- 175.

Dimitriadis, Y. A. , Coronado, J. L. , Morena, C. G. , Izquierdo, J. M. C. (1993), On-line handwritten symbol recognition, using ART based neural network hierachy, In *Proceedings of the IEEE First International Conference on Neural Networks, San Francisco, California, March 28- April 1,* pp. 944- 949.

Dunn, D. , Higgins, W. E. , Wakeley, J. (1994), Texture Segmentation using 2-D Gabor Elementary Functions, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, no. 2, pp. 131- 149.

English , T .M. , Gomez- Gil, M. D. P. , Oldman, W. J. B. (1993), A Comparison of Neural Network and Nearest- Neighbour Classifies of Hand Written Lower Case Letters, in *Proceedings of the IEEE First International Conference on Neural Networks, San Francisco, California, March 28- April 1,* pp.1618- 1621.

Fraser, D. D. , Khan, Z. , Levy D. C. (1992), A Neural Network for Meteor Trail Classification, in *Artificial Neural Networks 2,* edited by Alexander, I. , Taylor, J. , Brighton, United Kingdom, Elsevier Science Publishers B. V. , Netherlands, pp. 1155-1158.

Fukushima, K. (1988), A Neural Network for Visual Pattern Recognition, *IEEE Computer*, Vol 21, no. 3, pp. 65- 74.

Fukushima, K. , Miyake, S. , Ito, T. (1983), Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition, reprinted *in Artificial Neural Networks: Theoretical Concepts,* pp. 138- 144, edited by Vemuri, V. , IEEE Computer Society Press, United States of America.

Fukushima, K. , Wake, N. (1992), An improved learning algorithm for the neocognitron, in *Artificial Neural Networks 2,* edited by Alexander, I. , Taylor, J. , Brighton, United Kingdom, Elsevier Science Publishers B. V. , Netherlands, pp. 497- 503.

Ghosal, S., Mehrotra, R. (1993), A Two-Stage Neural Net for Segmentation of Range Images, - *Proceedings of the IEEE First International Conference on Neural Networks, San Francisco, California, March 28- April 1,*

Ghosh, J. , Bovik, A. C. (1991), Neural Networks for Textured Image Processing, in *Artificial Neural Networks and Statistical Pattern Recognition*, edited by Sethi I. K. , and Jain A.K., Vol. II, pp. 109-131, Elsevier Publishers B. V, Amsterdam.

Gouin, P. , Scofield, C. (1993) , Neural Network segmentation and recognition of text data on engineering documents, In *Artificial Neural Networks for Speech and Vision*, edited by Mammone, R. J., pp. 562 - 573, Chapman & Hall , London.

Greenspan, H. , Goodman, R. , Chellappa, R. (1991), Texture Analysis via Unsupervised and Supervised Learning, in *Proceedings of the IEEE First International Conference on Neural Networks, San Francisco, California, March 28- April 1,* pp. 639- 643.

Hanson, S. J. , Burr. D. J. (1991), What Connectionist Models Learn: Learning and Repesentation in Connectionist Networks, In *Neural Networks Theory and*

Applications, edited by Mammone, R. J. and Zeevi, Y. , Academic Press, San Diego, 1991.

Hebb, D. O. (1949), *The organisation of Behaviour*, John Wiley & Sons, Inc., New York, 1949.

Hertz, J. A. , Krogh, A. S. , Palmer, R. G. (1991*), Introduction to the theory of neural computation*, pp. 81- 89, 115- 163, Addison- Wesley Publishing Company, United States of America.

Hinton, G. E ., Williams, C. K. I. , Revow M. D. (1992), Combining Two Methods of Hand- Printed Digits, in *Artificial Neural Networks, 2 - Proceedings of the 1992 International Conference on Artificial Neural Networks* edited by Alexander, I. , Taylor, J. , Brighton, United Kingdom, Elsevier Science Publishers B. V. , Netherlands, pp. 53- 60.

Hopfield, J. J. (1982), Neural Networks and Physical Systems with Emergent collective computational abilities, *Proceedings of the National Academy of Science*, Vol. 79, pp. 2554 - 2558.

Hung, C. & Lin, S. (1993), An ARTMAP based Hybrid Neural Network for shift invariant Chinese Character Recognition, In in *Proceedings of the IEEE International Conference on Neural Networks, San Francisco, California, March 28- April 1,* pp. 1600- 1605.

Hussain, B. & Kubaka, M. R. (1994), A Novel Feature Recognition Neural Network and its Application to Character Recognition, *IEEE Transactions On Pattern Analysis and Machine Intelligence*, Vol 16, no. 1, pp. 98- 106.

Jahne, B. (1993), *Digital Image Processing* - Concepts, Algorithms, and Scientific Applications, second edition, Springer- Verlag Berlin Heidelberg, Germany.

Jain, A. K. (1989), *Fundamentals of digital Image Proessing*, Prentice-Hall, United States of America.

Kauppinen, H., Seppanen, T. Pietikainen, (1995), An Experimental Comparison of Autoregressive and Fourier-Based Descriptors in 2D shape Classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 12, no 2, pp. 201- 206.

Khotanzad A., and Lu J. -H. (1991), Shape and Texture Recognition by a Neural Network, in *Artificial Neural Networks and Statistical Pattern Recognition*, edited by Sethi I. K. & Jain A.K. , Vol. II, pp. 109-131, Elsevier Publishers B. V, Amsterdam.

Khotanzad, A., Hong, Y. H. (1990), Invariant Image Recognition by Zernike Moments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, no. 5, pp. 489-498.

Kulkarni, A. D. (1994), *Artificial Neural Networks for Image Understanding*, Van Nostrand Reinhold, New York.

Kung, S. Y. , Diamantaras, K. , Mao, W. D. , Taur, J. S. (1991), Generalized Perceptron Networks with Nonlinear Discriminant Functions, In *Neural Networks Theory and Applications*, pp. 245- 279, edited by Mammone, R. J. and Zeevi, Y. , Academic Press, San Diego.

Laine, A. , Fan, J. (1993), Texture Classification by Wavelet Signature Packets, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, no. 11, pp. 1187- 1191.

Lewis, R. (1990), *Practical Image Processing*, Ellis Horwood Limited, Great Britain.

Lisboa, P. J. G. (1992), Image classification using Gabor Representations with a Neural Network, In *Neural Networks for Vision, Speech and Natural Language*, pp 113-125, edited by Linggard R., Myers D.J, Nightingale Chapman & Hall , London.

Lu, S-Y., Hernandez, J.E., Clark, G. , A. (1991), Texture Segmentation by Clustering of Gabor Feature Vectors, In *Proceedings of the IEEE International Joint Conference on Neural Networks, Seattle, July 8-12, pp. 1683- 1687.*

Maclennan, B , *Gabor Representations of Spatiotemporal Visual Images*, Technical Report CS-91-144, University of Tennessee, Knoxville, TN 37996, September 1991.

Mallat, S. G. , (1989) A theory of multiresolution signal decomposition: the wavlet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11 no. 7, pp. 674- 693.

McCulloch, W. S. , Pitts, W. H. (1943), *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, Vol 5, pp. 115- 133.

Minsky, M. L. , Pappert, S. (1969), *Perceptrons*, Cambridge, MA: MIT Press.

Navarro, R., Nestares, O. , Cristobal, G. (1995), Image Representation with Gabor Wavelets and its applications, to be published *in Advances in Imaging and Electron Physics*, edited by Hawkes, P. W., Academic Press Inc. , Orlando, Florida.

Pao, Y. , (1989), *Adaptive Pattern Recognition and Neural Networks*, pp. 127- 128, Addison-Wesley, United States of America.

Perantonis, S. J. (1992), Higher-order neural networks for invariant pattern recognition , In *Neural Networks current applications* edited by P. G. J. Lisboa, pp. 218 -231, Chapman & Hall, London.

Pernot, E. , Vallet, F. (1991), Determining the relevant parameters for the classification on a multi-layer perceptron: application to radar data, In *Artificial Neural Networks*, edited by Kohonen, T. , Makisara, K. , Simula, O. , Kangas, J. , Elsevier Science Publishers B. V. , North- Holland, pp. 797- 802.

Rajavelu, A. , Musavi, M. T. , Shirvaikar, V. (1989), A Neural Network Approach to Character Recognition, *Neural Networks*, Vol. 2, part 5, pp. 387- 394.

Reiss, T. H. (1991), The Revised Fundamental Theorem of Moment Invariants, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 13, no. 830- 834.

Rosenblatt, F. (1958), The Peceptron: a Probabilistic Model for Information Storage and Organisation in the Brain , *Psychological Review,* Vol 65, pp 386-408.

Rumelhart, D. E., Hinton, G. E. , Williams, R. J. (1986), Learning Internal Representations by Error Propagation, In *Parallel Distributed Processing - Explorations in the Microstructure of Cognition, Volume 1: Foundations*, edited by Rumelhart, D. E., McClelland, J. L.and the PDP Research Group, pp. 318-362, MIT Press, United States of America.

Schalkoff, R. J. (1989), *Digital Image Processing and Computer Vision*, John Wiley & Sons, United States.

Schorrenberg, F. (1992), *Fukushima's Neocognitron: An Implementation, pers. comm.*, Department of Computer Science, Texas A&M University, United States.

Sommerville, I. (1989), *Software Engineering*, Addison-Wesley Publishers Ltd, Great Britain, pp. 87- 108.

Stephen, K. W. (1984), *From Neuron to Brain: A Cellar Approach to the Function of the Nervous System*, Sinauer Associates, United States of America, pp. 1-15.

Szakacsits, S. (1994), *NeoC Explorer*, pers. comm. , Jozef Attila University, Szeged, Hungary.

Teh, C. , Chin, R. T. (1988), On Image Analysis by the Methods of Moments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10 no. 4, pp. 496-513.

Viennet, E. , Soulie, F. F. (1992), Scene segmentation using Multiresolution Analysis and MLP, In *Artificial Neural Networks, 2 - Proceedings of the 1992 International Conference on Artificial Neural Networks*, edited by Alexander, I. , Taylor, J. , Brighton, United Kingdom, Elsevier Science Publishers B. V. , Netherlands, pp. 1599-1602.

Waite, J. B. (1992), Training Multilayered Perceptrons for Facial Feature Location: A Case Study, in *Neural Networks for Vision, Speech and Natural Language,* pp. 31 - 49, edited by Linggard R., Myers D.J, Nightingale, Chapman & Hall , London.

Wasserman, W. D. , (1989), *Neural Computing: theory and practice*, Van Nostrand Reinhold, New York.

Wilson, R. , Granlund, G. H. (1984), The Uncertainty Principle in Image Processing, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6 no. 6, pp. 759-767.

Wurtz R. P. (1994), *Multilayer Dynamic Link Networks for Establishing Image Point Correspondences and Visual Object Recognition,* Phd Thesis, Faculty of Physics and Astronomy, Ruhr-University, Bochum.

Zernike, F. (1934), *Physica*, Vol. 1, pp. 689.

APPENDIX I

```
/* C++ code to convert the 32x32 text images in the fl3 database to 19x19 images */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <dir.h>

/*      in the input file active pixels are represented by '*' while active pixels in the output
        image are indicated  by 'X'
*/

char ON='X',OFF='.', BL='*';

int main (int argc, char *argv[ ])
{
typedef   unsigned short image[32][32];
typedef   long double matrix[12][12];

  image old,scaled;
  float rows,col, M[4][4];
  FILE  *inp,*out;
  unsigned int  n,m,count,x,y,s,i,j,k;
  int p,q,total;
  char   line[200],outn[30],dis[4],direc[10],curdir[30];
  matrix  R;
  float  x1,y1,x_,y_,radius,a,p1,
                  num,denom,O,argum;;
  char *result,*tmp,digit;

  struct ffblk ffblk;
  int done;

  out = fopen(argv[1], "w");

  for(k=1;k<50;k++)
  {
   itoa(k,dis,10);
   strcpy(direc,".\\");
   strcat(direc,dis);              /* recurse through 49 directories for 49 writers ( 1.. 49) */
   chdir(direc);
   getcurdir(0,curdir);
   printf("working in %s\n",curdir);
  done = findfirst("*.txt",&ffblk,0);

  while (!done)
```

```
{


inp = fopen(ffblk.ff_name, "r");
digit = ffblk.ff_name[1];
                                        /* read first redundant line in image */
fgets(line,200,inp);

                                      /* read in image into array pict */
result = fgets(line,200,inp);
count = 0;
for(y=0;y<32;y++)
      {
            for(x=0;x<32;x++)
                  {
                  if (line[count] == BL)
                        old[x][y] = 1;
                  else
                        old[x][y] = 0;
                  count++;
                  };
            result = fgets(line,200,inp);
            count= 0;
      };
fclose(inp);

/* end of reading in image */

            /*    x - is the normal x axis
                  y - is the normal vertical y axis */

for(j=1;j<=2;j++)
  {
   for(i=1;i<=19;i++)              /* prints two extra lines at top */
        fprintf(out,"%c",OFF);
   fprintf(out,"\n");
  }


for(y=0;y<16;y++)               /*  generates  coordinates for 16x16 scaled down  image */
  {
    fprintf(out,"%c%c",OFF,OFF);   /* two extra pixels on left */
    for(x=0;x<16;x++)
        {
          total =0;
          for(p=-2;p<0;p++)             /* calculate average of 2x2  neighbourhood */
            for(q=-2;q<0;q++)
                total = total + old[2*(x+1)+p][2*(y+1)+q];
            scaled[x][y] = int(total / 2);
            if (scaled[x][y])
              fprintf(out,"%c",ON);
            else
              fprintf(out,"%c",OFF);

          }

      fprintf(out,"%c",OFF);            /*  1 extra for right */
      fprintf(out,"\n");

      }


  for(i=1;i<=19;i++)                          /* print extra line at bottom */
    fprintf(out,"%c",OFF);
  fprintf(out,"\n%c\n",digit);

  done = findnext(&ffblk);
  }                                    /* end of while */
  chdir("..\\");
  }                                    /* end for */
  fclose(out);
  return 0;
}
```

APPENDIX II

```
/*
        C++ program to calculate the Zernike moments of a 19 x19 image.

                ■  INPUT : text file name on command line containing image or images.
                ■  OUTPUT: magnitudes of 47 Zernike moments.

*/

#include <math.h>
#include <stdio.h>
#include <string.h>
#include <complex.h>
#include <dir.h>

/*
        this line is to change the character that represents active and inactive pixels
 */

char ON='X',OFF='.';

int B=35;                       /* set the scaling factor beta */


double Fact( int I)
 {
        int K;
        double F;

        if (I == 0)             /* procedure to calculate the factorial of a number */
         F = 1;
        else
         {
                if ( (I > 0) && (I <= 500) )
         {
                F = 1;
                for(K=1;K<=I;K++)
                 F = F * K;
         }
                else
                        F = 0;
         };
        return(F);
};
```

```
int main (int argc, char *argv[])
{
typedef   unsigned short image[65][65];
typedef   long double matrix[12][12];

  image pict,n_pict;
  float rows,col, M[4][4];
  FILE  *inp,*out,*out1;
  unsigned int  p,q,n,m,count,x,y,s;
  char   line[200],outn[30],ex[5];
  matrix  R;
  float  x1,y1,x_,y_,radius,a,p1,
                 num,denom,O,argum;
  char *result,*tmp,digit;
  complex A;
  complex i = sqrt(complex(-1));

  struct ffblk ffblk;
  int done;

  if (argc < 2)
   {
         printf ("usage: test <infile> \n");
         return 1;
          }

  done = findfirst(argv[1],&ffblk,0);

  while (!done)

         /*
                   recurses through directory if command line contains wildcards
         */

  {
  inp = fopen(ffblk.ff_name, "r");
  if (inp == NULL)
   {
         perror ("Error opening source file");
         return 2;
   }

  strcpy(outn,ffblk.ff_name);
  tmp = strchr(outn,'.');
  ex[0] = 'Z';
  ex[1] = tmp[2];
  ex[2] = tmp[3];
```

```
ex[3] = '\0';
strcpy(tmp+1,ex);
out = fopen(outn, "w");

rows=col=19;                          /* set image size */
result = fgets(line,200,inp);
result = fgets(line,200,inp);

while(!(feof(inp)))                   /* read in image into array pict */
{

count = 0;
for(y=0;y<rows;y++)
        {
                for(x=0;x<col;x++)
                        {
                        if (line[count] == ON)
                                pict[x][y] = 1;
                        else
                                pict[x][y] = 0;
                        count++;
                        };
                result = fgets(line,200,inp);
                count= 0;
        };
                                                /* end of reading in image */

if (rows > col)
        radius = (col-1) /2;
else
        radius = (rows-1)/2;

        /* NOTE
                        x - is the normal x axis
                        y - is the normal vertical y axis */


        /* moments  (p,q) = (0,0) (0,1) (1,0) (1,1) */

for(p=0;p<=1;p++)
        for(q=0;q<=1;q++)
        {
                M[p][q] = 0;
                for(x=0;x<col;x++)
                 for (y=0;y<rows;y++)
                {
                        x1 = (x - radius) /radius;
                        y1 = -(y - radius) /radius;
                        if ( ((x1 * x1)+(y1 * y1)<=1) && ( x1*y1!=0 ) )
                         M[p][q] = M[p][q] + (pow(x1,p) * pow(y1,q)) * pict[x][y];

                };
        };

                                                    /* standardize image */

        x_ = M[1][0] / M[0][0];                             /* translation */
        y_ = M[0][1] / M[0][0];

        a = pow( (B/M[0][0]), .5);                  /* scale */


        printf("\nWorking ....");

        for(y=0;y<rows;y++)
         {
                for(x=0;x<col;x++)
                        {
                                x1 = (x - radius) /radius ;
                                y1 = -(y - radius) /radius;
                                x1 = x1/a + x_;
                                y1 = y1/a + y_;             /* map image to unit circle */
                                x1 = x1*radius + radius;
                                y1 = radius - (y1*radius)  ;
                                if (
                                        ( (int(x1) >= col-0.4) || (int(x1) < 0.3) )
                                  || ( (int(y1) >= rows-0.4) || (int(y1) < 0.3) )
                                  )
                                  n_pict[x][y]=0;
                                else
                                  n_pict[x][y] = pict[int(x1)][int(y1)];
                             /* if (n_pict[x][y]==1)
                                 fprintf(out1,"*");
                                else
                                 fprintf(out1,".");*/
                        };
         };



        printf("\n%c",digit);
        for(n=2;n<=12;n++)
         for(m=0;m<=n;m++)                          /* calculate Zernike moments */
    if(  (n-m) % 2 == 0 )
    {                                               /* check if M is correct */
     A = 0;
```

```
for(y=0;y<col;y++)
      for (x=0;x<rows;x++)
  {
      x1 = (x - radius) /radius;
      y1 = -(y - radius) /radius;
      if ((x1 * x1) + (y1 * y1) <= 1)
       {
        p1 = sqrt( (x1 *x1) + (y1*y1));


        O = acos(x1/p1);                        /* calculate Theta */

        if (y1<=0) O = 2.*M_PI -O;

        R[n][m] = 0;
        for(s=0;s<= ((n-m) / 2);s++)
         {
             if (p1!=0)
              num = ( pow(-1,s) * Fact(n-s) ) * pow(p1,n-(2*s));
             else
              num = 0;
             denom = (Fact(s) * Fact( ((n+abs(m))/ 2) -s))
                         *  Fact( ((n-abs(m)) /2) -s);
             R[n][m] = R[n][m] + ( num / denom );
         };
              /* complex calculation  */
       if (pict[y][x] != 0)
        A = A + (n_pict[x][y] * conj( R[n][m] * exp(i*(m*O)) )) ;
       };

     };
     A = ((n+1)*A) / M_PI;

             /* print out magnitudes of Zernike moments */

     fprintf(out,"%-10.4f",abs(A));

     }                          /* end of m & n loop */

     fprintf(out,"\n%s", line);
     digit=line[0];
  result = fgets(line,200,inp);
  if (!((line[0]==ON) || (line[0]==OFF)))   /* check for more than one image in file */
     break;
  }  /* end of while for same file */
     fclose(out);
     fclose(out1);
     done = findnext(&ffblk);
```

```
    }
                /* end of loop to find all files that match 1st argument */

    fclose(inp);
    return 0;

}
```