OBJECT RECOGNITION BY USING ARTIFITIAL NEURAL NETWORK:

Turning point based shape recognition using NN

By

MALIK ABDALLHA OSMAN ABDULRHMAN IDRIS

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

OBJECT RECOGNITION BY USING ARTIFITIAL NEURAL NETWORK:
Turning point based shape recognition using NN

by

Malik Abdallha Osman abdulrhman Idris

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

Herman Agustiawan

Dr. Herman Agustiawan
Associate Professor
Electrical & Electronic Engineering Department
Universiti Teknologi PETRONAS

Associate Professor Dr Herman Agustiawan

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2008

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

MALIK ABDALLHA OSMAN ABDULRHMAN IDRIS

ABSTRACT

An artificial neural network (ANN) classifier for recognizing an object based on their shapes is presented, regardless their position, orientation or size. To extract features of an object, the significant point on the object known as corner or break point is extracted and the object shape is approximated by connecting this extracted break point with straight line. Shape features are associated with each segment consisting of three successive break points, or any two lines in the approximated shape. These features are the ratio between any two adjacent lines and the angle between them. The extracted features are used as input the ANN. The neural network configuration used in this project is multi-layer perceptron using back-propagation learning algorithm. In this project two type of shape have been recognized by a MLP. The network performance is evaluated by presenting several examples to the network and determines the difference between the tested image and the original shape used in the training, until the differences are minimized.

ACKNOWLEDGEMENTS

Acknowledgement is given to University technology Petronas.

To my respective supervisor, Associate Professor Dr Herman Agustiawan of the School of Electrical & Electronic Engineering department at University Technolohy Petronas, for giving guidance, frequent input and feedback for the duration of the project.

My family and my friends.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN:	Artificial neural network
BP	: back-probation algorithm
Net	: Network
NN	: Neural network
Sim	: Simulate
Train	: Train the network

CHAPTER 1
INTRODUCTION

1.1 Background Study

Shape recognition from an image is crucial stage and it has considerable role in the field of computer vision and image processing. Several applications of shape recognition systems can be found in the industrial environment and in the marine applications. For example in marine application shape recognition system can be used for automatic target recognition, motion and structure estimation of moving object such as waves, also can be found in the application of recognizing an object from a distance specially in most military application, application of Shape Recognition algorithms for A robotic test bed, shape recognition and orientation detection for industrial applications using ultrasonic sensors, etc.

Shape usually is characterized by invariant geometrical relationship of the relative distance among set of the shape features of the object. Two objects are considered to have the same shape if they have similarities in rotation and scale that map the shape features of an object into those of the other object.

Shape recognition system is usually defined by the set of the features used to characterize the shape and the matching algorithm used. Shape features are categorized into to types either global features such as silhouette or contour or local features such as line segment, point, corner, holes or combination of these features are defined as local shape features. In defining the important parameter of the project, the first step of defining the shape is to extract its features from booth model and the test shape under some consideration such as the beginning point of scaling and segmenting the object shape [7]. After extracting the features of both model shape and tested, the task of the classification consist both comparing between the shape features extracted from the test shape and matching them with the stored set of features, from the previous two line we conclude that the recognition problem could be either simple or complicated, in other words it's just depend on the available object and it's shape features. But of course it's desirable that the shape features are invariant with respect to the position and orientation. Figure 1.1 shows examples of potential break point. It's also appreciated and desirable that the shape features can be recognizable to classification system even when only partial shape features are presented to the recognition system and also for classification purpose the lesser the similarities between the two classified object the easier to the classification system to recognize and distinguish between them. So for shape classification the local shape features are preferred since the object are not classified based on the entire features and also and also the global feature will only affect the portion of the shape features which to some extend can be ignored since the local features corresponding to the other section of the shape will remain unaffected .

In recent years several approach have been introduced for shape recognition[1], but the introduction and use of artificial intelligences such as neural network with it's complex and high parallel classification leads to fast and high efficient recognition system. The basic approach is to train the network several times by set of extracted features of the original model shape and test the resulting network using testing shape. In this project the emphasis is given to recognize an object based on it's shape extracted feature using Artificial neural network (ANN), Hence the main chapter in this dissertation discuses how to develop shape recognition system using artificial neural network. Most of shape recognition system used measurement of line smoothing around the object in order to extract the shape features of an object. In this dissertation the outer boundary of an object is approximated in such way that it give the approximate shape of an object, Then the dominant point are extracted from the approximated shape manually, The approximated shape features used as input to train the network.

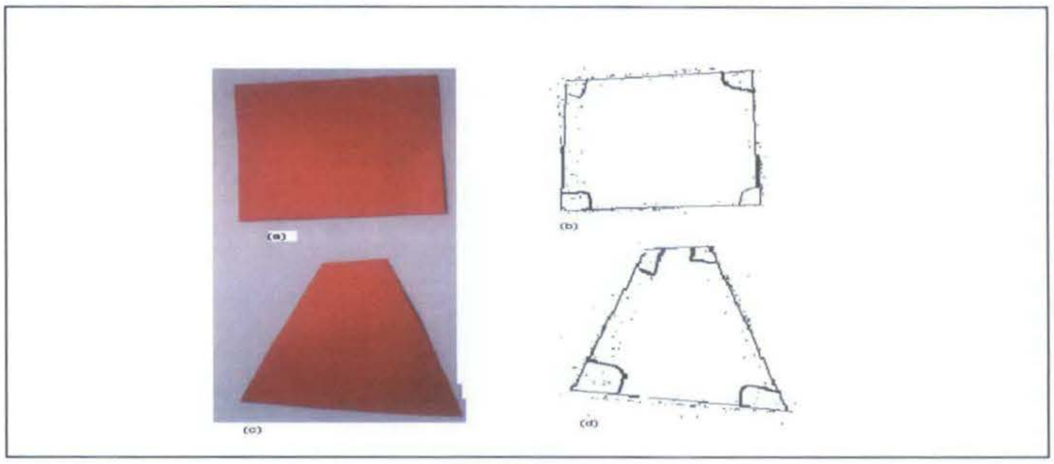

Figure 1.1: (a) Object A (b) Extracted shape of object A(c) Object B (d) Extracted shape of object B .

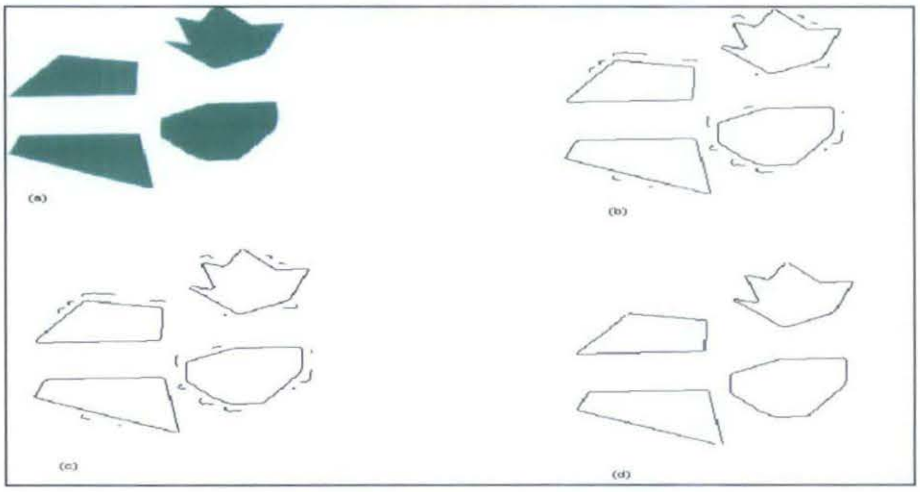

Figure 1.2: (a) Original object (b) Contour of object a (d) Smooth contour of shape b

By having approximated shape of an object, the next step is to extract it turning point. Since the shape is defined based on its most visual part or in other words some of their dominant point is rich in information contend and sufficiently characterize the shape of an object to great extend. The concept of dominant point is also known as break point or corner point, has been applied in different engineering field of studies such as shape recognition and module or significant structure parameters estimation [2][9].

In this project the concept of dominant point of critical point is used to extract the features of the shapes. Examples of this point are the most critical and significant point in the structure of the shape such as corner point extreme curvature point. Shape example figure 1.2, Figure 1.3 shows an example of break point of simplified shapes.

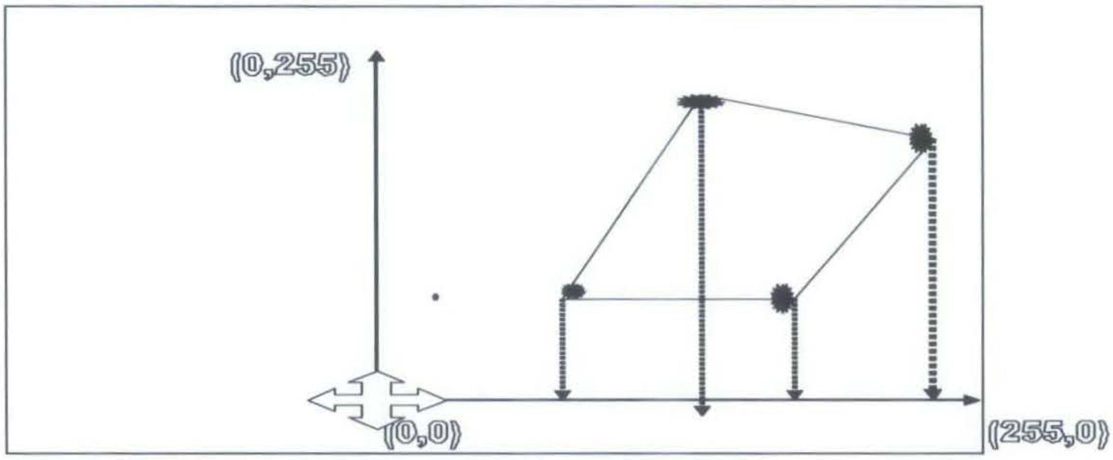

Figure 1.3: Break point of square shape

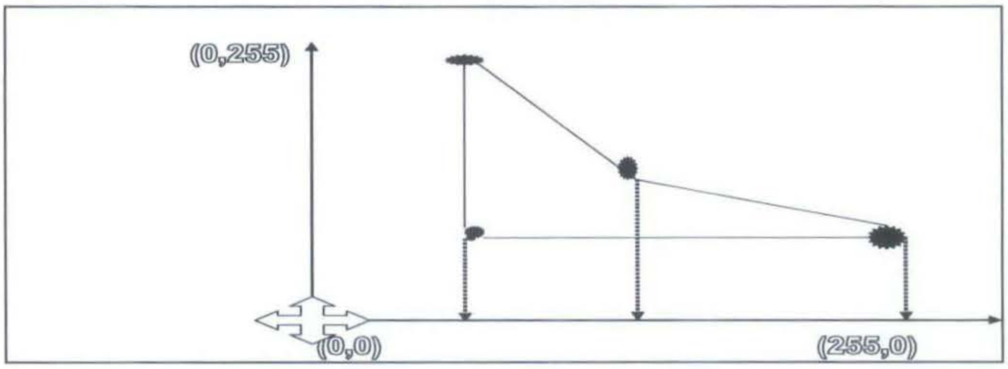

Figure 1.4: Break point shape B

Shape features used in this project are derived from curvature method [9], by joining extracted break points with straight line. A segment consist several break point will be used to represent the approximated shape of an object.

1.2 Objective

To recognize object, using it's shape features and artificial neural network (ANN).

1.3 Problem Statement

System has been developed to recognize static shapes using artificial neural network which can perform the classification task with respect to some parameter reasonably. The system limitation required the complete shape features in order to recognize the shape of an object, in addition the orientation, the scaling and the size of the object is taken arbitrary. The problem addressed in this final year project is that of recognizing an object from its shape features (refer to figure (1.1)) using artificial neural network. The moving and overlapping object and partial shape features are beyond the limitation of the designed system. Figure 1.6 shows and explain some limitation due to overlap and shade. The system will fail in recognizing an object in such situation. As illustrated in the block diagram figure 1.5.

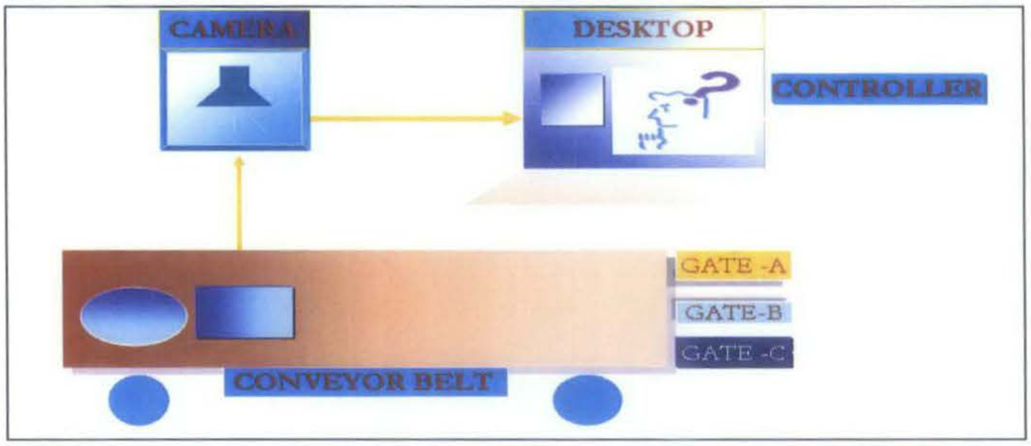

Figure 1.5: Block diagram of the classification system

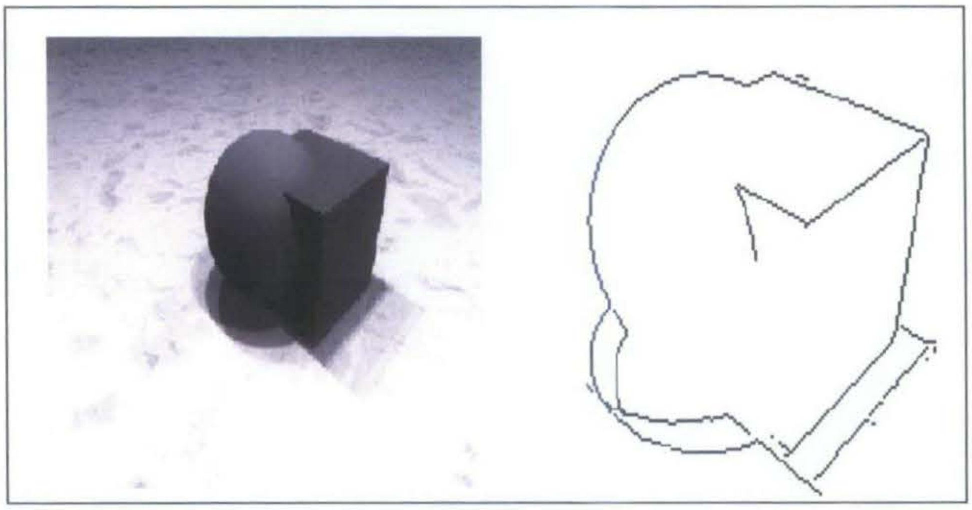

Figure 1.6: Overlap objects (a) Gray level image (b) Extracted counter shape

1.4 Significance of Project

Object recognition by using artificial neural network covers wide range of applications in many industries depending upon the nature of work. Each Artificial Neural network is trained according to the nature of work required in the industries.

This project is intended to recognize object A and object B. Therefore Artificial Neural network is trained for recognition of dominant point of shape A and shape B. This project have quite significant in the real time applications in industry where products are classified and processed to different gates to get correct packing of goods according to their shape.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

Before the age of the computer, there were many mathematical problems that humans could not easily solve, or more precisely (and this distinction is extremely important) humans were too slow in solving. Computers enabled these often simple but slow and tedious tasks to be performed quickly and accurately. The first problems solved with computers were calculating equations to resolve important physical problems, and later displaying a nice GUI, making word processors and so on. However, there are many common tasks which are trivial for humans to perform (without even any conscious effort) yet which are extremely difficult to formulate in a way that a computer may easily solve. These include:

- ❖ Signal processing such as (pattern recognition, voice recognition, image processing etc.)
- ❖ Compression
- ❖ Data reconstruction (e.g. classification where part of the data is missing)
- ❖ Data mining
- ❖ Data simplification

Scientists, engineers and mathematicians tried to make an intellectual abstraction which would enable a computer work in a similar way to that in which the human brain works - a neural network.

The detection of invariant breakpoints is a crucial preprocessing stage in classification problem such as shape recognition. In the previous chapter methods to extract break point of static shape were discussed.

Based on experimental result done before by researchers in the same field such as polygonal approximation was shown that the resulting break point are invariant with respect to orientation and scaling size of the shape. Thus the normal process and natural procedure to extract the breakpoint of any shape of an object, is the ratio between two lengths of two adjacent straight lines and the corresponding angle subtended by them as illustrated in (Eq 2.6).

A method for classifying shapes using artificial intelligences is presented in this dissertation, the specific artificial intelligences used is neural network. There are many choices for the neural network configuration but the specific neural network used is mult-ilayer perceptron which has been widely used in several supervised pattern classification problem and issues.

The next section, section 2.2 discusses the mult-ilayer preceptron (MLP) using Back-propagation (BP) learning algorithm and illustrates its general configuration and philosophy. In addition to above method to approximate and estimate segment accurately and to detect the accurate position of shape features by compensating the variation of object shape is discussed in section 2.6, also the selection of training data is discussed in the same section. Shape recognition algorithm is given in section 2.4.

2.2 Topology of the neural network

The principal importance of a neural network is not only the way a neuron is implemented but also how their interconnections (more commonly called topology) are made. The topology of a human brain is too complicated to be used as a model because a brain is made of hundreds of billions of connections which can't be effectively described using such a low-level (and highly simplified) model[3]. The topology of concern in this project is therefore not the topology of a human brain but actually a simple topology designed for easy implementation on a digital computer. One of the easiest forms of this topology at the moment is made of three layers:

- one input layer (the inputs of our network)
- one hidden layer

- one output layer (the outputs of our network)

All neurons from one layer are connected to all neurons in the next layer.

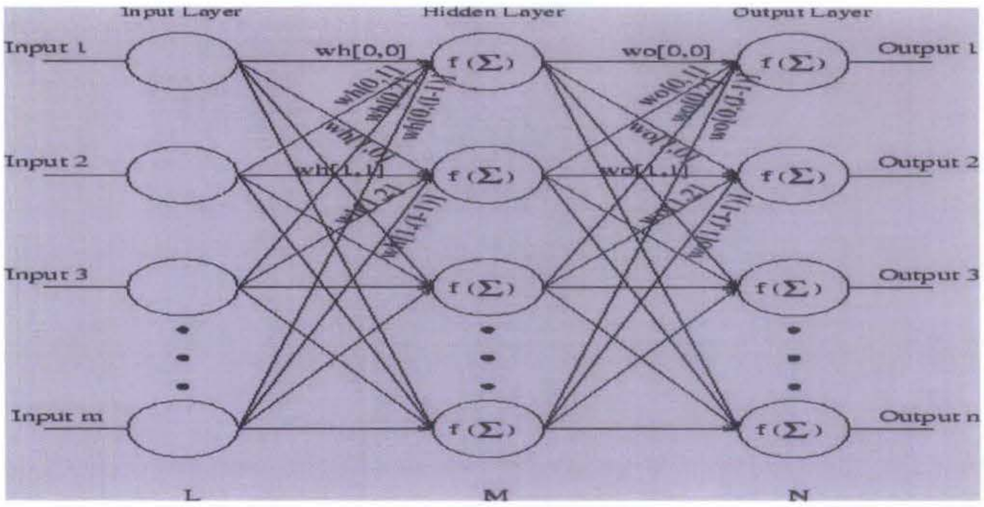

Figure 2.1: Back-propagation Network topology

This forms a whole network with full interconnection, further, the weight (and therefore the importance) of each connection is not represented (for practical reason) here but must exist in the reality.

2.3 Multilayer Perceptron

The multi-layer percepton (MLP) [3] is a feedforward network in which the preprocessing element or neurons are arranged in successive layers with the outputs of the nodes in any layer serving as an input to the nodes the next layer as shown in Figure 2.1. It has been shown that the network can be trained to learn arbitrary mapping between an input and output by presenting known input-output pairs to the network or an examples of input outputs to the network. In object classification applications of the type given an emphasis in this final year project, the network can be trained to determine the classification of unknown shape by training the network with similar pattern of known classification for many times, at least in such way that the network can learn and become familiar with type of the pattern. The system philosophy is that the shape or pattern features are applied to the first layer which know as input layer see figure 2.2, therefore the number of nodes or neurons in the input layer is equal to the number of pattern features, the nodes in the output layers serve to determine the output class family whether it's belong to class A or class B pattern respectively. Hence the number of the nodes in the output layers is equal to the number of patterns to be classified.

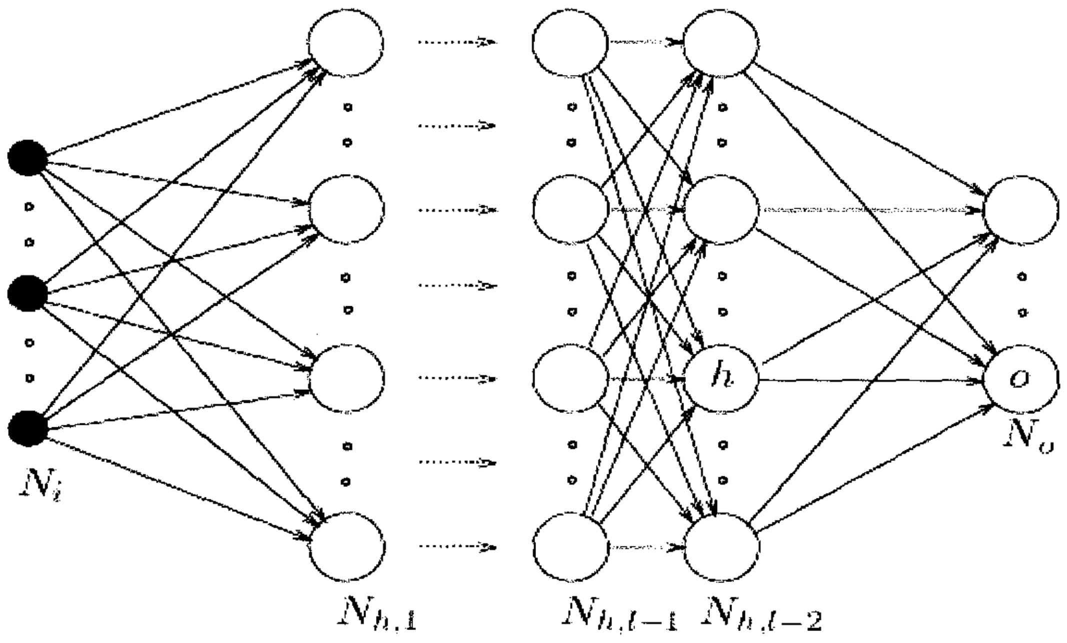

Figure 2.2: Multilayer percepton with N hidden layers & two output nodes

The layers between the input and the output layers are called hidden layers. The number of the hidden layer depend on the type of the problem needed to be solved or in other words it depend on the number of shapes segment that represent in addition the number of nodes in the hidden layer is determined imperially. The interaction between a neuron in a layer and those in the next layer is done by using weights connections. The weights quantify the amount of activation among connected neurons, for example consider the net shown in figure 2.1. Each node except the input layer nodes are connected based on the flowing equations:

$$X_J^{k+1} = \sum y_i^{\,k}\, w_{Ji}^{\,k} \quad \text{...(2.1)}$$

$$Y_J^{k} = f(x_j^{k}) \quad \text{...(2.2)}$$

Equation 2.1 is the total input received by neuron j in the layer k+1,where $y_i{}^k$ is the output of the i th neuron in the layer k ,$w^k{}_{ji}$ is the weight between the i th neuron in the layer k and the j th neuron in the layer k+1. Equation 2.2 describe the activation function , which illustrate the output of the j th neuron in the layer k , where $f(x_j{}^k)$ is assumed to be an invariable in nonlinear function, since we are using backprobogation hence usually the activation function is chosen to be sigmoid function which is defined based on Eq2.3

$$f(x_j^k) = \frac{1}{1+ e^{-}x_j^k} \qquad \text{.......(2.3)}$$

The input nodes serve as fan – out in such way that the output of an input neurons is the input it self, without any change in the input values.

Training the network involve adjusting the weight of the network so that it has the capability of classifying unknown pattern correctly. So starting with initial value of weights, the weights are adjusted using a suitable learning algorithm so that when the pattern is belonging to class (A) is presented to the network, the output of the node A_1 is high while the output of the another nodes is low . The most widely used learning algorithm is back-propagation (BP), the algorithm is also illustrate the general delta rule, which modify the difference between the input and the output absolute value.

The back-propagation algorithm consist two pass forward pass and backward pass. in the forward pass an input is pattern of known classification shape is propagated thought the network, through each successive layer until an output is generated simply as just shown in equation 2.1 and 2.2. The output pattern or the system output then compared with the desired output values. In the backwards pass, the error generated in the forward pass is transmitted back-word from the output layer to each node in the hidden layers that contribute directly to the output.

Based on the difference or amount of error received by the node the weights are updated layer by layer.

Hence training by suitable number of patterns belonging to each class enable the network to understand or to tend to understand towards certain input value and it's corresponding output. To illustrate the delta rule used by back-propagation algorithm let us assume $y^0_{j,s}$ and D_j to be the sates of the output nodes J and it's corresponding desired value s, so the weight are updated correspondingly based on the equation 2.4 and 2.5.

$$E(w) = (\tfrac{1}{2}) \sum_{j,s} (y^0_{j,s}(w) - d_{j,s})^2 \qquad (2.4)$$

So using the generalized delta rule the weight are updated according to

$$\Delta w^0_{j,i[n]} = \eta \partial^0_{j,s}\, x_{j,s} + \alpha \Delta_s w^0_{j,i[n-1]} \qquad (2.5)$$

Where:

α = the momentum parameter ($0 \leq \alpha \leq 1$),

n= the iteration number,

$\Delta_s w^0_{j,i[n-1}$ previous weight change ,

$x_{j,s=}$ net input at jth output unit ,

$\partial^0_{j,s} = y^0_{j,s}(w) - d_{j,s} * {}'\mathbf{f}(\,.\,)$,

$'\mathbf{f}(\,.\,)$ = the first derivative of f (.).

2.4 Back-propagation

Back-propagation is a supervised learning technique used for training artificial neural networks. It was first described by Paul Werbos in 1974 and further developed by David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams in 1986.It is most useful for feed-forward networks (networks that have no feedback or simply, that have no connections that loop). The term is an abbreviation for "backwards propagation of errors". Back-propagation requires that the transfer function used by the artificial neurons (or "nodes") be differentiable [3].

Summary of the technique:

Present a training sample to the neural network.

Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.

For each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output. This is the local error.

Adjust the weights of each neuron to lower the local error.

2.5 Limitation of neural network

There are some limitations to neural computing. The key limitation is the neural network's inability to explain the model it has built in a useful way. Analysts often want to know why the model is behaving as it is. Neural networks get better answers but they have a hard time explaining how they.

2.6 Shape feature generation

Shape features is generated by using the curvature method, by considering the ratio between ant to adjacent line and the angle that subtended by them, so the segment consist of any of the three corner point in the shape of an object based on Eq2.6.

$$S_i = \{r_i \ , \alpha_i, r_{i+1}, \alpha_{i+1}\} \text{..} (2.6)$$

Where:

$i=1,2\ 3.........,k,$

K=number of segment (break points),

r_i = ratio of the length. Of two adjacent lines on the approximated shape (L_i/L_{i+1}),

α_i = entire angle between two adjacent line (L_i/L_{i+1}).

2.6.1 Selection and Preparation of Training Data

A neural network is useless if it only sees one example of a matching input/output pair. It cannot infer the characteristics of the input data for which we are looking for from only one example; rather, many examples are required. This is analogous to a child learning the difference between (say) different types of animals - the child will need to see several examples of each to be able to classify an arbitrary animal. If they are to successfully classify birds (as distinct from fish, reptiles etc.) they will need to see examples of sparrows, ducks, pelicans and others so that he or she can work out the common characteristics which distinguish a bird from other animals (such as feathers, beaks and so forth). It is also unlikely that a child would remember these differences after seeing them only once - many repetitions may be required until the information 'sinks in'.

It is the same with neural networks. The best training procedure is to compile a wide range of examples (for more complex problems, more examples are required) which exhibit all the different characteristics we are interested in. It is important to select examples which do not have major dominant features which are of no interest to us, but are common to the input data anyway. One famous example is of the US Army 'Artificial Intelligence' tank classifier. It was shown examples of Soviet tanks from many different distances and angles on a bright sunny day, and examples of US tanks on a cloudy day. Needless to say it was great at classifying weather, but not so good at picking out enemy tanks.

Some times if necessary, prior to training, it's appreciated if we add some noise or other randomness to the example (such as a random scaling factor). This helps to account for noise and natural variability in real data, and tends to produce a more reliable network.

Sine the standard un-scaled sigmoid node transfer function is being used, hence the desired output must never be set to exactly 0 or 1! The reason is simple: whatever the inputs, the outputs of the nodes in the hidden layer are restricted to between 0 and 1 (these values are the asymptotes of the function. To approach these values would require enormous weights and/or input values, and most importantly, they cannot be exceeded. Once again, it cannot be overemphasized a neural network is only as good as the training data! Poor training data inevitably leads to an unreliable and unpredictable network.

Having selected an example, we then present it to the network and generate an output.

CHAPTER 3
METHODELOGY

3.1 Introduction

The objective of the project is to design recognition system using artificial intelligences. Artificial intelligences are defined by Marvin Minsky in 1968. He defines AI as "the science of making machines do things that would require intelligence if done by men." The specific artificial intelligence used in this dissertation is neural network using multilayered perceptron with back-propagation learning algorithm. Figure3.1 configuration of the classification system implemented.

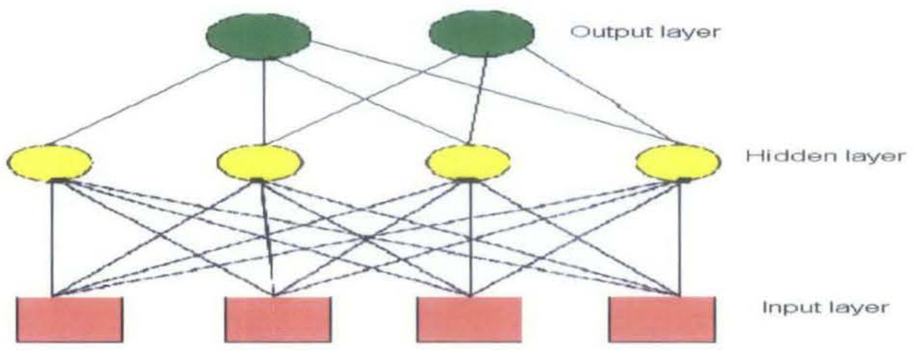

Figure 3.1: Block diagram structure of multilayer perceptron NN

As mentioned in the previous chapters neural network learn thing based on training the network many times, by presenting number of considerable samples from the original data. The neural network configuration being implemented is as shown in the figure3.1; with input layer know as input layer and middle layer known as hidden layer and output layer known as output layer. The design procedure is by generation considerable number of samples from the original data of the shape dominant point, in such way that the generated data samples are not exactly same in the entire matrixs that present certain shape, in other words at least they are different to certain degree of sensitivity.

The generated data is divided into to group, the first grub is named train data and the second groups named testing data. The neural network is training with 70 data from the 100 data set, and then rest of data is used for testing the network. The network is then tested with unknown data and the network performance is evaluated. The block diagram, figure (3.2) shows the simulation steps:

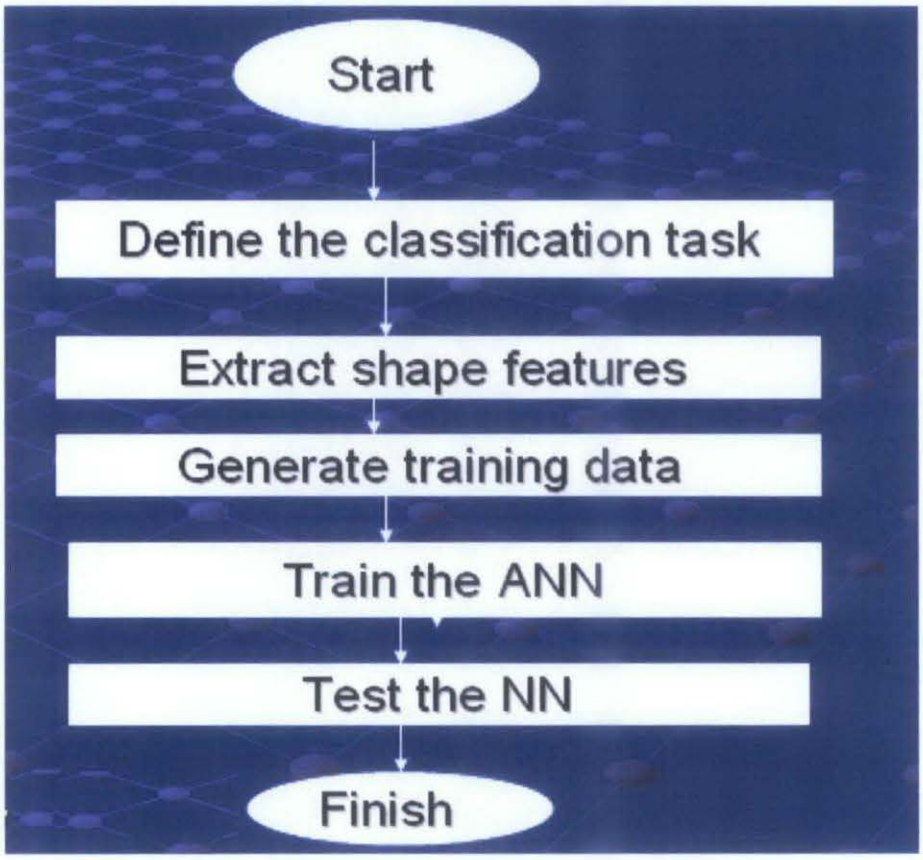

Figure 3.2: simulation process and steps

The training and testing algorithm is shown in the block diagram below

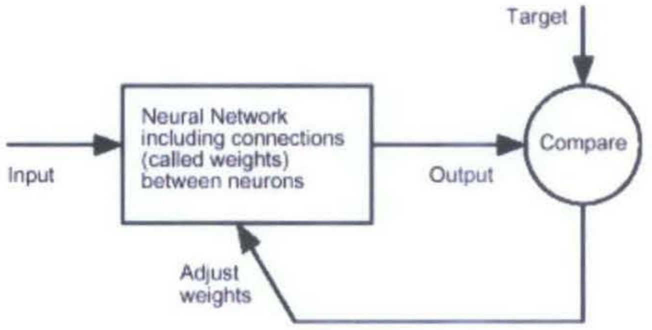
Target
Neural Network including connections (called weights) between neurons
Input
Output
Compare
Adjust weights

Figure 3.3: Block diagram of the training and testing the neural network

3.2 Shape feature generation

Shape features are generated by using Curvature Method. By considering the ratio between any two adjacent lines and the angle that subtended by them, so the segment consist of any of the three corner point in the shape of an object.

$S_i = \{r_i, \alpha_i, r_{i+1}, \alpha_{i+1}\}$ ....(3.1)

Where:

i=1, 2 3………, k;

K=number of segment (break points);

r_i = ratio of the length. Of two adjacent lines on the approximated shape (L_i/L_{i+1});

α_i = entire angle between two adjacent line (L_i/L_{i+1}).

3.3 Shape feature generation steps

- Extract the contour of an object.
- Determine the dominant point in the shape.
- Use Curvature Method to find the relation between any two adjacent lines and their corresponding angle, and Generate shape matrix.

3.4 Software

NN tool box in MATLAB.

3.5 System Architecture Flow

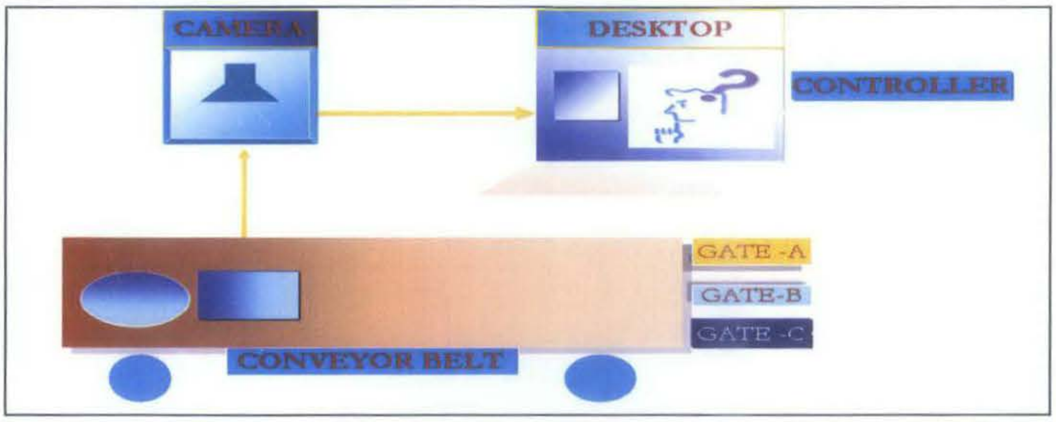

Figure 3.4: System Architecture Flow

4.1.2 Converting angle in degree to radians

Mat lab can not understand the angle in degree. I have used the formula below to convert the angle to radiance so that mat lab can understand.

Anglex1= x

Y1=deg2rad (Anglex1)...(4.1)

4.1.3 Shape features table 1

Table 4.1 Shape A feature generated

Segment	Breakpoint	Shape feature
1	(1,2,3,4)	(1.0000 , 1.5708 , 1.0000 , 1.5708)
2	(2,3,4,5)	(1.0000 , 1.5708 , 1.0000 , 1.5708)
3	(3,4,5,6)	(1.0000 , 1.5708 , 1.0000 , 1.5708)
4	(4,5,6,7)	(1.0000 , 1.5708 , 1.0000 , 1.5708)

4.1.4 Shape B feature extraction

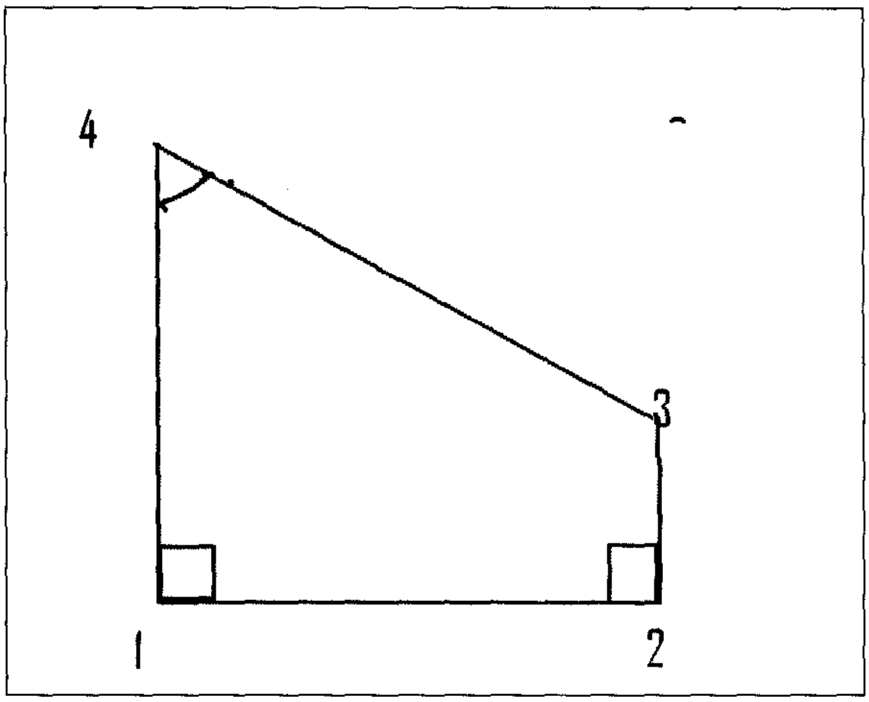

Figure 4.2: shape B

4.1.5 Shape features table 2

Table 4.2 Shape B feature generated

Segment	Breakpoint	Shape features
1	(1,2,3,4)	(0.5000 , 1.5708 , 2.2360 , 2.3562)
2	(2,3,4,5)	(2.2360 , 2.3562 , 0.8944 , 0.7854)
3	(3,4,5,6)	(0.8944 , 0.7854 ,1.0000 , 1.5708)
4	(4,5,6,7)	(1.0000 , 1.5708 ,0.5000 , 1.5708)

4.2 Training data set

The training data set are generated from the original shape, in such way that the 100 samples that generated from both shapes are at least has difference 0.1 between them.

```
Anglex1=90;
Anglex2=135;
Anglex3=45;
Y1=deg2rad(Anglex1);
Y2=deg2rad(Anglex2);
Y3=deg2rad(Anglex3);

x1=Y1;
x2=Y2;
x3=Y3;
```

Refer to appendix B for the generated shape data from the original shape of A and B.

4.3 The desired shapes features

4.3.1 Shape A

```
t1=[ 1.0000   1.5708   1.0000   1.5708; 0.5000   1.5708   2.2360   2.3562] ;
```

4.3.2 Shape B

```
t2=[0.8944   0.7854   1.0000   1.5708;1.0000   1.5708   0.5000   1.5708];
```

4.4 Neural network configuration

The two shape features are considered non-liner classified, the network topology implemented is multilayer perceptron MLP using back-propagation algorithm. The project implemented is to classify between to object, hence the number neurons in the output is set constant, $\text{Output}_{\text{neurons}} = 2$, the number of neurons in the input layer are also constant because the neurons in the input layer must be same as the number of

shape feature generated hence $\text{Input}_{\text{neuron}}$ = shape feature segment.
The number of neurons in the hidden layer is variable; in such way we can change the number of neurons in the hidden layers in order to improve the network performance.

The figure below shows the illustration of MLP being implemented.

4.4.1 Network configuration block diagram

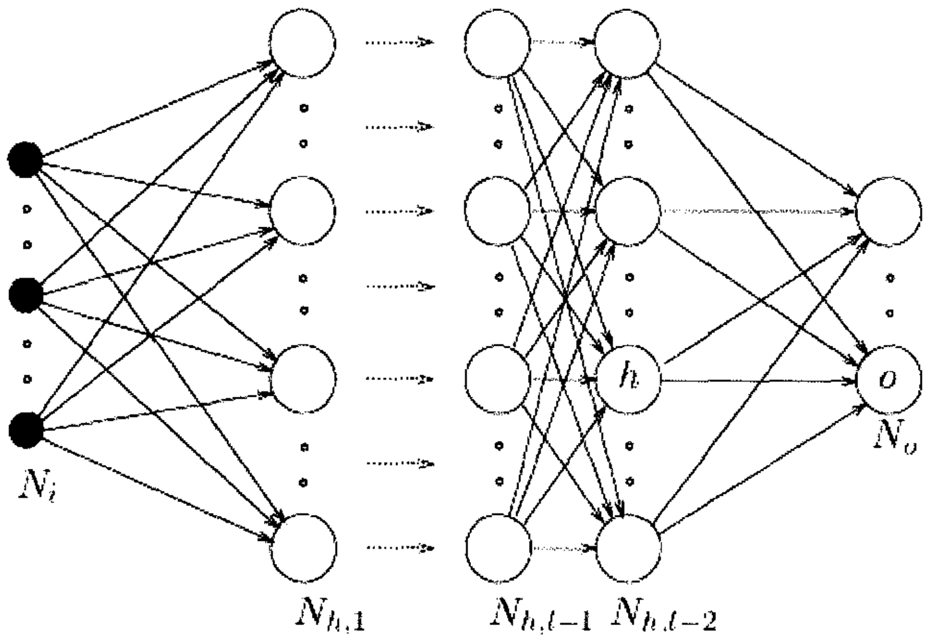

Figure 4.3: MLP configuration

The network configuration is Back-propagation with constant number of neurons in the input layers which is equivalent to the shape features and also tow output each belong to either shape A and the other belong to shape B, the number of neurons in the hidden layer if variable and can be adjusted in order to improve the network performance .

Neural Network with four input and tow outputs is implemented.

4.5 Mat lab implementation of ANN classification system

4.5.1 Network creation using mat lab

```
net = newff([0.5 2.5;0.785 2.5 ;  0.9  2.5; 0.79 2.5], [4,2], {'logsig' 'logsig'}) ;
```

where : newff is feed forward function to create the MLP network .

logsig : is the neuron activation function used usually in Backpropagation refer to Eq (3.3) .

hence the line command above create the network with four hidden neurons, four input and two output .

4.5.2 Set network parameters

```
>> net.trainParam.epochs =1000;
>> net.trainParam.goal =0.01;
>> net.trainParam.lr =0.001;
>> net.trainParam.show =1;
>> net.trainParam.time =1000;
```

4.5.3 Simulate the neural network with the rest of data set

```
y1=sim (net,A) ;
y1=sim (net,B) ;
```

where A : A1..................A_n

n=10;

where B : B1..................B_n

n=10;

Refere to appendix B for the simulation data .

4.6 Simulation result

Training result

```
TRAINLM, Epoch 0/100, MSE 0.574302/0, Gradient 6.57681e-012/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.
```

```
TRAINLM, Epoch 0/100, MSE 0.16058/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.574302/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.16058/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.574302/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.16058/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.574302/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.16058/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.574302/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.16058/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.574302/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.

TRAINLM, Epoch 0/100, MSE 0.16058/0, Gradient 0/1e-010
TRAINLM, Minimum gradient reached, performance goal was not met.
```

Error A

0.7

Error

1.2

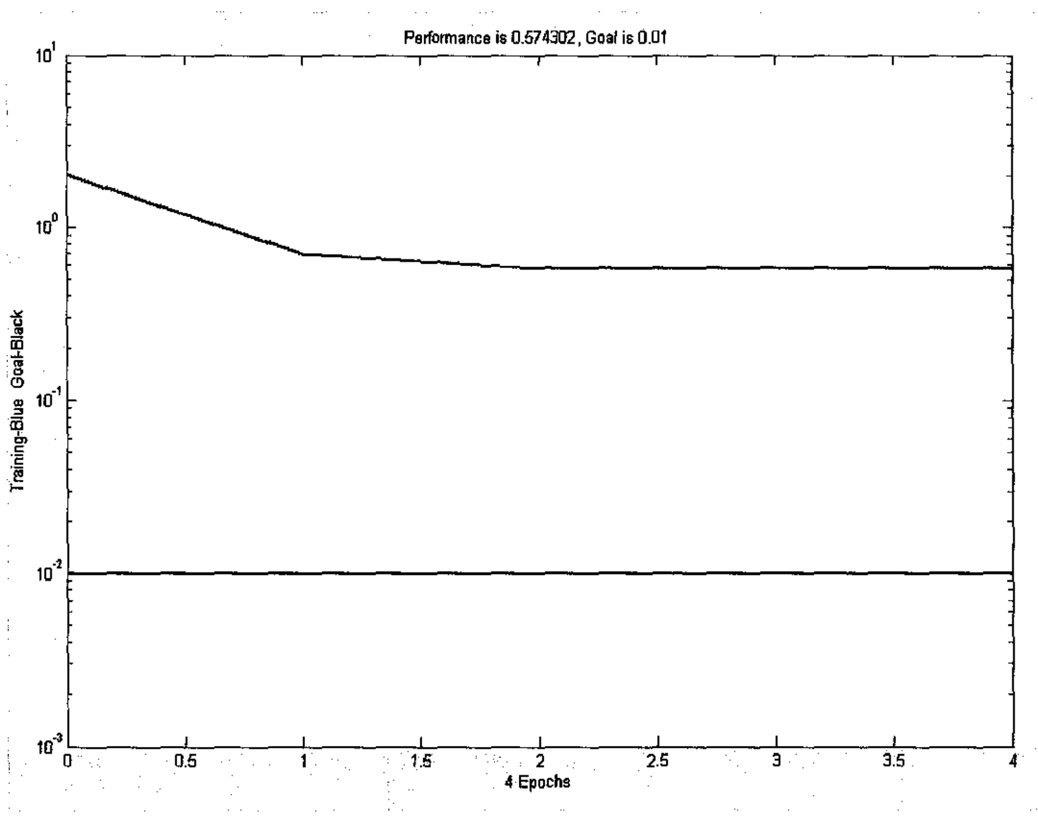

Figure 4.4: Training performances.

Form the above graph the best training eeror achived was 0.7 .

4.7 Discussion

The main objective of this project was to classify object based on their shapes dominant point. The network was trained with many samples from the original data in such way the difference between the original data and the presented data is minimized

The network prediction was significantly improved after increasing the number of training data.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The recognition system is based on artificial intelligences and the shape features generated. The neural network performance is very poor and that is due to similarities between the two shape features that define the classification problem as shown in the result chapter, the calculated squares mean average is very high. The larger the difference in the shape features that define the classification problem the better then network recognition performance, the network performance is also adjustable because the number of neurons in the hidden layer is variable, by adjusting the number of neurons in the hidden layer we can improve the network performance as another alternative.

Shape classification using multilayer percetron and BPN as learning algorithm was implemented to recognize the shape features. Only 70 samples were used for training the network. The network converges after 100 training cycles. It was determined that the performance of the network degraded slightly after 100 training cycles. In the preprocessing stage, there were difficulties trying to Segment the connected adjacent line and their corresponding angle, but the segment of the shape were approximated in such way to cover the contour of the shape. As shown in Fig 1.2.

Some portion of the shapes may actually consists of two connected portion, and this would normally be misclassified. Further research is needed in order to smooth the boundary and to apply the implemented network for real time application. In order to improve the network prediction, the problem of real time applications lay in synchronizing the real data set taken by digital camera and the post processing stage before presenting the shape dominant point to the network. But this alone can not guarantee 100% accuracy.

I believe that the recognition percentage would have been higher if the training data set had been increased. This was partially verified by increasing the training samples from 70 to 80. The recognition percentage increased significantly.

5.2 Recommendation

Further research in how to develop a program that take an image by using digital camera and has a capability to translate it into form that mat Labe software directory will accept the segment image in such way that the network will take the segmented image automatically as input and then classify it based on it's experience towards such images.

REFERENCES

[1] Costa Cesar, *Shape Analysis and Classification,* 4th Edition, *CRC press Boca Raton London New York Washington D.C.*

[2] N. Ansari and E. J . Delp. Partial shape recognition : a land-mark based approach. IEEE Tans. Pattern Analysis Machine intelligence , PAMI 12, No 5:470-483, May1990.

[3] S.N. Sivanand and S.N Deepa. Introduction to neural network using Mat lab 6.0 .Tata . McGraw-Hil

[4] Gail A., and Grossberg, Stephen, "The Art of Adaptive object Recognition by a Self-Organizing Neural Network, Computer", March, 1988."

[5] "Fukushima, Kuniko, "Cognitron: A Self-Organizing Multilayered Neural Network", *Biological Cybernetics,* Volume 20, 1975."

[6] Gonzalez/woods, *Digital Image Processing,* 4th Edition, Addison Wesley 50803, 2004

[7] S.M.collines D.J.skorton and R. E. Keber. Digital image processing and analysis in echocardiography. McGraw-Hil, New yourk,1986.

[8] Application of artificial neural networks in image recognition and classification of crop and weeds", *Department of Crop and Soil Sciences, Cornell University, Ithaca, NY, USA 14853-2801. "Received 5 May 1999; accepted 30 June 2000."*

[19] G. B. Tomas Jr. Calculus and analytical geometry .Addition wesly ,1997.

APPENDIX A

MATLAB CODE FOR RECOGNITION SYSTEM

```
%-------------------------------------------------------------
% UNIVERSITI TEKNOLOGI PETRONAS
% FACULTY OF ENGINEERING
% ELECTRICAL & ELECTRONICS ENGINEERING Department
%-------------------------------------------------------------
% Author: Malik abdullha osman
% Date   : 10 April 2008
%-------------------------------------------------------------
```

Mtabe code

```
>> % backprop a per-period backpropagation training for a multilayer
feedforward
%          neural network.
%   Network = backprop(Layers,N,M,SatisfactoryMSE,Input,Desired)
returns
%   Network, a two field structure of the form Network.structure =
Layers
%   and Network.weights where weights is a cell array specifying the
final
%   weight matrices computed by minimizing the mean squared error
between
%   the Desired output and the actual output of the network given a
set of
%   training samples: Input and the SatisfactoryMSE (satisfactory
mean
%   squared error)
%
%   Input:
%    Layers - a vector of integers specifying the number of nodes at
each
%     layer, i.e for all i, Layers(i) = number of nodes at layer i,
there
%     must be at least three layers and the input layer Layers(1)
must
%     equal the dimension of each vector in Input, likewise,
Layers(end)
%     must be equal to the dimension of each vector in Desired
%     N - training rate for network learning (0.1 - 0.9)
%     M - momentum for the weight update rule [0.1 - 0.9)
%     SatisfactoryMSE - the mse at which to terminate computation
%     Input - the training samples, a P-by-N matrix, where each
Input[p] is
%      a training vector
%     Desired - the desired outputs, a P-by-M matrix where each
Desired[p]
%      is the desired output for the corresponding input Input[p]
%
%   This algorithm uses the hyperbolic tangent node function
%   2/(1+e^(-net)) - 1, for use with bipolar data
%
%   NOTE: due to its generality this algorithm is not as efficient
as a
%   one designed for a specific problem if the number of desired
layers is
```

```
%   known ahead of time, it is better to a) 'unfold' the loops
inside the
%   loop presenting the data. That is, calculate the input and
output of each
%   layer explicitly one by one and subsequently the modified error
and weight
%   matrix modifications b) remove momentum and training rate as
parameters
%   if they are known
net = train(net, A1, t1)

net = train(net, B1, t2)

net = train(net, A2, t1)

net = train(net, B2, t2)

net = train(net, A3, t1)

net = train(net, B3, t2)

net = train(net, A4, t1)

net = train(net, B4, t2)

net = train(net, A5, t1)

net = train(net, B5, t2)

net = train(net, A6, t1)

net = train(net, B6, t2)

net = train(net, A7, t1)

net = train(net, B7, t2)


net = train(net, A8, t1)

net = train(net, B8, t2)

net = train(net, A9, t1)

net = train(net, B9, t2)

net = train(net, A10, t1)

net = train(net, B10, t2)

net = train(net, A11, t1)

net = train(net, B11, t2)

net = train(net, A12, t1)

net = train(net, B12, t2)

net = train(net, A13, t1)

net = train(net, B13, t2)

net = train(net, A14, t1)

net = train(net, B14, t2)
net = train(net, A15, t1)
net = train(net, B15, t2)
net = train(net, A16, t1)
net = train(net, B16, t2)
net = train(net, A17, t1)
net = train(net, B17, t2)
net = train(net, A18, t1)
net = train(net, B18, t2)
net = train(net, A19, t1)
net = train(net, B19, t2)
net = train(net, A20, t1)
net = train(net, B20, t2)
net = train(net, A21, t1)
net = train(net, B21, t2)
net = train(net, A22, t1)
net = train(net, B22, t2)
net = train(net, A23, t1)
net = train(net, B23, t2)
net = train(net, A24, t1)
net = train(net, B24, t2)
net = train(net, A25, t1)
net = train(net, B25, t2)
net = train(net, A26, t1)
net = train(net, B26, t2)
net = train(net, A27, t1)
net = train(net, B27, t2)
net = train(net, A28, t1)
net = train(net, B28, t2)
net = train(net, A29, t1)
net = train(net, B29, t2)
net = train(net, A30, t1)
net = train(net, B30, t2)
net = train(net, A31, t1);
net = train(net, B31, t2);
net = train(net, A32, t1);
net = train(net, B31, t2);
net = train(net, A33, t1);
net = train(net, B31, t2);
net = train(net, A34, t1);
net = train(net, B31, t2);
net = train(net, A35, t1);
net = train(net, B35, t2);
net = train(net, A36, t1);
net = train(net, B36, t2);

% return the trained network
Network.structure = L;
Network.weights = w;
Network.mse = mse;
Network.presentations = presentations;
```

APPENDIX B
NN TRAINING DATA & SIMULATION DATA

NN training data and simulation data

```
Anglex1=90;
Anglex2=135;
Anglex3=45;
Y1=deg2rad(Anglex1);
Y2=deg2rad(Anglex2);
Y3=deg2rad(Anglex3);

x1=Y1;
x2=Y2;
x3=Y3;
```

Data set generated from shape A

```
A1 =[0.9 x1 1 x1 ; 1 x1 1 x1; 1 x1 1 x1;1 x1 1 x1];
A2 =[1 x1 0.95 x1 ; 1 x1 1 x1; 1 x1 1 x1;1 x1 1 x1];
A3 =[1 x1 1 x1 ; 0.99 x1 1 x1; 1 x1 1 x1;1 x1 1 x1];
A4 =[1 x1 1 x1 ; 1 x1 0.89  x1; 1 x1 1 x1;1 x1 1 x1];
A5 =[1 x1 1 x1 ; 1 x1 1 x1; 0.90 x1 1 x1;1 x1 1 x1];
A6 =[1 x1 1 x1 ; 1 x1 1 x1; 1 x1 0.95 x1;1 x1 1 x1];
A7 =[1 x1 1 x1 ; 1 x1 1 x1; 1 x1 1 x1;0.897 x1 1 x1];
A8=[1 x1 1 x1 ; 1 x1 1 x1; 1  x1 1 x1;1  x1 0.9  x1];
A9=[0.9 x1 1 x1 ; 1 x1 1 x1; 1  x1 1 x1;1  x1 0.9  x1];
A10 =[1 x1 8.99 x1 ; 1 x1 1 x1; 1 x1 1 x1;0.897 x1 1 x1];
A11 =[1 x1 1 x1 ; 8.99 x1 1 x1; 1 x1 0.95 x1;1 x1 1 x1];
A12=[1 x1 1 x1 ; 1 x1 0.96 x1; 1 x1 0.95 x1;1 x1 1 x1];
A13 =[1 x1 1 x1 ; 1 x1 1 x1; 0.90 x1 0.98 x1;1 x1 1 x1];
A14 =[1 x1 1 x1 ; 1 x1 1 x1; 0.90 x1 1 x1;0.99 x1 1 x1];
A15 =[1 x1 0.95 x1 ; 1 x1 1 x1; 0.96 x1 1 x1;1 x1 1 x1];
A16 =[0.9 x1 1 x1 ; 1 x1 1 x1; 1 x1 1 x1;1 x1 0.889 x1];
```

```
A17 =[0.91 x1 1 x1 ; 1 x1 1 x1; 1 x1 1 x1;1 x1 1 x1];
A18 =[1 x1 0.965 x1 ; 1 x1 1 x1; 1 x1 1 x1;1 x1 1 x1];
A19 =[0.92 x1 1 x1 ; 0.99 x1 1 x1; 1 x1 1 x1;1 x1 1 x1];
A20 =[0.9 x1 1 x1 ; 1 x1 0.89  x1; 1 x1 1 x1;1 x1 1 x1];
A21 =[1 x1 1 x1 ; 1 x1 1 x1; 0.90 x1 1 x1;1 x1 0.98 x1];
A22 =[1 x1 1 x1 ; 1 x1 1 x1; 1 x1 0.95 x1;1 x1 0.9 x1];
A23 =[1 x1 1 x1 ; 1 x1 1 x1; 1 x1 1 x1;0.897 x1 899 x1];
A24=[1 x1 1 x1 ; 1 x1 1 x1; 1  x1 1 x1;1  x1 0.97  x1];
A25 =[0.91 x1 0.92 x1 ; 1 x1 1 x1; 1 x1 1 x1;1 x1 1 x1];
A26 =[0.95 x1 0.895 x1 ; 1 x1 1 x1; 1 x1 1 x1;1 x1 1 x1];
A27 =[0.92 x1 0.91 x1 ; 0.99 x1 1 x1; 1 x1 1 x1;1 x1 1 x1];
A28 =[0.9 x1 0.98 x1 ; 1 x1 0.89  x1; 1 x1 1 x1;1 x1 1 x1];
A29 =[1 x1 0.95 x1 ; 1 x1 1 x1; 0.90 x1 1 x1;1 x1 0.98 x1];
A30 =[1 x1 876 x1 ; 1 x1 1 x1; 0.90 x1 1 x1;1 x1 0.98 x1];
A31 =[1 x1 8791 x1 ; 1 x1 1 x1; 1 x1 0.95 x1;1 x1 0.9 x1];
A32 =[1 x1 0.889 x1 ; 1 x1 1 x1; 1 x1 0.95 x1;1 x1 0.9 x1];
A33 =[1 x1 .092  x1 ; 1 x1 1 x1; 1 x1 1 x1;0.897 x1 899 x1];
A34 =[0.92 x1 0.91 x1 ; 0.99 x1 1 x1; 1 x1 0.98 x1;1 x1 1 x1];
A35 =[0.9 x1 0.98 x1 ; 1 x1 0.89  x1; 1 x1 0.98 x1;1 x1 1 x1];
A36 =[1 x1 0.95 x1 ; 1 x1 1 x1; 0.90 x1 0.98 x1;1 x1 0.98 x1];
A37 =[1 x1 876 x1 ; 1 x1 1 x1; 0.90 x1 0.98 x1;1 x1 0.98 x1];
A38 =[1 x1 8791 x1 ; 1 x1 1 x1; 1 x1 0.97 x1;1 x1 0.9 x1];
A39 =[1 x1 0.889 x1 ; 1 x1 1 x1; 1 x1 0.987 x1;1 x1 0.9 x1];
A40 =[1 x1 .092  x1 ; 1 x1 1 x1; 1 x1 0.99 x1;0.897 x1 899 x1];
```

Data set for generating shape B

```
B1 =[0.51 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 1 x1;1 x1 0.5 x1];
B2 =[0.5 x1 2.246  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 1 x1;1 x1 0.5 x1];
B3 =[0.5 x1 2.236  x2 ; 2.246  x2 0.8944 x3; 0.8944  x3 1 x1;1 x1 0.5 x1];
B4 =[0.5 x1 2.236  x2 ; 2.236  x2 0.8954 x3; 0.8944  x3 1 x1;1 x1 0.5 x1];
B5 =[0.5 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.896  x3 1 x1;1 x1 0.5 x1];
B6 =[0.5 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.95 x1;1 x1 0.5 x1];
B7 =[0.5 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 1 x1;0.95 x1 0.5 x1];
B8 =[0.5 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 1 x1;1 x1 0.458 x1];
B9 =[0.51 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8844  x3 1 x1;1 x1 0.5 x1];
B10 =[0.51 x1 2.246  x2 ; 2.236  x2 0.8944 x3; 0.8954  x3 1 x1;1 x1 0.5 x1];
B11 =[0.5 x1 2.236  x2 ; 2.246  x2 0.8944 x3; 0.8964  x3 1 x1;1 x1 0.5 x1];
B12 =[0.52 x1 2.236  x2 ; 2.236  x2 0.8954 x3; 0.8884  x3 1 x1;1 x1 0.5 x1];
B13 =[0.5 x1 2.236  x2 ; 2.236  x2 0.8954 x3; 0.7954  x3 1 x1;1 x1 0.5 x1];
B14 =[0.53 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.898  x3 1 x1;1 x1 0.5 x1];
B15 =[0.51 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8794  x3 0.95 x1;1 x1 0.5 x1];
B16 =[0.53 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.7974  x3 0.95 x1;1 x1 0.5 x1];
B17 =[0.51 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8955  x3 1 x1;0.95 x1 0.5 x1];
B18 =[0.5051 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8854  x3 1 x1;1 x1 0.458 x1];
B19 =[0.51 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.9 x1;1 x1 0.5 x1];
B20 =[0.5 x1 2.246  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.91 x1;1 x1 0.5 x1];
B21 =[0.5 x1 2.236  x2 ; 2.246  x2 0.8944 x3; 0.8944  x3 0.92 x1;1 x1 0.5 x1];
B22 =[0.5 x1 2.236  x2 ; 2.236  x2 0.8954 x3; 0.8944  x3 0.91 x1;1 x1 0.5 x1];
B23 =[0.51 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.9 x1;1 x1 0.52 x1];
B24 =[0.5 x1 2.246  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.91 x1;1 x1 0.53 x1];
B25 =[0.5 x1 2.236  x2 ; 2.246  x2 0.8944 x3; 0.8944  x3 0.92 x1;1 x1 0.53 x1];
B26 =[0.5 x1 2.236  x2 ; 2.236  x2 0.8954 x3; 0.8944  x3 0.91 x1;1 x1 0.52 x1];
B27 =[0.51 x1 2.336  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.9 x1;1 x1 0.5 x1];
B28 =[0.5 x1 2.346  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.91 x1;1 x1 0.5 x1];
B29 =[0.5 x1 2.246  x2 ; 2.246  x2 0.8944 x3; 0.8944  x3 0.92 x1;1 x1 0.5 x1];
B30 =[0.5 x1 2.246  x2 ; 2.236  x2 0.8954 x3; 0.8944  x3 0.91 x1;1 x1 0.5 x1];
B31 =[0.51 x1 2.246  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.9 x1;1 x1 0.52 x1];
```

```
B32 =[0.5 x1 2.236  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.91 x1;1 x1 0.53 x1];
B33 =[0.5 x1 2.235  x2 ; 2.246  x2 0.8944 x3; 0.8944  x3 0.92 x1;1 x1 0.53 x1];
B34 =[0.5 x1 2.2461  x2 ; 2.236  x2 0.8954 x3; 0.8944  x3 0.91 x1;1 x1 0.52 x1];
B35 =[0.51 x1 2.336  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.9 x1;1 x1 0.51 x1];
B36 =[0.5 x1 2.346  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 0.91 x1;1 x1 0.51 x1];
B38 =[0.5 x1 2.246  x2 ; 2.246  x2 0.8944 x3; 0.8944  x3 0.92 x1;1 x1 0.51 x1];
B39 =[0.5 x1 2.246  x2 ; 2.236  x2 0.8954 x3; 0.8944  x3 0.91 x1;1 x1 0.51 x1];
B40 =[0.5 x1 2.246  x2 ; 2.236  x2 0.8944 x3; 0.8944  x3 1 x1;1 x1 0.522 x1];
```