

Handwriting Recognition Using Artificial Neural Network

by

Goh Siew Yin

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

DECEMBER 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

+

QA

76.87

.65614

2004

1) Neural networks (computer science)

--- -- 71.0000

CERTIFICATION OF APPROVAL

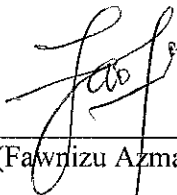
Handwriting Recognition Using Artificial Neural Network

by

Goh Siew Yin

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 by,



(Fawwazu Azmadi Hussin)

Fawwazu Azmadi Hussin

Lecturer

Electrical & Electronics Engineering

New Academic Block NO 22

Universiti Teknologi PETRONAS

31750 Tronoh

Perak Darul Ridzuan, MALAYSIA

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

December 2004

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.



GOH SIEW YIN

ABSTRACT

Character recognition is one of the areas where neural network technology is being widely used. However, a successful neural network application requires efficient implementation of image processing and feature extraction mechanism.

This project will demonstrate neural network application in recognition of constrained isolated English uppercase alphabets, from A to Z. The neural network scheme employs the Multi Layer Feed Forward Network as the alphabet classifier. This network is trained using the Back-Propagation algorithm to identify similarities and patterns among different handwriting samples. Meanwhile, the feature extraction scheme applies the combination of five distinct methods. They are Edge Detection, Kirsch Edge Detection, Line Intersection Detection, Alphabet Profile Feature and Modified Alphabet Encoder, while Image Processing involves the process of noise removal from the scanned grayscale image alphabets. Image Processing makes the handwriting easier for extraction.

In the handwriting recognition system, the neural network will use 30 sets of handwriting samples, each consisting of 26 English uppercase as training inputs and to create an automated system to recognize the handwriting alphabets in different sizes and styles. The statistical studies were done on the network to check the ability and performance of the network. This is to improve and modifying the network in order to increase its accuracy and reliability. It was found that feature extraction plays an important role in making the neural recognition system better for a more accurate detection. The handwriting recognition system is then integrated into MATLAB Graphical User Interface (GUI) that users can use very easily while hiding the complexity of the whole mechanism.

ACKNOWLEDGEMENT

Final report for Final Year Project was produced in conjunction with two academic semesters that has been undergone for one year. Throughout the long period, numbers of people have contributed in achieving the project objectives.

Firstly, the author's heartfelt gratitude is forwarded to her supervisor, Mr. Fawnizu Azmadi Hussin, for his selfless imparting of knowledge and advice, which guided the author throughout her project.

Secondly, the author would like to thank Mr. Lim Khai Loke for contributing ideas, providing useful information and guidance for betterment of the project especially in feature extraction field.

Author also wish to thank all the colleagues who taught, guided, advised, shared-knowledge, comments and helps throughout the project especially in MATLAB programming.

Finally, thank you to all the others whose names has failed to mention on this page, but has in one way or another contributed to the accomplishment of this project.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENT	ii
TABLE OF CONTENTS	iii
LIST OF FIGURES	v
LIST OF TABLES	vi
CHAPTER 1: INTRODUCTION	1
1.1 Background of Study	1
1.2 Problem Statement.....	2
1.3 Objectives	3
1.4 Scope.....	3
CHAPTER 2: LITERATURE REVIEW	4
2.1 Basic Concepts in Neural Network.....	4
2.2 Neural Network	5
2.3 The Multi Layer Feed-forward Network	6
2.4 Multi Layer Feed-forward Network Training.....	6
2.5 Backpropagation Algorithm	7
2.6 Image Processing	10
2.7 Feature Extraction.....	11
2.7.1 Modified Alphabets Encoder.....	11
2.7.2 Edge Detection Method	13
2.7.3 Kirsch Edge Detection.....	13
2.7.4 Image Compression	16
2.7.5 Line Intersection Detection.....	16
2.7.6 Alphabet Profile Feature.....	17
2.8 MATLAB GUI	18
2.8.1 GUI Layout.....	19
2.8.2 User Online Testing Capability	20
CHAPTER 3: METHODOLOGY	21
3.1 Procedure Identification.....	21
3.2 Handwriting Samples.....	22
3.3 Multi Layer Feed-Forward Network (Modified).....	22
3.3.1 Computation of Level of Confidence and Level of Substitution... 23	
3.4 Image Processing Module.....	24
3.5 Feature Extraction Module	24

3.5.1	Modified Alphabet Encoder Module	25
3.5.2	Kirsch Edge Detection Module	26
3.5.3	Image Compression Module.....	26
3.5.4	Alphabet Profile Feature Module	27
3.6	Neural Network Architecture	28
3.7	MATLAB GUI Module.....	29
3.7.1	Creating GUI with Guide.....	29
3.7.2	The Layout Editor.....	30
3.8	Development of the Handwriting Recognition System	32
CHAPTER 4: RESULT AND DISCUSSION		33
4.1	Identify the Best Value for LOC and LOS	33
4.2	Modified Alphabet Encoder and Edge Detection Method with MLFF Network	34
4.3	Adding another Hidden Layer	35
4.4	Modified MLFF Network	36
4.5	Statistical Study on the Feature Extraction Matrix.....	37
4.5.1	Standard Deviation Test on the Same Classes.....	37
4.5.2	Standard Deviation Test on the Different Classes.....	38
4.6	Final Handwriting Recognition System.....	38
4.7	Recognition Error Analysis	39
4.8	Testing and Simulation Network Modules	40
4.9	Identified Neural Network Modules.....	40
CHAPTER 5: CONCLUSION AND RECOMMENDATION		42
5.1	Conclusion	42
5.2	Recommendations.....	43
REFERENCES.....		44
APPENDIXES.....		46
Appendix A	Image Processing Module.....	46
Appendix B	Feature Extraction Module	47
Appendix C	Neural Network Module	54
Appendix D	Other Script Files	64
Appendix E	Samples Collection Form	66
Appendix F	Training Samples	67
Appendix G	Project Gantt Chart	68

LIST OF FIGURES

Figure 2.1: General Neural Network Model.....	5
Figure 2.2: Multi layer feed-forward network.....	6
Figure 2.3: MLFF Network Backpropagation.....	7
Figure 2.4: MLFF Forward Propagation Phase.....	8
Figure 2.5: Image Processing Process Flow.....	10
Figure 2.6: Modified Alphabet Encoder 15 Regions.....	12
Figure 2.7: Edge Detection Method.....	13
Figure 2.8: Kirsch Edge Detector Masks.....	14
Figure 2.9: Output of Kirsch Edge Detector.....	14
Figure 2.10: Image Convolution Process.....	15
Figure 2.11: Line Intersection Detection.....	16
Figure 2.12: Alphabet Profile Feature.....	17
Figure 2.13: GUI Design Principles.....	18
Figure 2.14: Introduction To Handwriting Recognition System Layout.....	19
Figure 2.15: Handwriting Recognition System Layout.....	19
Figure 2.16: User Online Testing.....	20
Figure 3.1: Procedure Identification.....	21
Figure 3.2: Image Processing Module Algorithms.....	24
Figure 3.3: Modified Alphabet Module Algorithm.....	25
Figure 3.4: Kirsch Edge Detection Module Algorithm.....	26
Figure 3.5: Image Compression Module Algorithm.....	26
Figure 3.6: Alphabet Feature Module Algorithm.....	27
Figure 3.7: GUI Layout Editor.....	30
Figure 3.8: Property Inspector.....	31
Figure 3.9: Handwriting Recognition System Development.....	32
Figure 4.1: Determination of LOC and LOS.....	34
Figure 4.2: Neural Network Module.....	40
Figure 4.3: Final Handwriting Recognition Module.....	41

LIST OF TABLES

Table 3.1: Summary Feature Extraction Modules.....	24
Table 3.2: Training Parameters.....	28
Table 4.1: Results for Recognition Accuracy.....	33
Table 4.2: System Accuracy Result.....	35
Table 4.3: System Accuracy Results	35
Table 4.4: System Accuracy Results	36
Table 4.5: Standard Deviation Test Results	37
Table 4.6 Standard Deviation Test Results.....	38
Table 4.7 Final Proposed Handwriting System Accuracy.....	39
Table 4.8: Summary of Neural Network Module.....	40

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Handwriting recognition of general handwritten character presents a number of challenges. Over the past year, handwriting recognition has been receiving a great deal of interest by researchers all around the world. Nowadays, handwriting recognition system has been implemented in numerous applications such as address and zip code recognition, signature identification, and forms processing to name a few.

Using artificial neural network for handwriting recognition is a field that is attracting a lot of attention. The use of neural network to model the handwriting recognition has been gaining a lot of success in term of accuracy and reliability. However, there is no easy scheme to achieve high accuracy recognition rates. Thus, there is a need to develop more sophisticated systems, which include areas of image processing, feature extraction as well as neural network itself.

Different types of neural networks have proposed to be implemented in the handwriting recognition system. Among them are Multi-layer Feed-forward with Backpropagation Network (MLFF with BP), Self-Organizing Mapping, (SOM), Fuzzy Adaptive Resonant Theory (Fuzzy ARTMAP), and Learning Vector Quantization (LVQ). The selection of the types of neural network is vital to the accuracy and reliability of the system.

The area of research has been motivated by two approaches. The first is to find the best feature extraction method. By doing the statistical study on the feature extractions' performance in recognition, the best feature extractor was obtained so that it can highlight the important features while minimizing the meaningless features. The second is to find the best neural network, which has good generalization power and minimum substitution error.

1.2 Problem Statement

Automatic recognition of handwritten characters is a problem that is currently garnering a lot of attention especially in mail delivery field. Address on envelopes can be scanned by machine easily if it has barcode. But a lot of letters sent are handwritten; therefore post office requires a person to sort the mails. Due to this problem, we are trying to minimize labor requirement by having an automated system that can sort mails according to the handwritten address.

Another problem that we are trying to address is when gathering information from a handwritten form. A complex process is needed to efficiently process small handwritten characters in small boxes. Hampered by large amount of variation between handwritten characters, it needs research to find techniques that will improve the ability of computers to represent and recognize handwritten characters. One approach is by using artificial neural network. In this approach, an artificial neural network is trained to identify similarities and patterns among different handwriting samples.

However, the scope of the handwriting recognition system built is only on recognizing the isolated character. Other advanced processing requirements to work on the letters and forms are not covered.

1.3 Objectives

The main objective of this project is to build a handwriting recognition system by implementing it in MATLAB. This handwriting system is designed to recognize 26 isolated constrained handwritten English uppercase alphabets, from A to Z. The system must also be reliable even with noisy characters or shifted characters.

1.4 Scope

The project of handwriting recognition using artificial neural network involves four areas of research. They are Image Processing, Feature Extraction, Neural Network and MATLAB GUI.

Image processing is a process of converting any scanned grayscale alphabets to binary image so that the image can be manipulated easily. Besides that, it also involves resizing and region-of-interest (ROI) analysis the image to obtain the correct size. Feature extraction is one of the vital parts of system. It creates the linkage between the scanned binary alphabet image and the neural network. The target of feature extraction is to distinguish the uniqueness of 26 classes of alphabet so that classification task by the neural network is much easier to perform.

Another integral part of the system is the neural network itself. The neural network classification efficiency will determine the performance of the overall system in term of accuracy and reliability. Therefore, the choice of neural network is important is the key factor here.

Lastly, results will be displayed using MATLAB GUI. The MATLAB GUI is very user friendly and it is easy to create as MATLAB comes with the GUI building tool.

CHAPTER 2

LITERATURE REVIEW

2.1 Basic Concepts in Neural Network

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections between elements.

There are three types of trained neural networks; supervised training neural networks, unsupervised training neural networks and reinforcement learning neural networks. We are more interested in the supervised training neural networks compared to unsupervised learning and reinforcement learning. This is because in unsupervised learning, the hidden neuron must find a way to organize themselves without help from the outside. No samples output provided to the network against which it can measure its predictive performance for a given input. While for reinforcement learning, it is work on reinforcement from outside. The connections among the neuron layer are randomly arranged, then reshuffled as the network is told how close it is to solving the problem. Both unsupervised and reinforcement suffers from relative slowness and inefficiency relying on a random shuffling to find the proper connection weights. As for supervised learning, the method is shown in Figure 2.1. There, the network weights are adjusted based on a comparison of the output and the target, until the network output matches the target. Thus, it improves the training performance [1].

In summary, there are a variety of kinds of design and learning techniques that enrich the choices that a user can make depending on the application.

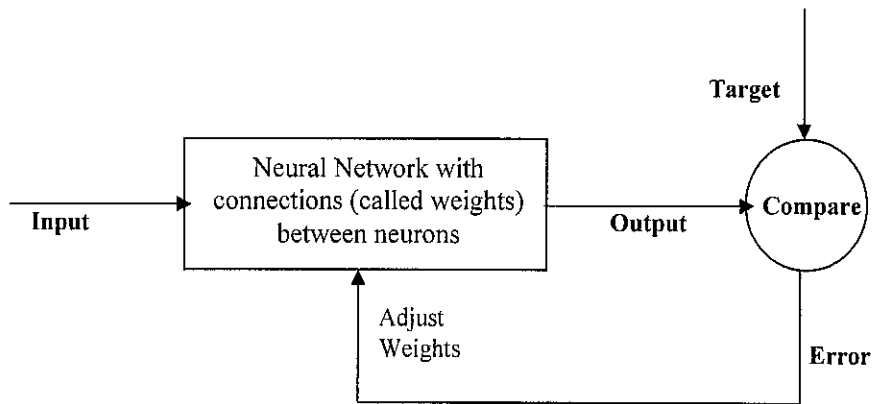


Figure 2.1: General Neural Network Model

2.2 Neural Network

A supervised neural network known as Modified Multi Layer Feed-forward Network is used for the classification and recognition of the isolated alphabets. It is chosen based on several reasons and its advantages.

- *Easy training* – all goal of training can be reached for all the training sets easily. Thus, the number of epochs needed is minimal.
- *Stability* – it can retain the previously trained alphabet pattern. Thus the neural network can accept wide range of alphabet patterns without forgetting the past information.
- *Accuracy* – it can generalize information for each class and reduce the substitution error rate, thus, increasing accuracy.

2.3 The Multi Layer Feed-forward Network

The Multi Layer Feed-forward Network with backpropagation is a supervised neural network. It can have multiple of inputs, outputs and layers of nodes or neurons. Figure 2.2 shows architecture of a three-layered feed-forward network. The leftmost is the layer which the input data is supplied; the rightmost layer is the output layer and the middle layer is the layer to interconnect input and the output layer. Each layer of the network is fully interconnected to its subsequent higher layer. The links between each neuron are called weights, where the knowledge is being stored [2].

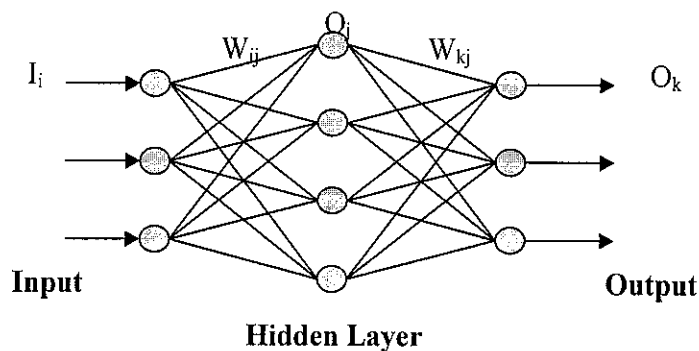


Figure 2.2: Multi layer feed-forward network

2.4 Multi Layer Feed-forward Network Training

The Multi Layer Feed-forward Network training utilizes the backpropagation algorithm. It is an optimization procedure based on gradient descent that adjusts weights to reduce the system error or cost function. The name backpropagation arises from the method in which corrections are made to the weights. During the learning phase, input patterns are presented to the network in some sequence. Each training pattern is propagated forward layer by layer until an output pattern is computed. The computed output is then compared to desired target output and error values are determined. The errors are used to feedback connections from which adjustments are made to the synaptic weights layer by layer in backward direction. Figure 2.3 illustrates an

MLFF network modified for the backpropagation training. The backward linkages are used only for the learning phase, whereas the forward connections are used for both learning and the operational phases.

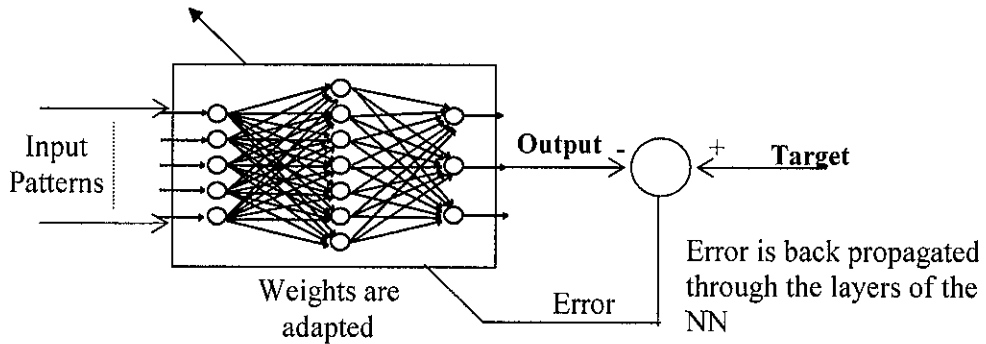


Figure 2.3: MLFF Network Backpropagation

Using BP, the hidden layer weights are adjusted using the errors from the subsequent layer. Thus the errors computed at the output layer are used to adjust the weights between the last hidden layer outputs. The error from the last hidden layer outputs is used to adjust the weights in the next to the last hidden layer and so on until the weights connections to the first hidden layer are adjusted. In this way, errors are propagated backward layer by layer with corrections being made to the corresponding weights in an iterative manner. The process is repeated a number of times for each pattern in the training set until the total output error converges to the minimum or until some limit is reached in the number of training iterations completed [2].

2.5 Backpropagation Algorithm

The backpropagation algorithm is a generalization of the Widrow-Hoff learning rule to multiple-layer network and nonlinear differentiable transfer functions. Input vectors and the corresponding output vectors are used to train a network until it can approximate a function; associate input vectors with specific output vectors or classify input vectors is defined in an appropriate way [2].

The algorithm consists of 2 phases; a feed-forward process and a backpropagation process. For the initial stage, the weights of the network are randomly selected. The learning rate, η and momentum, β is pre-set before the learning phase. Normally, the momentum rate is set at 0.95 and learning rate at 0.01. During the learning phase, an input vector is presented to the network and the vector propagates from the input layer to the output layer. Thus, the output of the hidden layer has the following notation

$$O_j = f(\text{net}_j) = \frac{1}{1 + e^{-\text{net}_j}}$$

where

$$\text{net}_j = \sum_j W_{ji} O_i + \theta_j$$

Similarly, the output layer becomes

$$O_k = f(\text{net}_k) = \frac{1}{1 + e^{-\text{net}_k}}$$

where

$$\text{net}_k = \sum_k W_{kj} O_j + \theta_k$$

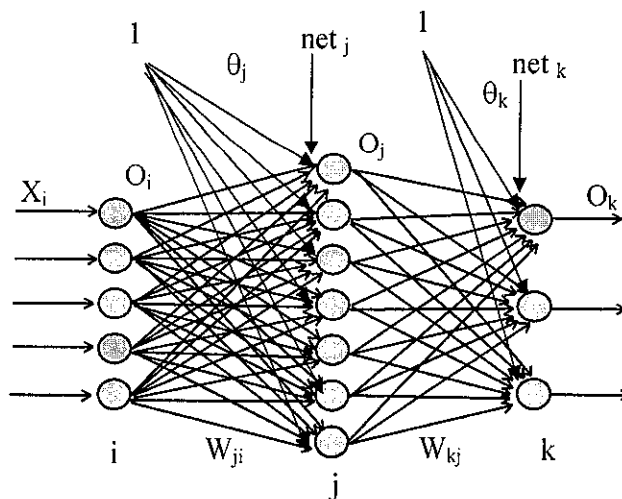


Figure 2.4: MLFF Forward Propagation Phase

The output vector generated from feed-forward process is then compared with the desired output vector. The cost function used is a sum squared error function, ξ which is given by

$$\xi = \sum_p \xi_p$$

$$\xi_p = \frac{1}{2} \sum_k (t_{pk} - O_{pk})^2$$

where t_{pk} is the desired output for the k th component of output pattern for the pattern p and O_{pk} is the corresponding actual output.

Backpropagation process is then using the error to adjust weights accordingly, based on the steepest descent method, as follows:

$$\Delta W_{kj}(t+1) = \eta \delta_k O_j + \alpha \Delta W_{kj}(t)$$

$$\delta_k = O_k(1 - O_k)(t_k - O_k)$$

$$W_{kj}(t+1) = W_{kj}(t) + \Delta W_{kj}(t+1)$$

and

$$\Delta W_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta W_{ji}(t)$$

$$\delta_j = O_j(1 - O_j)(t_j - O_j) \sum_k \delta_k W_{kj}$$

$$W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}(t+1)$$

2.6 Image Processing

Image processing involves the process of noise removal from the scanned grayscale image alphabets. The end product of this process is a noiseless grayscale image.

The noise removal process applies a Wiener filter (a type of linear filter) to an image adaptively, tailoring itself to the local image variance. Where the variance is large, Wiener filter performs little smoothing. Where the variance is small, Wiener filter performs more smoothing. This approach often produces better results than linear filtering. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high frequency parts of an image.

Then, the image threshold is set to 50% to obtain the binary image. This process involves in converting the input image to grayscale format and then uses threshold value to convert this grayscale image to binary. The output binary image has values of 0 (black) for all pixels in the input image with luminance less than level and 1 (white) for all other pixels. From the binary image, the region of interest (ROI) of the letter is identified [3]. The letter then is resized to 32×32 pixels. Figure 2.5 illustrates the flow of image processing technique applied to the samples.

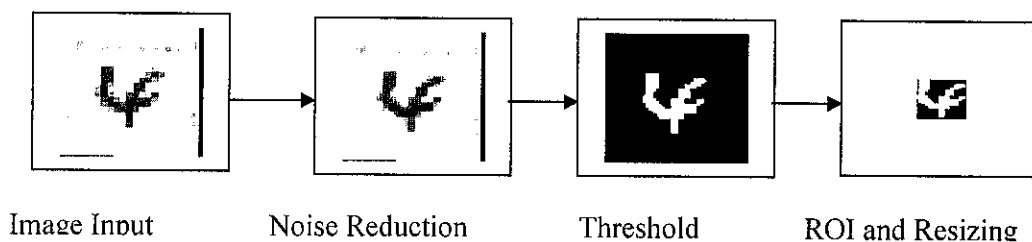


Figure 2.5: Image Processing Process Flow

2.7 Feature Extraction

Feature extraction is important in defining the characteristic of each alphabet. If each alphabet well defined, it will increase the recognition rate because the neural network able to distinguish the all classes easily. Among the technique I have come across are Modified Alphabets Encoder, Edge Detection Method, Kirsch Edge Detection, Lines Intersection Detection, and Alphabet Profile Feature.

With wide choices of feature extractor, there is a need to determine their performance. There are some general conditions for a good feature extractor.

- **Discrimination** – features extracted should be significantly different for characters belong to different classes
- **Reliability** – features for the characters from the same class should be similar to each other.
- **Feature space is small** – the amount of the features should be smaller than the original image to ensure the recognition speed is fast enough.
- **Independence** – features should not have any correlation with each other.
- **Fast processing speed** – the feature extractor should have low complexity and low computation in order to speed the feature extraction process.

2.7.1 Modified Alphabets Encoder

The modified alphabet encoder is very suitable to extract 26 English upper case alphabets using its 15 regions. With the size of 32×32 bitmap, the bitmap is divided into 15 regions, which consist of 3 horizontal regions (H1, H2, and H3), 8 vertical regions (V1, V2, V3, V4, V5, V6, V7, and V8), and 4 diagonal regions (D1, D2, D3, and D4). This is shown on the Figure 2.6.

Computationally, one feature region is defined as the number of marked bits divided with the size bit of the region. The computed number will be sent to the neural network for recognition.

$$\text{Region Feature} = \frac{\text{Number Marked bits}}{\text{Region Bits Size}}$$

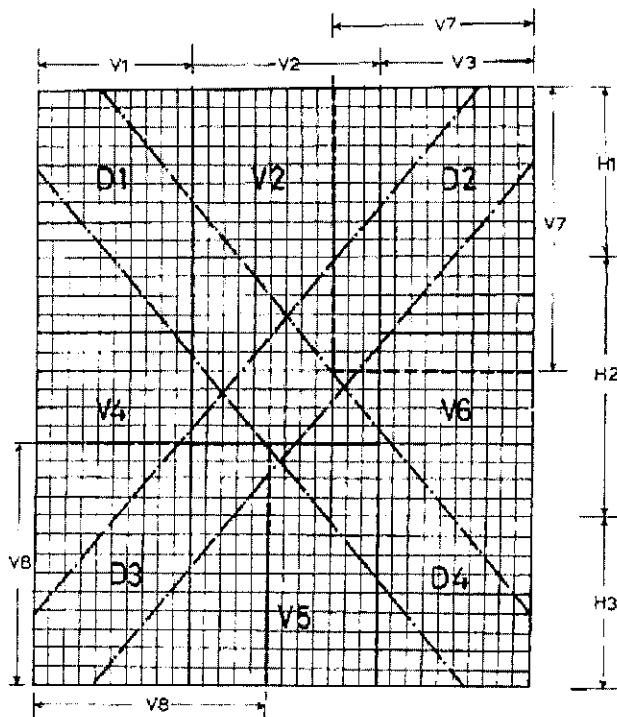
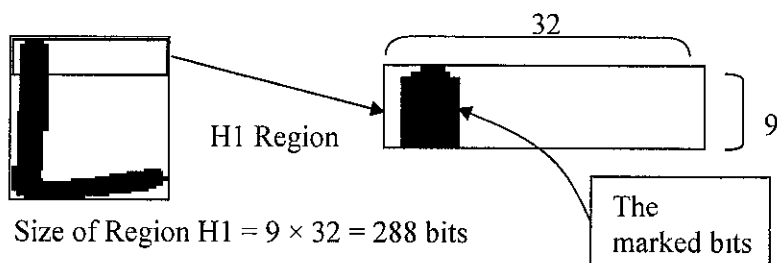


Figure 2.6: Modified Alphabet Encoder 15 Regions

For example,



Size of Region H1 = $9 \times 32 = 288$ bits

The marked bits = 50 bits

H1 feature = $50/288$

= 0.1736

2.7.2 Edge Detection Method

The idea of this method to detect the outer edge of a character from four corners view, which is left, top, right, and bottom direction. The algorithm of this method is very straightforward. It just needs to scan through line by line vertically and horizontally until there is a change of the pixel value. This process is illustrated in the diagram below.

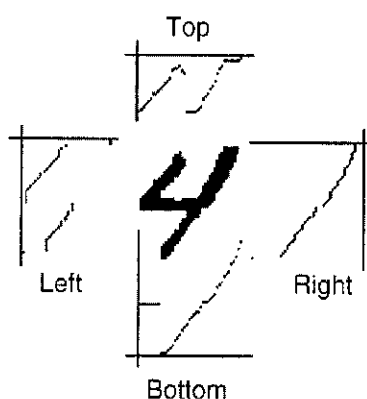


Figure 2.7: Edge Detection Method

2.7.3 Kirsch Edge Detection

This method is quite useful in extracting local feature of alphabets. It is a good discriminator for all the 26 classes of alphabets because it is able to detect unique features of each alphabet. This method involves detection of horizontal, vertical, right diagonal and left diagonal edge by using 8 masks or filters. These 8 masks will be convoluted with the binary alphabet image to produce 8 images (2 horizontal, 2 vertical, 2 right diagonal, and 2 left diagonal).

The pair of each direction images will undergo the maximum operation. The maximum operation is a process of selecting the maximum pixel value from the pair image. Thus, there will be one image for each direction. Then, the four images will undergo image compression to reduce their size.

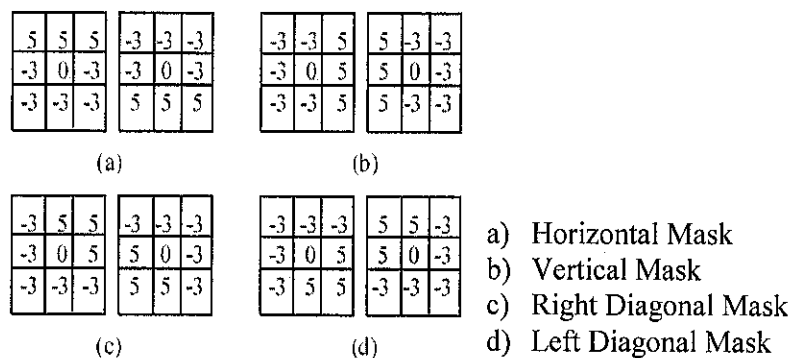


Figure 2.8: Kirsch Edge Detector Masks

For the handwriting recognition system, the size of input image is 32×32 pixels. Therefore, the output will be 4×32×32 pixels. After image compression, the output will be reduced to 4×8×8 pixels.

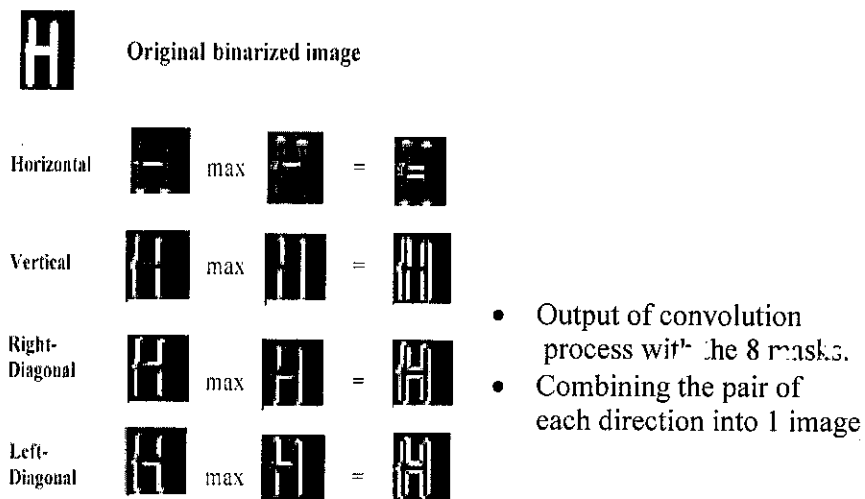


Figure 2.9: Output of Kirsch Edge Detector

2.7.3.1 Image Convolution

The convolution process involves a very simple two-dimensional convolution. The value of each pixel is computed through multiplication of two matrices and summing the results. One of these matrices is the image itself, and the other is the filter matrix. Usually the image size is bigger than the filter matrix.

There are several stages to implement the image convolution. The Figure 2.10 below is an example illustrating the image convolution with 1 horizontal filter. The first stage is zero padding. In this stage, the border of the image will be padded with zero because we need to obtain border pixels value. If the size of the input image is 32×32 pixels, then the image will become 34×34 pixels. After the zero padding process, we will compute for the value of each pixel by multiplying and shifting the filter matrix.

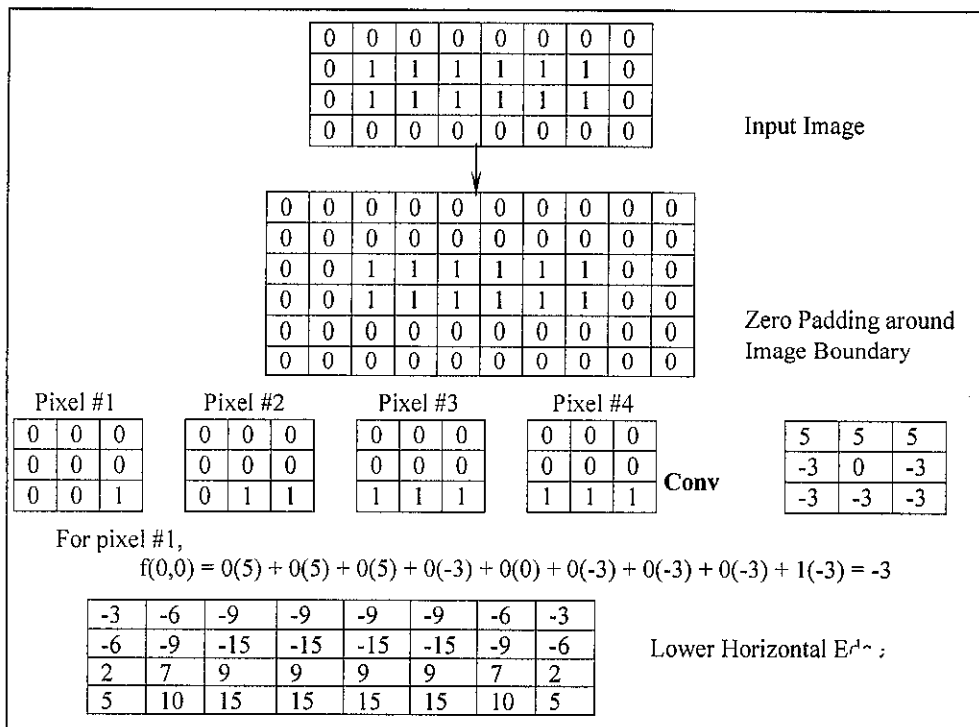


Figure 2.10: Image Convolution Process

2.7.4 Image Compression

After obtaining the image with size of 32×32 , we want reduce the size of the input to the neural network by 16 times so that less processing is needed in the neural network. This image compression is performed using this formula below.

$$T(i,j) = \frac{\sum_{x=2i-1}^{2j} \sum_{y=2j-1}^{2j} S(x,y)}{4}$$

$S(x,y)$ – Image 32×32 pixels
 $T(i,j)$ – Image 8×8 pixels

2.7.5 Line Intersection Detection

This method is extracting the local feature of an alphabet. It checks for the number of intersection between alphabet and 2 horizontal lines at $1/3$ and $2/3$ of the image height and also 1 vertical line at the center point of the alphabet.

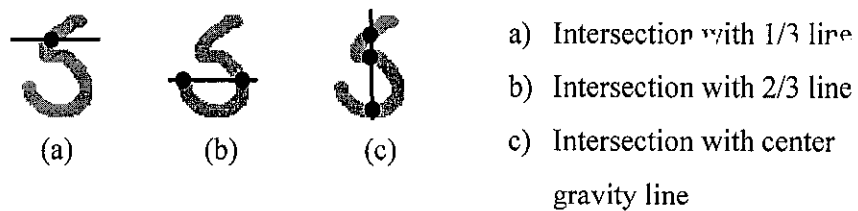


Figure 2.11: Line Intersection Detection

2.7.6 Alphabet Profile Feature

This method is highlighting the global feature of an alphabet. It detects the smoothness of the alphabets and also the width and height at different location. It involves obtaining the raw profiles of an alphabet by detecting the edge. The raw profile is differentiated to obtain the smoothness of the alphabet. The highest value of the differentiated raw profile is obtain and normalized for all four of the directions. To describe the size of the alphabet, the width and height at $1/5$, $1/2$ and $4/5$ of the bounding boxes are obtained.

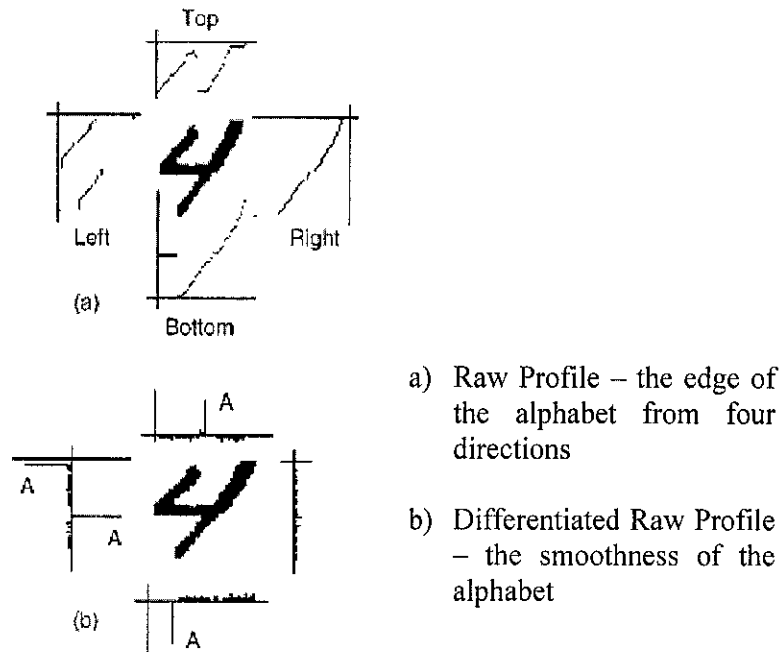


Figure 2.12: Alphabet Profile Feature

2.8 MATLAB GUI

A graphical user interface (GUI) is a user interface built with graphical objects. If the GUI is designed well-designed, it should be intuitively obvious to the user how its components function. By providing an interface between the user and the application's underlying code, GUIs enable the user to operate the application without knowing the commands would be required by a command line interface. GUI is created with GUIDE. This includes laying out the components, programming them to do specific things in response to user actions and saving and opening the GUI [4].

Hence, it is very suitable to display the input image and output recognition results on screen. Figure 2.13 shows the GUI design principles.

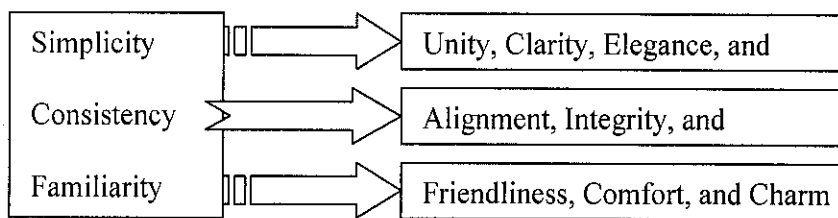


Figure 2.13: GUI Design Principles

The GUI layout created was divided into two parts. First part is the layout to show the introduction about handwriting recognition system as shown in Figure 2.14. While the second part is the layout to display all relevant data such as the alphabet image, the output image, the alphabet selection, neural network training, online testing, output results and so on. This can be seen from Figure 2.15.

2.8.1 GUI Layout

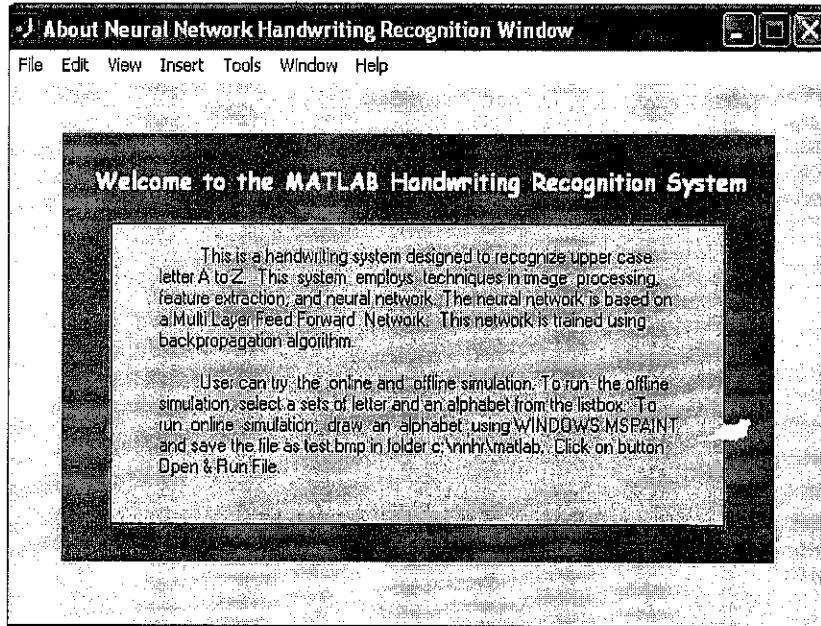


Figure 2.14: Introduction To Handwriting Recognition System Layout

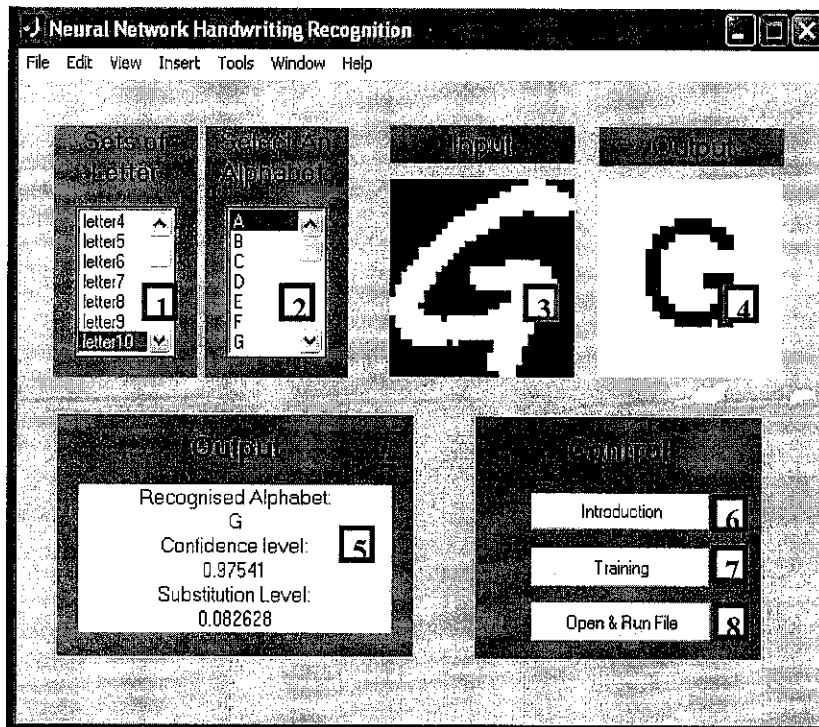


Figure 2.15: Handwriting Recognition System Layout

The designed MATLAB GUI layout has the following attributes and its functions:

- 1 : 30 sets of upper case handwriting alphabet from A to Z
- 2 : Select an alphabet to be recognized
- 3 : Show the input image of the selected alphabet
- 4 : Show the detected output image
- 5 : Show the recognition result
- 6 : Brief introduction about handwriting recognition
- 7 : Show the neural network training
- 8 : Run the input alphabet from users

2.8.2 User Online Testing Capability

The GUI provides the capability for user to input the handwritten character to test the system. User needs to draw an alphabet using WINDOWS MSPAINT and save the file as 'test.bmp' in folder c:\nnhr\matlab. Then, click button Open & Run File to view the result. This is illustrated in Figure 2.16.

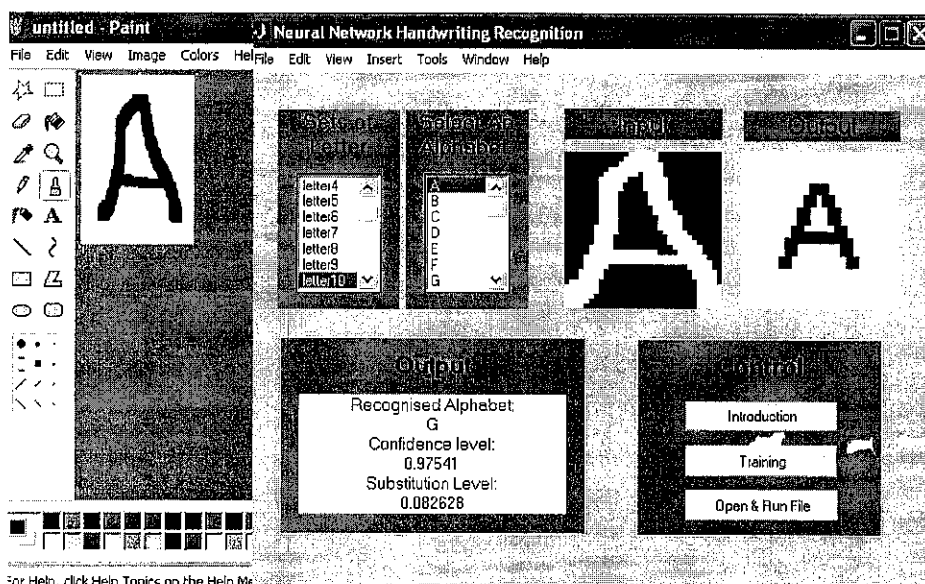


Figure 2.16: User Online Testing

CHAPTER 3 METHODOLOGY

3.1 Procedure Identification

The execution of project is divided into several stages and is illustrated in the flow chart below.

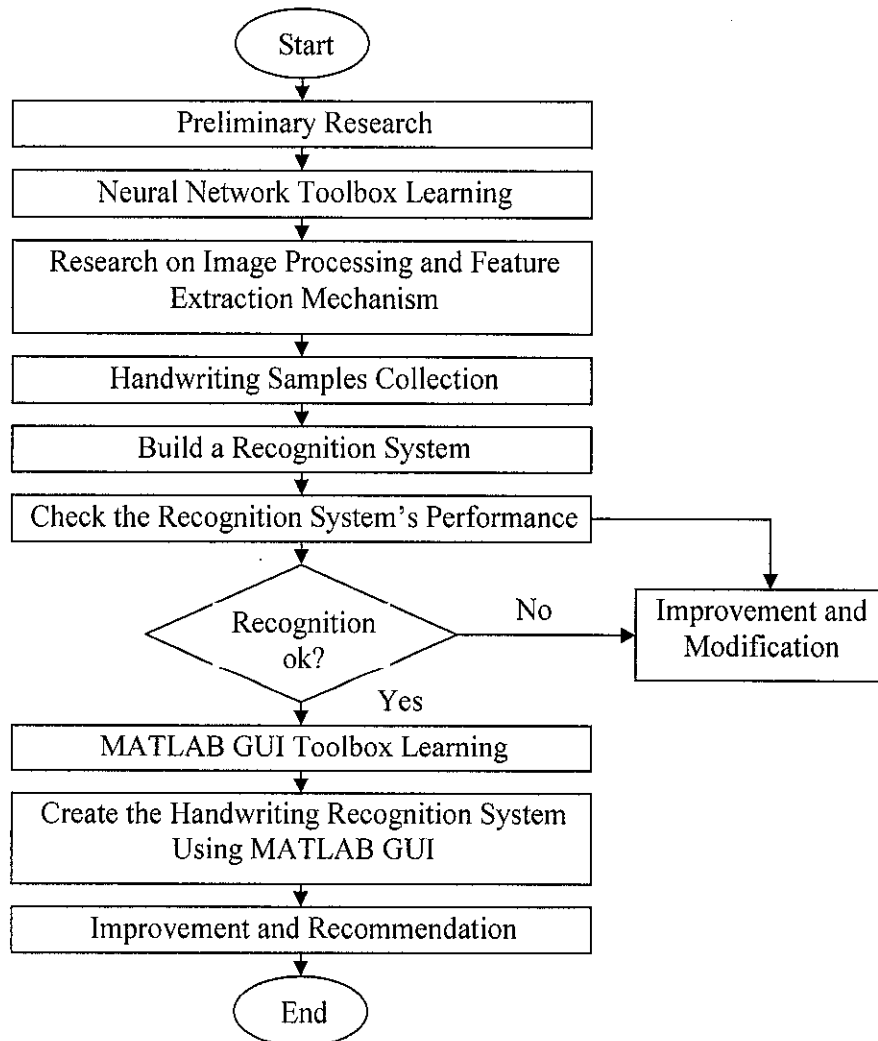


Figure 3.1: Procedure Identification

3.2 Handwriting Samples

The handwriting samples for the training set are very important because it can affect the accuracy of the system. Alphabets that are too skewed or distorted need to be discarded as training sets. As for the untrained sets of handwriting, they are randomly picked from the handwriting samples form. The untrained sets are 15 different individual names. The samples collected were then cut out separately, alphabet by alphabet, without specifying sizes. Please refer to Appendix F for training samples.

3.3 Multi Layer Feed-Forward Network (Modified)

The MLFF network has been modified to have the ability to load a different set of weights and biases if low confidence level of recognition is detected from the network output. Low level of confidence normally indicates a wrong alphabet recognition. Another parameter to consider from the output of the network is the level of substitution of other letters. Wrong recognition has high level of substitution of other letters as well. Combining these two parameters, a statement is formed as shown below.

If (confidence > threshold value) or (substitution < threshold value)
then
Accept the network output
Else
Load a new weights and biases (new network)

3.3.1 Computation of Level of Confidence and Level of Substitution

The confidence and substitution level of each letter is calculated through the output of the neural network. The 26 values in the output matrix represent the 26 alphabets. The neural network is trained to give the highest value (near to one) for the recognized letter and for the rest will be near to zero. Having this characteristic, we can evaluate level of confidence and substitution of the recognized letter. Computationally, the confidence level is the highest value in the output matrix and the substitution level is the sum of the rest of the matrix. Through trial and error as to refer to Section 4.1, it is the best the confidence threshold value is set at 0.9 and the substitution threshold value is set at 0.2.

For example,

Output matrix, $O = [0.95 \ 0.1 \ 0.1 \ 0.1 \ 0 \ 0 \ 0 \ 0.1 \ 0.01 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0.01 \ 0.01 \ 0.1 \ 0 \ 0 \ 0 \ 0 \ 0]$

Level of confidence = $\max(O)$
= 0.95 (first element)

Level of substitution = $0.02 + 0.1 + 0.001 + 0 + 0 + 0 + 0.01 + 0.01 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0.01 + 0. + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0$
= 0.151

3.4 Image Processing Module

This module implements four processes. The four are noise reduction, threshold, region of interest (ROI) and resizing. It takes in isolated alphabet images of any sizes and returns a binary image with the size of 32×32 pixels. This module uses the MATLAB Image Processing Toolbox extensively.

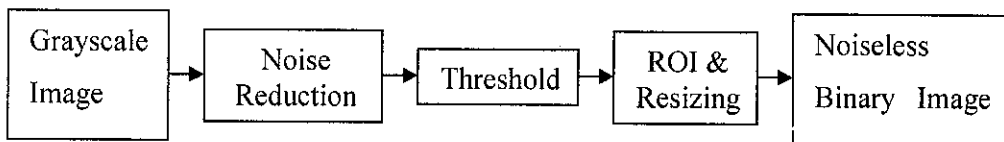


Figure 3.2: Image Processing Module Algorithms

3.5 Feature Extraction Module

These modules consist of one main module and six sub modules. The main module combines feature extraction matrix from the four feature extraction methods used; Modified Alphabet Encoder, Kirsch Edge Detection, Image Compression, and Alphabet Profile Feature. While, the four sub modules will implement feature extraction process. Refer to Appendix B.

Table 3.1: Summary Feature Extraction Modules

M File	Function
Featext.m	Combines all the feature matrix from all the extraction methods
Horin.m	Horizontal Alphabet Regions Encoder
Verti.m	Vertical Alphabet Regions Encoder
Diagonal.m	Diagonal Alphabet Regions Encoder
Kirsch.m	Kirsch Edge Detector
Imgcomp.m	Image Compression
Profiles.m	Profiling Alphabet Smoothness, Width, and Height

3.5.1 Modified Alphabet Encoder Module

This module is written by treating input binary image as a 32×32 matrix. With MATLAB capabilities in manipulating matrix, the horizontal, vertical and diagonal regions can be extracted and computed easily.

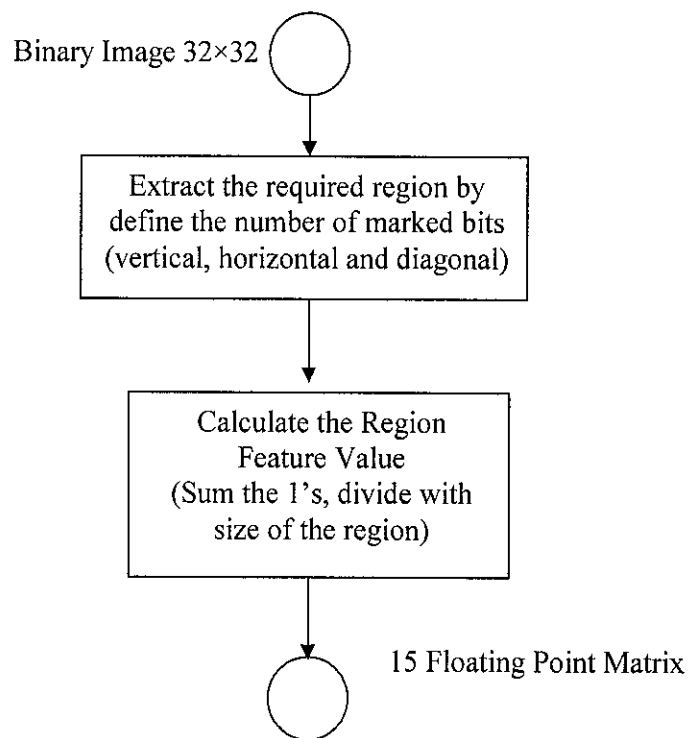


Figure 3.3: Modified Alphabet Module Algorithm

3.5.2 Kirsch Edge Detection Module

To implement this module, we need to use the MATLAB Image Processing Toolbox because it involves image convolution.

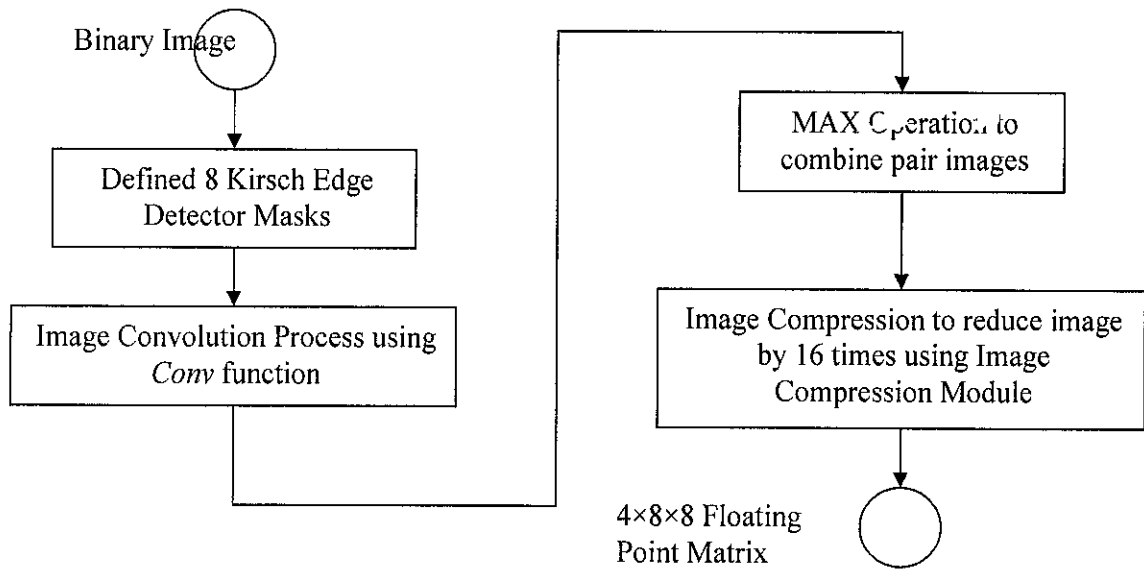


Figure 3.4: Kirsch Edge Detection Module Algorithm

3.5.3 Image Compression Module

The algorithm of this module is implemented based on the image compression formula.

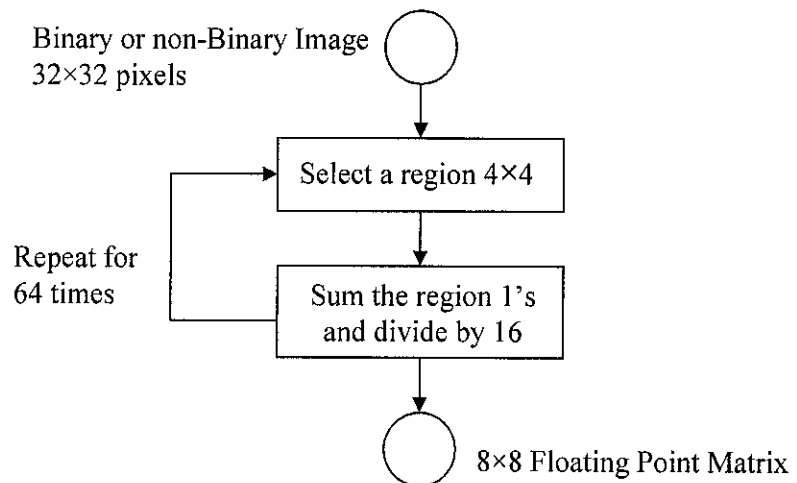


Figure 3.5: Image Compression Module Algorithm

3.5.4 Alphabet Profile Feature Module

The module will obtain the edge profile of the alphabet by scanning line by line. Edge profile data will be differentiated to obtain the smoothness alphabet profile.

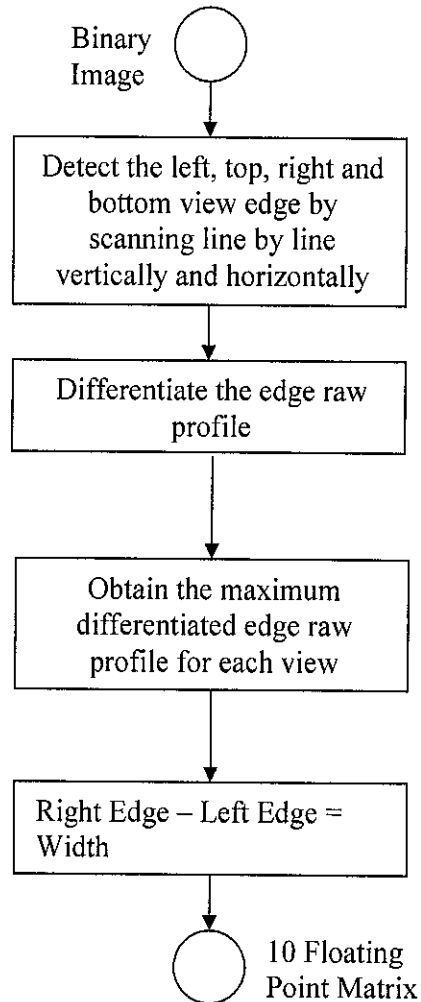


Figure 3.6: Alphabet Feature Module Algorithm

3.6 Neural Network Architecture

To create the MLFF network, the network architecture must be well defined. The number of neurons in the input layer is decided by the number of pixel in the bitmap. The bitmap in handwriting recognition system consists of 345 pixels. Thus, we need 345 input neurons. The output layer has 26 neurons, one neuron for each handwritten character to be recognized. As for the hidden layer, the neurons in this layer cannot be observed through the input or output behavior of the network. Complex patterns cannot be detected by a small number of hidden neurons; however too many of them can dramatically increase the computational burden. Furthermore, the greater the number of hidden neurons, the greater the ability of the network to recognize existing patterns. However, if the number of hidden neurons is too big, the network might simply memories all training examples. Hence, we decide to fit 20 neurons in the hidden layer and it bring to 20×26 size of MLFF network was created [2].

For the network training part, the MLFF network is trained with 30 sets of alphabets. For each training set, the training parameters are set as listed in Table 3.2. The training parameter is set prior to the training phase. The network is trained using backpropagation algorithm.

Table 3.2: Training Parameters

Training Parameters	Command Line
Performance Function = Sum-Squared Error	net.performFcn = 'sse'
Goal: 0.01	net.trainParam.goal = 0.01
Epochs: 5000	net.trainParam.epochs = 5000
Momentum: 0.95	net.trainParam.mc = 0.95

3.7 MATLAB GUI Module

This module is built using the MATLAB GUIDE (GUI Development Environment). By providing an interface between the user and the application's underlying code, GUIs enable the user to operate the application without knowing the commands that would be required by a command line interface. For this reason, applications that provide GUIs are easier to learn and use than those that are run from the command line. The sections that follow describe how to create GUIs with GUIDE. This includes laying out the components, programming them to do specific things in response to user actions, and saving and opening the GUI.

3.7.1 Creating GUI with Guide

MATLAB implements GUIs as figure windows containing various uicontrol objects. Each object must be programmed to perform the action we intend it to do when a user activates the component. In addition, GUI must be able to save and run. All of these tasks are simplified by GUIDE, the MATLAB graphical user interface development environment.

GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the callbacks where the functions that execute when users activate components in the GUI [4].

3.7.2 The Layout Editor

The Layout Editor enables creator to select GUI components (uicontrol objects) from the component palette, at the left side of Layout Editor, and arrange them in the layout area, to the right. When the Run button is pressed, the functioning GUI appears outside the Layout Editor. Figure 3.7 shows the Layout Editor for handwriting recognition system window.

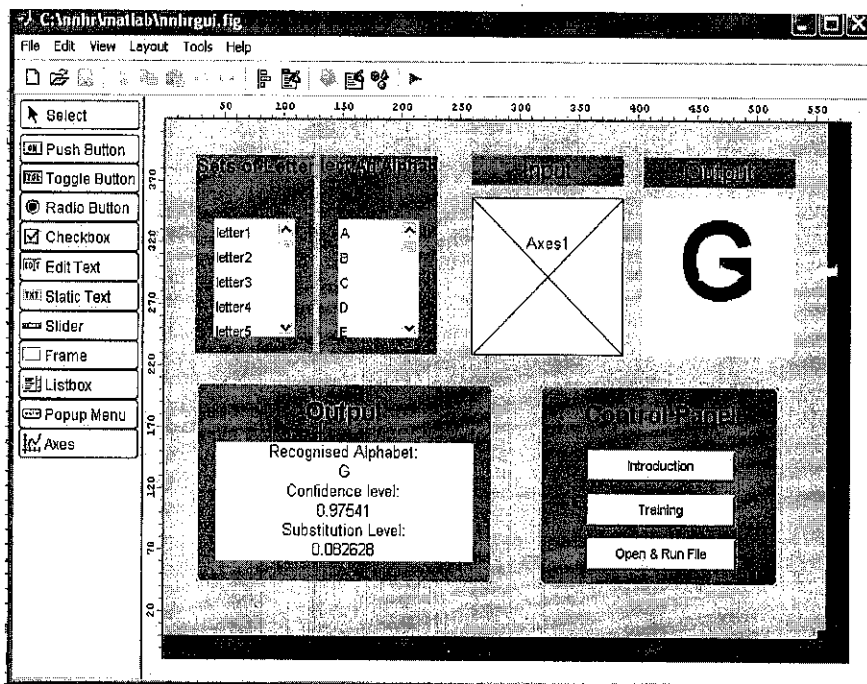


Figure 3.7: GUI Layout Editor

The Property Inspector shown in Figure 3.8 enables creator to set the properties of the components in your layout. It provides a list of all settable properties and displays the current value. Each property in the list is associated with an editing device that is appropriate for the values accepted by the particular property. For example, a color picker to change the BackgroundColor, a pop-up menu to set FontAngle, and a text field to specify the Callback string.

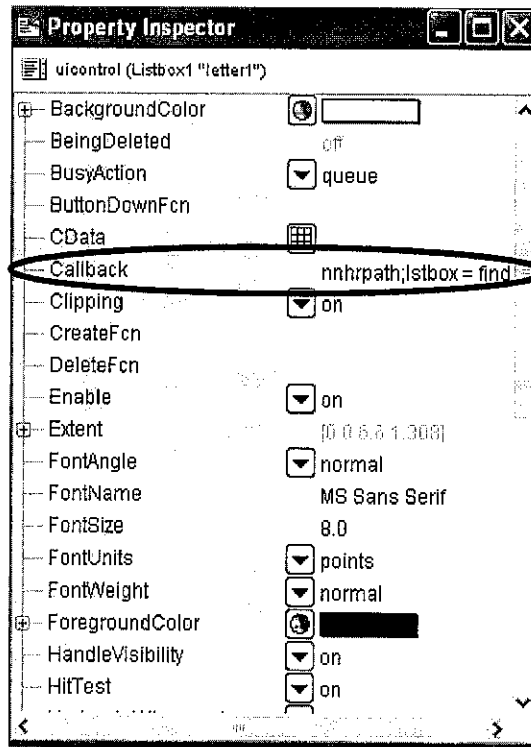


Figure 3.8: Property Inspector

When a user activates a component of the GUI, the GUI executes the corresponding callback. The name of the callback is determined by the component's Tag property. For example, a push button with the Tag print_button executes the callback. Refer to Figure 3.8. Once the push button is activated or whenever the button is pushed, MATLAB will run the callback code [4].

3.8 Development of the Handwriting Recognition System

In the process of handwriting recognition development, some statistical studies have to be done to test the network ability in recognition. The handwriting recognition system was developed through several improvements and modifications in order to increase its accuracy and reliability. Figure 3.9 shows the actual development of the system. The result can be referred to Chapter 4.

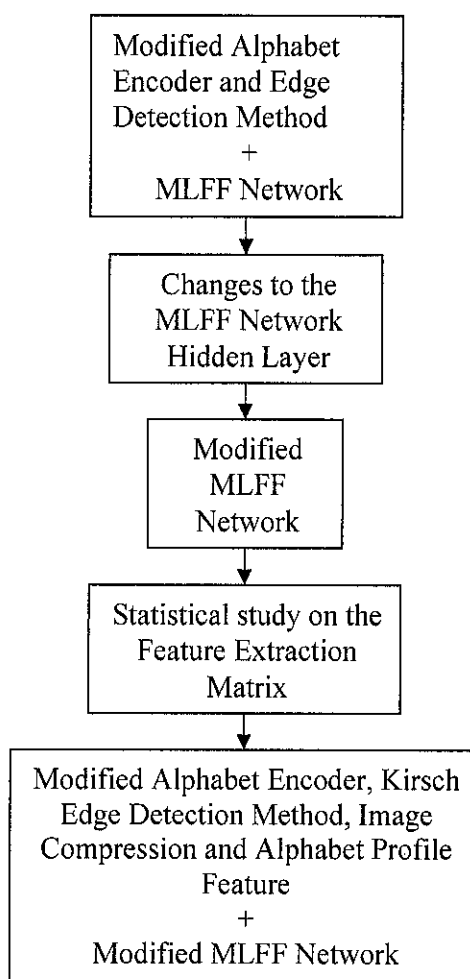


Figure 3.9: Handwriting Recognition System Development

CHAPTER 4

RESULT AND DISCUSSION

4.1 Identify the Best Value for LOC and LOS

Statistical study was done to identify and choose the best value for level of confidence (LOC) and level of substitution (LOS). These values are important because it will affect the recognition performance of the system. The experiment was done by performing the network simulation using different value of LOC and LOS. The simulation was done on a program which is written to recognize 26 upper case alphabets. (Refer to Appendix C.4 for the program). The values chosen were depending on the recognition accuracy. Values of LOC and LOS which give the highest recognition accuracy will be identified. Please refer to Table 4.1 for the results.

Table 4.1: Results for Recognition Accuracy

% Recognition Accuracy		Level of Confidence				
		0.80	0.85	0.90	0.95	1.00
Level of Substitution	0.10	88.4615	92.3077	92.3077	88.4615	84.6154
	0.15	73.0769	80.7692	88.4615	84.6154	84.6154
	0.20	76.9231	92.3077	96.1538	88.4615	84.6154
	0.25	84.6154	76.9231	88.4615	84.6154	84.6154
	0.30	84.6154	80.7692	92.3077	84.6154	84.6154

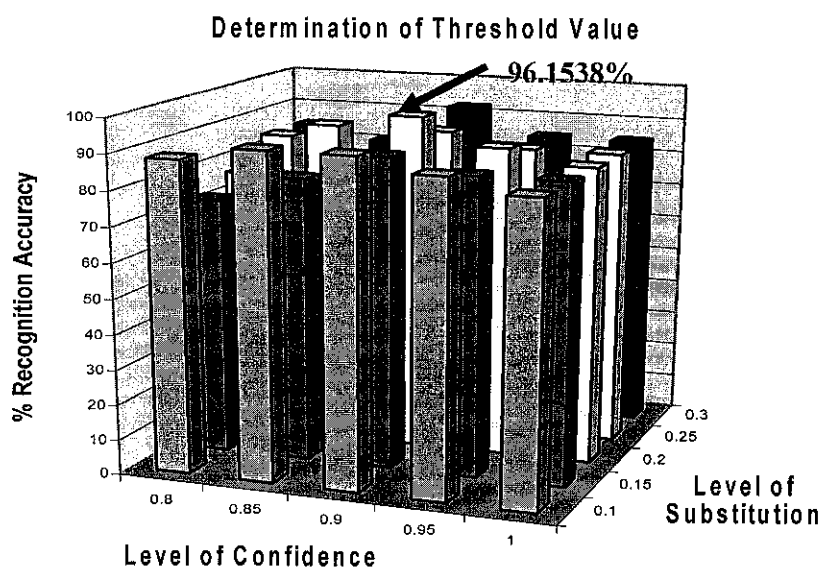


Figure 4.1: Determination of LOC and LOS

From Table 4.1 and Figure 4.1, we can say that the best value for level of confidence is 0.9 and value for level of substitution is 0.2. Both of these values give the highest percentage of recognition accuracy. Hence these values will be used as a threshold value to compare with the simulation result for this project.

4.2 Modified Alphabet Encoder and Edge Detection Method with MLFF Network

The first step of handwriting recognition system development is to create an initial handwriting recognition system by combining two feature extraction methods. The two feature extractors are modified alphabet encoder and edge detection. Modified alphabet encoder forms a 15×1 feature matrix, while edge detection method supplies another 36×1 matrix, which gives to the neural network a 51×1 input matrix.

As for the neural network, the MLFF network with size of 20×26 neurons is used. The network is trained using BP algorithm. The total samples trained are 780 alphabets (30 sets of A to Z).

Then, the accuracy of the neural network is tested to recognize 780 trained alphabet and 175 untrained alphabets.

Table 4.2: System Accuracy Result

Input		Neural Network	Output Accuracy
Modified alphabet encoder	15×1	MLFF network: 20×26	Untrained Samples: 29.6970 %
Edge Detection	36×1	Training:	Trained Samples:
Total	51×1	BP algorithm	57.6923 %

In the training process, the network can not be trained to reach the desired goal except for the first sample presented. As a result, the network provide high error rate. Hence, the next step is to add more neurons or hidden layer to the network.

4.3 Adding another Hidden Layer

A hidden layer had added to the network. The numbers of neurons added are 15. Thus, the size of the neural network now becomes $20 \times 15 \times 26$ neurons. As for the input and feature extraction methods, they remain the same. The network is then tested again using same inputs.

Table 4.3: System Accuracy Results

Input		Neural Network	Output Accuracy
Modified alphabet encoder	15×1	MLFF network: $20 \times 15 \times 26$	Untrained Samples: 21.2121 %
Edge Detection	36×1	Training:	Trained Samples:
Total	51×1	BP algorithm	43.0769 %

Table 4.3 shows that by adding a hidden layer could not help in increasing the recognition accuracy. As for the training process, the network can not be trained to reach the desired goal.

4.4 Modified MLFF Network

Based on the two systems in Section 4.1 and 4.2, the overall system accuracy is very low. These maybe cause by feature extraction methods and the neural network itself. Thus, the neural network is tested again by replaced the neural network with Modified MLFF network. In Modified MLFF network, each of the networks was trained with each pattern.

Table 4.4: System Accuracy Results

Input		Neural Network	Output Accuracy
Modified alphabet encoder	15×1	Modified MLFF network: Sets of 20×26 networks Training: BP algorithm	Untrained Samples: 32.1212 %
Edge Detection	36×1		Trained Samples:
Total	51×1		69.7436 %

From Table 4.4, shows that the change of the neural network has improved the accuracy rate on the untrained samples as well as trained samples. Furthermore, the network training goal is met easily. However, the accuracy of the system can be further improved by using good feature extractors. Hence, the next step in the handwriting recognition system is to select good feature extractors.

4.5 Statistical Study on the Feature Extraction Matrix

Six types of feature extraction methods are used. They are Modified Alphabet Encoder, Edge Detection Method, Kirsch Edge Detection, Image Compression, Line Intersection Feature, and Alphabet Profile Feature.

The purpose of this statistical study is to analyze the feature matrix for each of the 26 classes of alphabet. There are two statistical tests implemented. The first is standard deviation test on the same classes. The second is standard deviation test on the different classes.

4.5.1 Standard Deviation Test on the Same Classes

This is to test the reliability of the feature extraction method. We know that features for the same class should be similar to each other. Therefore, we want to look for a low standard deviation.

Table 4.5: Standard Deviation Test Results

Input	Test	Output	Comment
Training Sets: 754 alphabets	Standard deviation on same classes	Average Standard Deviation	
	Modified Alphabet Encoder	0.1260	Good
	Edge Detection Method	0.1371	Ok
	Kirsch Edge Detection	0.1060	Good
	Line Intersection	0.1516	Ok
	Image Compression	0.2515	Ok
	Alphabet Profile Feature	0.1570	Ok

4.5.2 Standard Deviation Test on the Different Classes

The purpose of this test is to determine the degree of discrimination among the different classes. Therefore, we want to look for high standard deviation because features from different classes are significantly different.

Table 4.6 Standard Deviation Test Results

Input	Test	Output	Comment
Training Sets: 754 alphabets	Standard deviation on different classes	Average Standard Deviation	
	Modified Alphabet Encoder	0.1941	Ok
	Edge Detection Method	0.2177	Ok
	Kirsch Edge Detection	0.1291	Ok
	Line Intersection	0.1925	Ok
	Image Compression	0.3751	Good
	Alphabet Profile Feature	0.2782	Good

Based on the results above, feature extraction method that gave the good performance in standard deviation test on the same classes and different classes will be identified. Hence, Kirsch Edge Detection, Image Compression, Modified Alphabet Encoder, and Alphabet Profile Feature methods are chosen.

4.6 Final Handwriting Recognition System

The final proposed handwriting recognition system incorporates Modified Alphabet Encoder, Kirsch Edge Detection Method, Image Compression, and Alphabet Profile Feature with Modified MLFF as the network. The system is tested with the trained and untrained samples yet again. Table 4.7 shows the results of the simulation.

Table 4.7 Final Proposed Handwriting System Accuracy

Input		Neural Network	Output Accuracy
Modified alphabet encoder	15×1	Modified MLFF network: Sets of 20×26 networks Training: BP algorithm	Untrained Samples: 76.6127 %
Kirsch Edge Detection	256×1		
Image Compression	64×1		Trained Samples: 91.6127 %
Alphabet Profile Feature	10×1		
Total	345×1		

The system has an acceptable accuracy rate for the untrained and trained samples. However, misclassification can be looked into to increase accuracy. A further improvement has to be implemented to improve the recognition performance of the system.

4.7 Recognition Error Analysis

Most of the recognition error comes from failure to recognize a few specific letters such as F, D, G, and N.

The system has the highest failure rate when it is presented with letter F. The main reason of the failure is the system basically fails to differentiate letter F from P. The same situation occurs when recognizing letter D. This is again it fails to differentiate letter D from O. The system also fails to recognize letter N from H. The suspected main reason of recognition error is because the system fails to detect curves lines in letters such as D, G, P, and O. One of the unique features which have not been analyzed is recommended to solve this problem. The method mentioned is Euler method. It involves in extracting the image by considering the number of island of each unique character. Notice that most of the characters failed to recognize have same characteristic, where all of them have one island. Thus, by adding Euler method in the feature extraction will help in increasing the recognition accuracy.

4.8 Testing and Simulation Network Modules

The testing and simulation module combines the image processing, feature extraction, and neural network module together as a handwriting recognition system. It also supplies input from any grayscale alphabet bitmap file to the system. To test network accuracy, the trained and untrained sets of alphabets were used as the input to the neural network. Refer Appendix C4, C5 and C6 for further details on the modules and the results.

4.9 Identified Neural Network Modules

The Neural Network modules have two main modules. The first module is to define and train the neural network. The second is for test and simulation purposes. Table 4.8 shows the modules involved.

Table 4.8: Summary of Neural Network Module

M File	Function
Nnhr1.m	Defining the training feature matrix and target
Nnhr2.m	Training the network
Nnhr3.m	Defining the network architecture
Simhr1.m	Simulate one set of alphabet A~Z
Simhr2.m	Simulate one alphabet
Simhr3.m	Simulate the training set (30 sets of alphabet A~Z)
Simhr4.m	Simulate the untrained sets (195 letters).

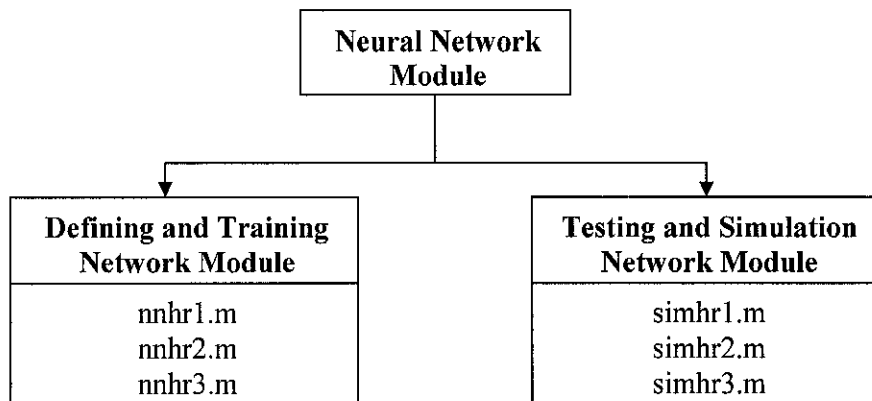


Figure 4.2: Neural Network Module

The final handwriting recognition system consists of four main modules. They are Image Processing Module, Feature Extraction Module, Neural Network Module and MATLAB GUI Module as shown in figure 4.3. Each module has its own functionality in the handwriting recognition system.

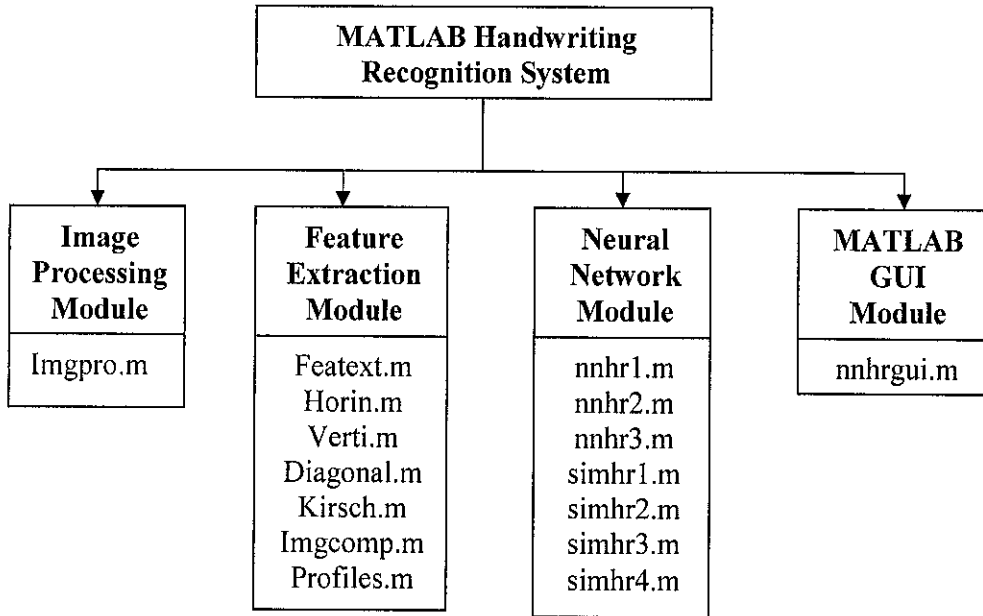


Figure 4.3: Final Handwriting Recognition Module

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Handwriting recognition has become an important field of research, because of its potential to overcome the obstacles placed by current interface methods and its power to further integrate computers into everyday life. The benefits of using artificial neural network for the purpose of handwriting recognition become clearer as technical advances are made. Neural network based character recognition is a viable procedure for large scale document processing applications. However, to be successful, it has to be tightly coupled with image processing and feature extraction components. Faulty bitmap extraction can render a highly efficient network recognizer useless.

This project is completed with the help of MATLAB. It has shown that MATLAB is a very powerful software tool to develop a complex system yet easy to program. With a short and simple coding, a constrained and isolated handwriting recognition system for English uppercase alphabets has been implemented. Based on the research and statistical study, it shows that with good feature extraction and neural network scheme will bring to high recognition accuracy. It has also been shown that there is a need to select good feature extractor so that unique features of an alphabet can be highlighted. Furthermore, the MATLAB GUI is a very useful tool to display the handwriting recognition output or results.

With the combination of image processing, feature extraction and artificial neural network, this project able to achieve the recognition accuracy at 70%. Hence, some improvement and modification has to be made to increase the system ability and performance.

5.2 Recommendations

The system accuracy can be increased through collection of a large number of handwriting samples. This is vital to the system because it needs to be trained with a variety of handwriting so that it is able to recognize different types of handwriting. As for now, the system is only able to recognize constrained type of handwriting because the training set used is from the constrained type of handwriting. Thus, the system will perform poorly when it is presented with the unconstrained handwriting. The recommended size for the handwriting database is around 1000 sets.

Another factor to consider is the type of neural network because it can affect accuracy greatly. The neural networks that had been explored are Multi Layer Feed forward Network (MLFF) and Modified MLFF network. Even though the Modified MLFF network has its own advantage, further analysis of different neural network architecture can improve the system ability. Unsupervised and combination of supervised and unsupervised neural network such as Self-Modified Mapping (SOM), Fuzzy Adaptive Resonant Theory (Fuzzy ARTMAP), and Learning Vector Quantization (LVQ) can be studied.

Since this project is only designed to recognize 26 isolated English upper case A to Z, it is recommended that modifications and improvements be made to this project to recognize a word instead of a character. This modification can be complemented with efficient techniques of feature extraction and character segmentation. The word is separated automatically into isolated character before it is used for recognition. This will be an interesting handwriting recognition project.

REFERENCES

- [1] Demuth and Beale. 1998, *Neural Network Toolbox User's Guide*, MA, The Math Works Inc. pp 1-3.
- [2] Negnevitsky. 2002, *Artificial Intelligence: A Guide to Intelligence Systems*, England, Pearson Education. Pp 163-214.
- [3] 2002, *MATLAB Image Processing Toolbox User Guide*. Version 3. The Math Works Inc.
- [4] 2002, *MATLAB Creating Graphical User Interfaces*. Version 6. The Math Works Inc.
- [5] Andrew T. Wilson. 'Off-line Handwriting Recognition Using Artificial Neural Networks'. University of Minnesota, Morris.
- [6] Il-Seok Oh and Ching Y. Suen, 2000, 'A class-modular feedforward neural network for handwriting recognition'. Department of Computer Science, Chonbuk National University, Chonju, Chonbuk 561-156, South Korea and Centre of Pattern Recognition and Machine Intelligence, Concordia University, Montreal, Quebec, Canada.
- [7] Jung-Hsien Chiang, 1997, 'A hybrid neural network model in handwritten word recognition'. Department of Information Management, Chaoyang of Technology, Taichung, Taiwan.

- [8] Spitz and Dengel. 1995, *Document Analysis Systems*, Singapore, World Scientific.
- [9] Y. Mizukami. 1998, "A Handwritten Chinese Character Recognition System Using Hierarchical Displacement Extraction Based on Directional Features." *Pattern Recognition Letters*. Vol.19. Elsevier Science Ltd.
- [10] J. Dorronsoro G. Fractman. 'Large Scale Neural Form Recognition'. Spain.
- [11] Toshihiro Suzuki. 'A Handwritten Character Recognition System by Efficient Combination Of Multiple Classifiers'.
- [12] Internet Sources:
1. <http://www.ph.tn.tudelft.nl/pr-intro.html>
 2. <http://www.elsevier.com>
 3. <http://www.cairo.utm.my>
 4. <http://www.mathworks.com>
 5. <http://www.idiap.ch>

APPENDIXES

Appendix A Image Processing Module

A.1 `Imgpro.m` Function Script

```
%Input   : Grayscale bitmap image
%Output  : Noiseless binary image
%Process : Image Processing using noise reduction, ROI,
resizing and thresholding
function imgalpha = imgpro(gryimg)

thresb = [];
rec = [];
imgalpha = imread(gryimg);           %read image file
imgalpha = wiener2(imgalpha,[5 5]);  %reduce noise
imgalpha = not(im2bw(imgalpha,0.5)); %threshold image
                                     at 0.5

[y,x] = find(imgalpha);
rec = [min(x) min(y) max(x)-min(x) max(y)-min(y)];
if rec(3) < 12;
    rec(3) = 12;
    [cgx,cgy] = center(imgalpha);
    rec(1) = cgy - 6;
end
imgalpha = imcrop(imgalpha,rec);     %ROI analysis
imgalpha = imresize(imgalpha,[32 32],'nearest');
                                     %resizing image to 32X32
imshow(imgalpha);
```

Appendix B Feature Extraction Module

B.1 Featext.m Function Script

```
%Input   : Image filename
%Output  : Feature matrix
%Process : Feature extraction - modified alphabet
%encoder, modified alphabet encoder, image compression
%and kirsch edge detection
function p = featext(pixalpha)

alpha = imgpro(pixalpha);      %read image alphabet
h = horin(alpha);
v = verti(alpha);
d = diagonal(alpha);          %modified alphabet encoder
pro = profiles(alpha);        %modified alphabet encoder
img = imgcomp(alpha);         %image compression
img = im2col(img,[8 8],'distinct');
[hd,vd,rdd,ldd] = kirsch(alpha); %kirsch edge detection
p = [h,v,d,hd,vd,rdd,ldd,img',pro]';%neural network input
```

B.2 Horin.m Function Script

```
%Input   : Image matrix 32X32
%Output  : Horizontal feature matrix 1X3
%Process : Extract horizontal feature for Modified
Alphabets Encoder
function h = horin(alpha)
h1 = [];
h2 = [];
h3 = [];                        %initialized horizontal matrix

h1 = alpha(1:9,:);              %find the h1 region
h1 = sum(h1);
h1 = h1';
h1 = sum(h1)/288;               %sum all 1

h2 = alpha(10:23,:);           %find the h2 region
h2 = sum(h2);
h2 = h2';
h2 = sum(h2)/448;               %sum all 1

h3 = alpha(24:32,:);           %find the h3 region
h3 = sum(h3);
h3 = h3';
h3 = sum(h3)/288;               %sum all 1
h = [h1 h2 h3];                 %horizontal matrix
```


B.3 Verti.m Function Script

```
%Input   : Image matrix 32X32
%Output  : Vertical feature matrix 1X8
%Process : Extract vertical feature for Modified Alphabet
Encoder
function v = verti(alpha)
v1 = [];
v2 = [];
v3 = [];
v4 = [];
v5 = [];
v6 = [];
v7 = [];
v8 = []; %initialized vertical matrix
v1 = alpha(1:13,1:10); %find the v1 region
v1 = sum(v1);
v1 = v1';
v1 = sum(v1)/130; %sum all 1 divide region size
v2 = alpha(1:19,11:22); %find the v2 region
v2 = sum(v2);
v2 = v2';
v2 = sum(v2)/228; %sum all 1 divide region size
v3 = alpha(1:13,23:32); %find the v3 region
v3 = sum(v3);
v3 = v3';
v3 = sum(v3)/130; %sum all 1 divide region size
v4 = alpha(14:32,1:10); %find the v4 region
v4 = sum(v4);
v4 = v4';
v4 = sum(v4)/190; %sum all 1 divide region size
v5 = alpha(20:32,11:22); %find the v5 region
v5 = sum(v5);
v5 = v5';
v5 = sum(v5)/156; %sum all 1 divide region size
v6 = alpha(14:32,23:32); %find the v6 region
v6 = sum(v6);
v6 = v6';
v6 = sum(v6)/190; %sum all 1 divide region size
v7 = alpha(1:15,20:32); %find the v7 region
v7 = sum(v7);
v7 = v7';
v7 = sum(v7)/195; %sum all 1 divide region size
v8 = alpha(20:32,1:15); %find the v8 region
v8 = sum(v8);
v8 = v8';
v8 = sum(v8)/195; %sum all 1 divide region size
v = [v1,v2,v3,v4,v5,v6,v7,v8]; %vertical matrix
```

B.4 Diagonal.m Function Script

```
%Input    : Image matrix 32X32
%Output   : Diagonal feature matrix 1X4
%Process  : Extract diagonal feature for Modified Alphabet
Encoder
function d = diagonal(alpha)

d1 = [];
d2 = [];
d3 = [];
d4 = [];           %initialized diagonal matrix

temp = [];
for n = 1:4
    temp = alpha(n,1:n+4);
    d1 = [d1 temp]; %find the d1 region from row 1 to 8
    temp = alpha(n,29-n:32);
    d2 = [d2 temp]; %find the d2 region from row 1 to 8
end

for n = 5:12
    temp = alpha(n,n-4:n+4);
    d1 = [d1 temp]; %find the d1 region from row 9 to 24
    temp = alpha(n,29-n:37-n);
    d2 = [d2 temp]; %find the d2 region from row 9 to 24
end

for n = 13:16
    temp = alpha(n,n-4:16);
    d1 = [d1 temp]; %find the d1 region from row 25 to 32
    temp = alpha(n,17:37-n);
    d2 = [d2 temp]; %find the d2 region from row 25 to 32
end

d1 = d1';
d1 = sum(d1)/size(d1,1); %sum all 1 divide d1 size
d2 = d2';
d2 = sum(d2)/size(d2,1); %sum all 1 divide d2 size

for n = 17:20
    temp = alpha(n,29-n:16);
    d3 = [d3 temp]; %find the d3 region from row 33 to 40
    temp = alpha(n,17:n+4);
    d4 = [d4 temp]; %find the d4 region from row 33 to 40
end

for n = 21:28
    temp = alpha(n,29-n:37-n);
    d3 = [d3 temp]; %find the d3 region from row 41 to 56
```

```

    temp = alpha(n,n-4:n+4);
    d4 = [d4 temp]; %find the d4 region from row 41 to 56
end

temp = [];
for n = 29:32
    temp = alpha(n,1:37-n);
    d3 = [d3 temp]; %find the d3 region from row 57 to 64
    temp = alpha(n,n-4:32);
    d4 = [d4 temp]; %find the d4 region from row 57 to 64
end

d3 = d3';
d3 = sum(d3)/size(d3,1); %sum all 1 divide d3 size
d4 = d4';
d4 = sum(d4)/size(d4,1); %sum all 1 divide d4 size

d = [d1 d2 d3 d4]; %diagonal matrix

```

B.5 Kirsch.m Function Script

```

%Input : Image matrix 32X32
%Output : Kirsch Edge feature matrix 1X320
%Process : Extract edge feature for Kirsch Edge Detector

function [hd,vd,rdd,ldd] = kirsch(alpha)

mh1 = 1/15 * [ 5 5 5
              -3 0 -3
              -3 -3 -3];
mh2 = 1/15 * [ -3 -3 -3
              -3 0 -3
              5 5 5]; %horizontal mask matrix
mv1 = 1/15 * [ -3 -3 5
              -3 0 5
              -3 -3 5];
mv2 = 1/15 * [ 5 -3 -3
              5 0 -3
              5 -3 -3]; %vertical mask matrix
mrd1 = 1/15 * [ -3 5 5
              -3 0 5
              -3 -3 -3];
mrd2 = 1/15 * [ -3 -3 -3
              5 0 -3
              5 5 -3]; %right diagonal mask matrix
mld1 = 1/15 * [ -3 -3 -3
              -3 0 5
              -3 5 5];

```

```

mld2 = 1/15 * [ 5 5 -3
               5 0 -3
               -3 -3 -3];    %left diagonal mask matrix

hd1 = conv2(alpha,mh1,'same');
hd2 = conv2(alpha,mh2,'same');
vd1 = conv2(alpha,mv1,'same');
vd2 = conv2(alpha,mv2,'same');
rdd1 = conv2(alpha,mrd1,'same');
rdd2 = conv2(alpha,mrd2,'same');
ldd1 = conv2(alpha,mld1,'same');
ldd2 = conv2(alpha,mld2,'same');    %image convolution

hd = max(hd1,hd2);
vd = max(vd1,vd2);
rdd = max(rdd1,rdd2);
ldd = max(ldd1,ldd2);                %max operation

figure(2);
imshow(hd);
figure(3);
imshow(vd);
figure(4);
imshow(rdd);
figure(5);
imshow(ldd);                %show image convolution results

hd = imgcomp(hd);
vd = imgcomp(vd);
rdd = imgcomp(rdd);
ldd = imgcomp(ldd);        %image compression

hd = im2col(hd,[8 8],'distinct');
vd = im2col(vd,[8 8],'distinct');
rdd = im2col(rdd,[8 8],'distinct');
ldd = im2col(ldd,[8 8],'distinct');    %change image
                                       matrix 8X8 to 64X1

hd = hd';
vd = vd';
rdd = rdd';
ldd = ldd';                %kirsch feature matrix

```

B.6 Impcomp.m Function Script

```
%Input   : Image matrix 32X32
%Output  : Compressed Image 8X8
%Process : Compress Image 32X32 to 8X8 using the
compression formula
function out = imgcomp(in)

out = [];

for i = 1:8
    for j = 1:8
        temp = in(4*i-3:4*i,4*j-3:4*j);
        temp = sum(temp);
        out(i,j) = sum(temp')/16;    %implementing the
                                     compression formula
    end
end
```

B.7 Profiles.m Function Script

```
%Input   : Image matrix 32X32
%Output  : Alphabet profile feature matrix 1X10
%Process : Extract smoothness, width, and height alphabet
profile

function pro = profiles(alpha)

r = ones(1,32)*32;
t = ones(1,32)*32;
l = ones(1,32)*32;
b = ones(1,32)*32;    %initialize right, top, left,
                       and bottom row profile

for n1 = 1:32
    for n2 = 1:32    %check to find the edge from
                    right boundary
        if alpha(n1,n2) == 1
            r(n1) = n2-1; %keep the length from
                           right boundary to edge
                           for all rows
        end
        break;
    end
end
end

for n1 = 1:32
```

```

    for n2 = 1:32          %check to find the edge from top
                          boundary
        if alpha(n2,n1) == 1
            t(n1) = n2-1; %keep the length from
                          top boundary to edge
                          for all columns

            break;
        end
    end
end
end

for n1 = 1:32
    for n2 = 1:32          %check to find the edge from
                          left boundary
        if alpha(n1,33-n2) == 1
            l(n1) = n2-1; %keep the length from
                          left boundary to edge
                          for all rows

            break;
        end
    end
end

for n1 = 1:32
    for n2 = 1:32          %check to find the edge from
                          bottom boundary
        if alpha(33-n2,n1) == 1
            b(n1) = n2-1; %keep the length from
                          bottom boundary to
                          edge for all columns

            break;
        end
    end
end

wid =32-[l(6)+r(6)l(16)+r(16)l(26)+r(26)];%width profile
hei =32-[t(6)+b(6)t(16)+b(16)t(26)+b(26)];%height profile

r = diff(r);
t = diff(t);
l = diff(l);
b = diff(b);              %alphabet smoothness profile

pro = [max(r) max(t) max(l) max(b) wid hei];
pro = pro/32;             %alphabet profile feature matrix

```

Appendix C Neural Network Module

C.1 nnhr1.m Function Script

```
%Input    : A-Z bitmap picture file
%Output   : Each alphabet feature matrix and their target
%Process  : Feature Extraction
%         - Modified Alphabets Encoder
%         - Edge Detection Method
%         Target Matrix - a identity matrix 26X26
function [alphabets,target] = nnhr1()
temp = [];
alpha = [];           %initialize
pixalpha = { 'a.bmp' 'b.bmp' 'c.bmp' 'd.bmp' 'e.bmp'
'f.bmp' 'g.bmp' 'h.bmp' 'i.bmp' 'j.bmp' 'k.bmp' 'l.bmp'
'm.bmp' 'n.bmp' 'o.bmp' 'p.bmp' 'q.bmp' 'r.bmp' 's.bmp'
't.bmp' 'u.bmp' 'v.bmp' 'w.bmp' 'x.bmp' 'y.bmp' 'z.bmp'
}';
for n = 1:26
    temp{n} = featext(pixalpha{n});
end
alphabets =
[temp{1},temp{2},temp{3},temp{4},temp{5},temp{6},temp{7},
temp{8},temp{9},temp{10},temp{11},temp{12},temp{13},temp{
14},temp{15},temp{16},temp{17},temp{18},temp{19},temp{20}
,temp{21},temp{22},temp{23},temp{24},temp{25},temp{26}]
target = eye(26);           %alphabets target
```

C.2 nnhr2.m Function Script

```
%Create 29 MLFF 20X26 neurons
%Train each of the network with each pattern
nnhrpath;
for n = 1:30
    dos(letpat{n});
    [alphabets,targets]=nnhr1; %call alphabet and target
    net=newff(minmax(alphabets),[20
26],{'logsig','logsig'},'traingdx'); %create MLFF
    net.LW{2,1} = net.LW{2,1}*0.001;
    net.b{2} = net.b{2}*0.001;
    net.performFcn = 'sse';
    net.trainParam.goal = 0.001;
    net.trainParam.show = 20;
    net.trainParam.epochs = 5000;
    net.trainParam.mc = 0.95;
    [net,tr] = train(net,alphabets,targets)
    save(nn{n},'net');
```

End

C.3 nnhr3.m Function Script

```
%Input    : Feature matrix
%Output   : Recognized letter
%Process  : Modified MLFF will simulate with a new
%          neural network if the confidence and
substitution level is not met.
function [alphanum,confi,unconfi,num,n] = nnhr3(p)

alphaconfi;
alphaunconfi;
nnhrpath;
alphanum = 27;
confi = alphaconfi(27);
unconfi = alphaunconfi(27);
con = [];
uncon = [];
alphanumber = [];
n = 1;           %initialize

num = randperm(nnum);

while ( (n <= nnum) & ( (confi <= alphaconfi(alphanum)) |
(unconfi >= alphaunconfi(alphanum)) ) )
    load (nn{num(n)}); %load neural network randomly
    a = sim(net,p); %simulate one letter
    [confi,alphanum] = max(a); %y max a - level of
                                confidence
    unconfi = sum(a(1:alphanum-1)) +
sum(a(alphanum+1:26)); %level of substitution
    con(n) = confi; %save confidence, substitution,
                    recognized letter
    uncon(n) = unconfi;
    alphanumber(n) = alphanum;
    n = n + 1; %System will simulate with a new
                neural network
end %if the confidence and
    substitution level is not met.

if (n == (nnum+1)) %if confidence and substitution
                    level below threshold value
[confi,n] = max(con./uncon); %find the pair of highest
                            confidence and the
    confi = con(n); %lowest substitution for all the
                    neural network
    unconfi = uncon(n); %assign confidence,
                        substitution, recognized letter
    alphanum = alphanumber(n);
end
```


C.4 simhr1.m Script

```
%Input   : One set of A-Z image bitmap
%Output  : Detected letters, level of confidence, level
of substitution, accuracy
%Process : Simulation of MMLFF networks for one set of
alphabets (A~Z)
pass = 0;
output = [];
letter = [];
confi = [];
unconfi = [];                                %initialize
pixalpha = { 'A.bmp' 'B.bmp' 'C.bmp' 'D.bmp' 'E.bmp'
'F.bmp' 'G.bmp' 'H.bmp' 'I.bmp' 'J.bmp' 'K.bmp' 'L.bmp'
'M.bmp' 'N.bmp' 'O.bmp' 'P.bmp' 'Q.bmp' 'R.bmp' 'S.bmp'
'T.bmp' 'U.bmp' 'V.bmp' 'W.bmp' 'X.bmp' 'Y.bmp' 'Z.bmp'
}';
allletter = [ 'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K'
'L' 'M' 'N' 'O' 'P' 'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y'
'Z' ]';
for n = 1:26                                %loop 26 time for each
letter
    correct = 1;
    p = featext(pixalpha{n});
    [alphanum,con,uncon] = nnhr3(p);
    letter = allletter(alphanum);           %selecting the letter
                                           according alphanum
    confi{n} = con;
    unconfi{n} = uncon;
    output{n} = [pixalpha{n} ' -> ' letter ];
    if n == alphanum
        pass = pass + 1;                    %sum of letter correct
        correct = 0;
    end
end
output = [output' confi' unconfi']
accuracy = pass / 26 * 100                 %calculate accuracy
```

C.4.1 simhr1.m Recognition Output

output=

'A.bmp -> A'	[0.9364]	[0.0422]
'B.bmp -> B'	[0.9829]	[0.0242]
'C.bmp -> C'	[0.8775]	[0.1250]
'D.bmp -> D'	[0.9828]	[0.0303]
'E.bmp -> E'	[0.8755]	[0.0107]
'F.bmp -> F'	[0.9214]	[0.0052]
'G.bmp -> G'	[0.9958]	[0.0199]
'H.bmp -> H'	[0.9811]	[0.0767]
'I.bmp -> I'	[0.9706]	[0.0231]
'J.bmp -> J'	[0.9886]	[0.0690]
'K.bmp -> K'	[0.9916]	[0.0350]
'L.bmp -> L'	[0.9776]	[0.0187]
'M.bmp -> M'	[0.9908]	[0.0456]
'N.bmp -> N'	[0.9276]	[0.1603]
'O.bmp -> O'	[0.9965]	[0.0477]
'P.bmp -> P'	[0.9870]	[0.0225]
'Q.bmp -> Q'	[0.9331]	[0.1718]
'R.bmp -> R'	[0.9832]	[0.0343]
'S.bmp -> B'	[0.8910]	[0.0102]
'T.bmp -> T'	[0.9577]	[0.0410]
'U.bmp -> U'	[0.9405]	[0.0131]
'V.bmp -> V'	[0.9734]	[0.1081]
'W.bmp -> W'	[0.9926]	[0.0332]
'X.bmp -> X'	[0.9784]	[0.0509]
'Y.bmp -> Y'	[0.9901]	[0.0449]
'Z.bmp -> Z'	[0.9797]	[0.0198]

accuracy =

96.1538

C.5 simhr3.m Script

```
%Input   : 30 set of A-Z image bitmap
%Output  : Detected letters, level of confidence, level
of substitution, accuracy
%Process : Simulate the training set (30 sets of alphabet
A~Z)
nnhrpath;
pass = 0;
pixalpha = { 'A.bmp' 'B.bmp' 'C.bmp' 'D.bmp' 'E.bmp'
'F.bmp' 'G.bmp' 'H.bmp' 'I.bmp' 'J.bmp' 'K.bmp' 'L.bmp'
'M.bmp' 'N.bmp' 'O.bmp' 'P.bmp' 'Q.bmp' 'R.bmp' 'S.bmp'
'T.bmp' 'U.bmp' 'V.bmp' 'W.bmp' 'X.bmp' 'Y.bmp' 'Z.bmp'
}';
allletter = [ 'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K'
'L' 'M' 'N' 'O' 'P' 'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y'
'Z' ]';
for numpat = 1:nnum
    dos(letpat{numpat});
    for n = 1:26          %loop 26 time for each letter
        correct = 1;
        p = featext(pixalpha{n});
        [alphanum,confi,unconfi] = nnhr3(p);
        letter = allletter(alphanum); %selecting the
                                     letter according
                                     alphanum
        if n == alphanum
            pass = pass + 1;    %sum of letter correct
            correct = 0;
        end
    end
    passes{numpat} = pass - 26*(numpat-1);
end
aveaccuracy = pass/(26*nnum)*100
```

C.5.1 simhr3.m Recognition Output

Output=

91.6127

C.6 simhr4 Script

```
%Input   : 30 set of A-Z image bitmap
%Output  : Detected letters, level of confidence, level
of substitution, accuracy
%Process : Simulate the training set (30 sets of alphabet
A~Z)
answer = {      'GOHSIEWYIN'...
                'BEHTEOWKIAK'...
                'THANKHONGHON'...
                'BILLYEHOHMENGHUI'...
                'ONGKARSENG'...
                'LIMYANHUANG'...
                'LEETSEWENG'...
                'LIMKHAIHONG'...
                'LIMCHEEWEE'...
                'NGKOKTHONG'...
                'CHANNEENEE'...
                'HEECHEELEONG'...
                'CHONGTINGWEN'...
                'PANGPINSENG'...
                'TEHKEEPIN'    };

total = 0;
pass = 0;
ntest = size(answer);
output = [];
letter = [];
confi = [];
unconfi = [];      %initialize
allletter = [ 'ABCDEFGHJKLMNOPQRSTUVWXYZ' ];
for numtest = 1:ntest(2)
    path = ['c:\nnhr\alphabets\'answer{numtest}'\*.bmp'];
    numchar = size(dir(path));
    dos(['copy ' path ' c:\nnhr\alphabets']);
    for n = 1:numchar(1)
        pixalpha = [ num2str(n) '.bmp' ];
        p = featext(pixalpha);
        [alphanum,con,uncon] = nnhr3(p);
        letter = allletter(alphanum); %selecting the letter
                                   according alphanum

        confi{total+1} = con;
        unconfi{total+1} = uncon;
        output{total+1} = [pixalpha ' -> ' letter ];
        total = total + 1;
        if answer{numtest}(n) == letter
            pass = pass + 1;          %sum of letter correct
            correct = 0;
        end end end
output = [output' confi' unconfi']
accuracy = pass/total*100
```

C.6.1 simhr4.m Recognition Output

output =

'1.bmp -> G'	[0.8899]	[0.1014]
'2.bmp -> O'	[0.9306]	[0.0097]
'3.bmp -> H'	[0.8783]	[0.0725]
'4.bmp -> S'	[0.8170]	[0.0275]
'5.bmp -> B'	[0.8642]	[0.1751]
'6.bmp -> E'	[0.9050]	[0.1544]
'7.bmp -> D'	[0.1504]	[0.0050]
'8.bmp -> Y'	[0.9489]	[0.0604]
'9.bmp -> I'	[0.9486]	[0.0051]
'10.bmp -> N'	[0.9696]	[0.2302]
'1.bmp -> B'	[0.8814]	[0.1773]
'2.bmp -> E'	[0.9513]	[0.0086]
'3.bmp -> R'	[0.8657]	[0.1394]
'4.bmp -> T'	[0.9715]	[0.0449]
'5.bmp -> E'	[0.9513]	[0.0086]
'6.bmp -> O'	[0.9880]	[0.0517]
'7.bmp -> W'	[0.9081]	[0.0359]
'8.bmp -> K'	[0.9851]	[0.0133]
'9.bmp -> I'	[0.9053]	[0.0338]
'10.bmp -> A'	[0.9939]	[0.0190]
'11.bmp -> K'	[0.9177]	[0.0717]
'1.bmp -> T'	[0.9938]	[0.0188]
'2.bmp -> H'	[0.9421]	[0.0391]
'3.bmp -> A'	[0.8895]	[0.0073]
'4.bmp -> H'	[0.9274]	[0.1106]
'5.bmp -> K'	[0.9861]	[0.1193]
'6.bmp -> H'	[0.9535]	[0.0242]
'7.bmp -> O'	[0.9731]	[0.0089]
'8.bmp -> Q'	[0.4275]	[0.0103]
'9.bmp -> U'	[0.7581]	[0.0782]
'10.bmp -> H'	[0.8098]	[0.0290]
'11.bmp -> C'	[0.9143]	[0.2000]
'12.bmp -> H'	[0.5963]	[0.0561]
'1.bmp -> B'	[0.9853]	[0.0507]
'2.bmp -> I'	[0.9730]	[0.0330]
'3.bmp -> L'	[0.9913]	[0.0506]
'4.bmp -> L'	[0.9343]	[0.0972]
'5.bmp -> Y'	[0.9818]	[0.0859]
'6.bmp -> Y'	[0.9710]	[0.0477]
'7.bmp -> E'	[0.9826]	[0.0344]
'8.bmp -> O'	[0.9149]	[0.0791]
'9.bmp -> H'	[0.9936]	[0.0513]
'10.bmp -> M'	[0.8784]	[0.0295]
'11.bmp -> E'	[0.9826]	[0.0344]
'12.bmp -> N'	[0.9894]	[0.0235]

'13.bmp -> O'	[0.9617]	[0.0055]
'14.bmp -> H'	[0.9936]	[0.0513]
'15.bmp -> U'	[0.9934]	[0.0100]
'16.bmp -> I'	[0.9730]	[0.0330]
'1.bmp -> O'	[0.8602]	[0.0461]
'2.bmp -> O'	[0.0222]	[0.0018]
'3.bmp -> F'	[0.0845]	[0.0073]
'4.bmp -> A'	[0.8455]	[0.0119]
'5.bmp -> A'	[0.9301]	[0.0434]
'6.bmp -> F'	[0.9207]	[0.0974]
'7.bmp -> I'	[0.4741]	[0.0341]
'8.bmp -> F'	[0.1975]	[0.0141]
'9.bmp -> O'	[0.0222]	[0.0018]
'10.bmp -> F'	[0.0845]	[0.0073]
'1.bmp -> L'	[0.9925]	[0.1033]
'2.bmp -> I'	[0.6060]	[0.0113]
'3.bmp -> M'	[0.2279]	[0.0090]
'4.bmp -> Y'	[0.9663]	[0.2164]
'5.bmp -> Q'	[0.0352]	[8.0795e-004]
'6.bmp -> N'	[0.9694]	[0.1715]
'7.bmp -> V'	[0.9789]	[0.0327]
'8.bmp -> K'	[0.4979]	[0.0313]
'9.bmp -> Q'	[0.0352]	[8.0795e-004]
'10.bmp -> N'	[0.9694]	[0.1715]
'11.bmp -> Q'	[0.3002]	[0.0023]
'1.bmp -> L'	[0.9652]	[0.2053]
'2.bmp -> E'	[0.8921]	[0.0844]
'3.bmp -> E'	[0.9732]	[0.0934]
'4.bmp -> T'	[0.9443]	[0.0842]
'5.bmp -> I'	[0.9475]	[0.0445]
'6.bmp -> E'	[0.8138]	[0.0214]
'7.bmp -> U'	[0.8805]	[0.1198]
'8.bmp -> V'	[0.8161]	[0.0259]
'9.bmp -> U'	[0.6893]	[0.0486]
'10.bmp -> G'	[0.9097]	[0.2056]
'1.bmp -> L'	[0.9174]	[0.0289]
'2.bmp -> I'	[0.9234]	[0.0388]
'3.bmp -> M'	[0.9758]	[0.0167]
'4.bmp -> K'	[0.9908]	[0.0636]
'5.bmp -> H'	[0.8566]	[0.2432]
'6.bmp -> A'	[0.9018]	[0.0178]
'7.bmp -> I'	[0.9913]	[0.0225]
'8.bmp -> H'	[0.9357]	[0.0062]
'9.bmp -> C'	[0.9211]	[0.0750]
'10.bmp -> N'	[0.9576]	[0.1672]
'11.bmp -> G'	[0.9736]	[0.0320]
'1.bmp -> L'	[0.9599]	[0.0238]
'2.bmp -> I'	[0.8996]	[0.1661]
'3.bmp -> M'	[0.9970]	[0.0125]
'4.bmp -> C'	[0.9965]	[0.0099]

'5.bmp -> H'	[0.8923]	[0.1283]
'6.bmp -> E'	[0.9875]	[0.0470]
'7.bmp -> E'	[0.9484]	[0.0368]
'8.bmp -> W'	[0.9952]	[0.0040]
'9.bmp -> E'	[0.9875]	[0.0470]
'10.bmp -> E'	[0.9875]	[0.0470]
'1.bmp -> N'	[0.7064]	[0.0258]
'2.bmp -> Q'	[0.9045]	[0.0439]
'3.bmp -> K'	[0.9462]	[0.1139]
'4.bmp -> O'	[0.9742]	[0.0619]
'5.bmp -> K'	[0.8938]	[0.0234]
'6.bmp -> T'	[0.9879]	[0.0363]
'7.bmp -> B'	[0.9360]	[0.0685]
'8.bmp -> D'	[0.8763]	[0.0949]
'9.bmp -> N'	[0.7064]	[0.0258]
'10.bmp -> C'	[0.9077]	[0.1740]
'1.bmp -> C'	[0.9767]	[0.0417]
'2.bmp -> H'	[0.7812]	[0.0209]
'3.bmp -> A'	[0.8875]	[0.0146]
'4.bmp -> N'	[0.9972]	[0.1093]
'5.bmp -> N'	[0.9508]	[0.0958]
'6.bmp -> E'	[0.9861]	[0.0180]
'7.bmp -> E'	[0.8909]	[0.0278]
'8.bmp -> N'	[0.9508]	[0.0958]
'9.bmp -> E'	[0.9861]	[0.0180]
'10.bmp -> E'	[0.8909]	[0.0278]
'1.bmp -> H'	[0.9893]	[0.0085]
'2.bmp -> E'	[0.9116]	[0.1719]
'3.bmp -> E'	[0.9881]	[0.0148]
'4.bmp -> C'	[0.9816]	[0.2336]
'5.bmp -> H'	[0.9893]	[0.0085]
'6.bmp -> E'	[0.9116]	[0.1719]
'7.bmp -> E'	[0.9881]	[0.0148]
'8.bmp -> L'	[0.9146]	[0.0072]
'9.bmp -> E'	[0.9116]	[0.1719]
'10.bmp -> O'	[0.9954]	[0.0071]
'11.bmp -> N'	[0.9928]	[0.0022]
'12.bmp -> G'	[0.9172]	[0.0961]
'1.bmp -> C'	[0.9944]	[0.0450]
'2.bmp -> H'	[0.9710]	[0.0905]
'3.bmp -> O'	[0.8873]	[0.0090]
'4.bmp -> N'	[0.8662]	[0.2304]
'5.bmp -> O'	[0.9788]	[0.0196]
'6.bmp -> T'	[0.9376]	[0.0281]
'7.bmp -> I'	[0.9818]	[0.0262]
'8.bmp -> N'	[0.8662]	[0.2304]
'9.bmp -> O'	[0.9788]	[0.0196]
'10.bmp -> Q'	[0.8925]	[0.0646]
'11.bmp -> E'	[0.9831]	[0.0540]
'12.bmp -> N'	[0.8662]	[0.2304]

'1.bmp -> F'	[0.9206]	[0.0413]
'2.bmp -> A'	[0.9998]	[0.0510]
'3.bmp -> N'	[0.9622]	[0.2405]
'4.bmp -> A'	[0.8363]	[0.0252]
'5.bmp -> P'	[0.8615]	[0.0285]
'6.bmp -> I'	[0.9727]	[0.0269]
'7.bmp -> N'	[0.9144]	[0.1658]
'8.bmp -> S'	[0.9792]	[0.0144]
'9.bmp -> Z'	[0.8683]	[0.1644]
'10.bmp -> N'	[0.9144]	[0.1658]
'11.bmp -> A'	[0.8363]	[0.0252]
'1.bmp -> T'	[0.9682]	[0.0695]
'2.bmp -> E'	[0.9740]	[0.0178]
'3.bmp -> H'	[0.9870]	[0.0103]
'4.bmp -> K'	[0.9415]	[0.0494]
'5.bmp -> E'	[0.9252]	[0.1109]
'6.bmp -> E'	[0.9740]	[0.0178]
'7.bmp -> P'	[0.9919]	[0.0755]
'8.bmp -> I'	[0.9143]	[0.1325]
'9.bmp -> N'	[0.8841]	[0.0134]

accuracy =

76.9697

Appendix D Other Script Files

D.1 Alphaconfi.m Function Script

```
%Define the threshold value for the level of confidence
alphaconfi = [0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9
0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9
0.9 0.9]';
```

D.2 Alphaunconfi.m Function Script

```
%Define the threshold value for the level of substitution
alphaunconfi = [0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2
0.2 0.2 0.2]';
```

D.3 nnhrpath.m Script

```
letter1 = 'copy c:\nnhr\alphabets\letter1\*.bmp
c:\nnhr\alphabets';
letter2 = 'copy c:\nnhr\alphabets\letter2\*.bmp
c:\nnhr\alphabets';
letter3 = 'copy c:\nnhr\alphabets\letter3\*.bmp
c:\nnhr\alphabets';
letter4 = 'copy c:\nnhr\alphabets\letter4\*.bmp
c:\nnhr\alphabets';
letter5 = 'copy c:\nnhr\alphabets\letter5\*.bmp
c:\nnhr\alphabets';
letter6 = 'copy c:\nnhr\alphabets\letter6\*.bmp
c:\nnhr\alphabets';
letter7 = 'copy c:\nnhr\alphabets\letter7\*.bmp
c:\nnhr\alphabets';
letter8 = 'copy c:\nnhr\alphabets\letter8\*.bmp
c:\nnhr\alphabets';
letter9 = 'copy c:\nnhr\alphabets\letter9\*.bmp
c:\nnhr\alphabets';
letter10 = 'copy c:\nnhr\alphabets\letter10\*.bmp
c:\nnhr\alphabets';
letter11 = 'copy c:\nnhr\alphabets\letter11\*.bmp
c:\nnhr\alphabets';
letter12 = 'copy c:\nnhr\alphabets\letter12\*.bmp
c:\nnhr\alphabets';
letter13 = 'copy c:\nnhr\alphabets\letter13\*.bmp
c:\nnhr\alphabets';
```

```

letter14 = 'copy c:\nnhr\alphabets\letter14\*.bmp
c:\nnhr\alphabets';
letter15 = 'copy c:\nnhr\alphabets\letter15\*.bmp
c:\nnhr\alphabets';
letter16 = 'copy c:\nnhr\alphabets\letter16\*.bmp
c:\nnhr\alphabets';
letter17 = 'copy c:\nnhr\alphabets\letter17\*.bmp
c:\nnhr\alphabets';
letter18 = 'copy c:\nnhr\alphabets\letter18\*.bmp
c:\nnhr\alphabets';
letter19 = 'copy c:\nnhr\alphabets\letter19\*.bmp
c:\nnhr\alphabets';
letter20 = 'copy c:\nnhr\alphabets\letter20\*.bmp
c:\nnhr\alphabets';
letter21 = 'copy c:\nnhr\alphabets\letter21\*.bmp
c:\nnhr\alphabets';
letter22 = 'copy c:\nnhr\alphabets\letter22\*.bmp
c:\nnhr\alphabets';
letter23 = 'copy c:\nnhr\alphabets\letter23\*.bmp
c:\nnhr\alphabets';
letter24 = 'copy c:\nnhr\alphabets\letter24\*.bmp
c:\nnhr\alphabets';
letter25 = 'copy c:\nnhr\alphabets\letter25\*.bmp
c:\nnhr\alphabets';
letter26 = 'copy c:\nnhr\alphabets\letter26\*.bmp
c:\nnhr\alphabets';
letter27 = 'copy c:\nnhr\alphabets\letter27\*.bmp
c:\nnhr\alphabets';
letter28 = 'copy c:\nnhr\alphabets\letter28\*.bmp
c:\nnhr\alphabets';
letter29 = 'copy c:\nnhr\alphabets\letter29\*.bmp
c:\nnhr\alphabets';
letter30 = 'copy c:\nnhr\alphabets\letter30\*.bmp
c:\nnhr\alphabets';
letpat = {letter1; letter2; letter3; letter4; letter5;
letter6; letter7; letter8; letter9; letter10; letter11;
letter12; letter13; letter14; letter15; letter16;
letter17; letter18; letter19; letter20; letter21;
letter22; letter23; letter24; letter25; letter26;
letter27; letter28; letter29; letter30};
nn = {'nnhr1' 'nnhr2' 'nnhr3' 'nnhr4' 'nnhr5' 'nnhr6'
'nnhr7' 'nnhr8' 'nnhr9' 'nnhr10' 'nnhr11' 'nnhr12'
'nnhr13' 'nnhr14' 'nnhr15' 'nnhr16' 'nnhr17' 'nnhr18'
'nnhr19' 'nnhr20' 'nnhr21' 'nnhr22' 'nnhr23' 'nnhr24'
'nnhr25' 'nnhr26' 'nnhr27' 'nnhr28' 'nnhr29' 'nnhr30'};
nnum = 30;

```


Appendix F Training Samples

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Appendix G Project Gantt Chart

Appendix G.1 Project Gantt Chart for FYPI

No.	Detail / Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
1	Selection of Project Topics	█																			
2	Preliminary Research Work		█																		
3	Submission of Preliminary Report			█																	
4	Brainstorming and Planning				█																
5	Research, Literature and Program Learning					█	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
6	Submission of Progress Report								█												
7	Training and Testing AI System							█	█	█	█	█	█	█	█	█	█	█	█	█	█
8	Submission of Final Draft												█								
9	Submission of Interim Report														█						
10	Oral Presentation																				█

Appendix G.2 Project Gantt Chart for FYP II

NO.	DETAILS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Handwriting Recognition															
	- Research, Literature and Program Learning															
	- Data Collecting															
	- Training and Testing AI System															
	- Build A Handwriting Recognition System															
2	Submission of Progress Report 1															
3	Submission of Progress Report 2															
4	Submission of Draft Report															
5	Submission of Final Report (soft cover)															

NOTES

The due dates of the following milestones are as follow (after Week 15):

- Submission of Technical Report - 19/11/2004
- Oral Presentation - from 6/12/2004 till 8/12/2004
- Submission of Final Report (hard cover) - 24/12/2004